⟨CodeHex16⟩

unipd.codehex16@gmail.com

Norme di Progetto

Data	12/11/2024
Versione	1.3.0

Sommario

Norme di progetto

Ruoli

Luca Ribon	Redattore, Verificatore
Francesco Fragonas	Redattore
Gabriele Magnelli	Redattore, Verificatore
Filippo Sabbadin	Redattore, Verificatore
Luca Rossi	Verificatore
Yi Hao Zhuo	Verificatore

Registro delle Versioni

Versione	Data	Autore	Cambiamenti	Verificatore
1.3.0	Luca Rossi	24/04/2025	Aggiunte regole di sviluppo	Gabriele Magnelli
1.2.0	23/04/2025	Gabriele Magnelli	Definite le attività per le ultime sezioni e aggiunte sezioni per le metriche di qualità	Francesco Fragonas
1.1.0	10/04/2025	Gabriele Magnelli	Correzioni rispetto alle indicazioni ricevute in RTB	Luca Ribon
1.0.0	06/03/2025	Francesco Fragonas	Sistemazione riferimenti	Gabriele Magnelli
0.6.1	04/03/2025	Francesco Fragonas	Sistemazione indice	Gabriele Magnelli
0.6.0	26/02/2025	Gabriele Magnelli	Conclusa sezione verifica	Luca Ribon
0.5.0	12/11/2024	Luca Ribon	Correzione del contenuto, formattazione e stesura sezione Sviluppo	Luca Rossi

Versione	Data	Autore	Cambiamenti	Verificatore
0.4.0	08/01/2025	Gabriele Magnelli	Stesura nuove sezioni, migliorie e correzioni varie	Yi Hao Zhuo
0.3.0	12/12/2024	Gabriele Magnelli	Redazione sezioni Processi di supporto e Processi organizzativi	Yi Hao Zhuo
0.2.0	30/11/2024	Francesco Fragonas	Redazione Processi di accordo	Filippo Sabbadin
0.1.1	30/11/2024	Francesco Fragonas	Revisione Introduzione	Filippo Sabbadin
0.1.0	12/11/2024	Filippo Sabbadin	Prima stesura	Gabriele Magnelli

Indice

1. Introduzione 1
1.1. Scopo del documento 1
1.2. Scopo del prodotto 1
1.3. Glossario 1
1.4. Riferimenti 2
1.4.1. Riferimenti normativi 2
1.4.2. Riferimenti informativi 2

2. Processi primari 4
2.1. Processo di fornitura 4
2.1.1. Scopo e descrizione 4
2.1.2. Attività 4
2.1.3. Rapporti con il proponente 5
2.1.4. Documentazione prodotta 5
2.1.4.1. Valutazione dei capitolati 5
2.1.4.2. Preventivo dei costi 5
2.1.4.3. Analisi dei requisiti 6
2.1.4.4. Piano di progetto 6
2.1.4.5. Piano di qualifica 6
2.1.5. Strumenti utilizzati 6
2.2. Sviluppo 7
2.2.1. Scopo e descrizione 7
2.2.2. Attività 7
2.2.2.1. Analisi dei requisiti 7
2.2.2.2. Progettazione 8
2.2.2.3. Codifica e Testing 9
2.2.2.4. Regole di sviluppo del codice 9
2.2.2.4.1. Naming delle branch 9
2.2.2.4.2. Messaggi di commit 10
2.2.2.4.3. Protezione dei branch 10
2.2.2.4.4. Label dedicate 10
2.2.2.4.5. Testing 10
2.2.2.4.6. Stile del codice 11
2.2.3. Strumenti usati 11

3. Processi di supporto 12

3.1. Documentazione 12
3.1.1. Scopo e descrizione 12
3.1.2. Attività 12
3.1.2.1. Realizzazione 12
3.1.2.2. Verifica e approvazione 12
3.1.2.3. Aggiornamento 13
3.1.3. Struttura dei documenti 13
3.1.3.1. Intestazione 13
3.1.3.2. Registro delle modifiche 13
3.1.3.3. Indice 14
3.1.3.4. Corpo del documento 14
3.1.4. Documenti del progetto 14
3.1.5. Elenchi puntati 14
3.1.6. Immagini use case 14
3.1.7. Formato delle date 15
3.1.8. Composizione tipografica 15
3.1.9. Strumenti 15
3.2. Gestione della configurazione 15
3.2.1. Scopo e descrizione 15
3.2.2. Attività 15
3.2.3. Versionamento 16
3.2.4. Repository 16
3.2.4.1. Struttura della repository documentazione 16
3.2.5. Sincronizzazione 17
3.2.5.1. Branch 17
3.2.5.2. Pull request 17
3.2.6. Strumenti usati 18
3.3. Gestione qualità 18
3.3.1. Scopo e descrizione 18
3.3.2. Attività 18
3.3.3. Verifica 19
3.3.3.1. Scopo e descrizione 19
3.3.3.2. Analisi statica 19
3.3.3.3. Analisi dinamica 19
3.3.3.3.1. Test 20
3.3.4. Validazione 20
3.3.5. Strumenti usati 21

4. Processi organizzativi 22
4.1. Gestione dei processi 22
4.1.1. Scopo e descrizione 22
4.1.2. Attività 22
4.1.3. Ruoli 22
4.1.4. Ticketing (Issue Tracking System) 24
4.1.5. Strumenti usati 24
4.2. Coordinamento 25
4.2.1. Attività 25
4.2.2. Comunicazioni e Riunioni 25
4.2.3. Verbali 26
4.2.4. Strumenti usati 26
4.3. Miglioramento 26
4.3.1. Scopo e descrizione 26
4.3.2. Attività 27
4.3.3. Strumenti usati 27
4.4. Formazione 27
4.4.1. Scopo e descrizione 27
4.4.2. Attività 27
4.4.3. Strumenti usati 28

5. Metriche e standard per la qualità 29
5.1. Funzionalità 29
5.2. Performance 29
5.3. Affidabilità 30
5.4. Sicurezza 30
5.5. Usabilità 30
5.6. Portabilità 31
5.7. Manutenibilità 31

6. Metriche di qualità 32
6.1. Nomenclatura delle metriche 32
6.2. Metriche per i processi 32
6.2.1. Processi primari 32
6.2.1.1. Fornitura 32
6.2.1.1.1. Budget At Completion(MPC-BAC) 32
6.2.1.1.2. Estimated Cost(MPC-EC) 32
6.2.1.1.3. Completion Cost(MPC-CC) 32

6.2.1.1.4. Actual Cost(MPC-AC) 32
6.2.1.1.5. Earned Value(MPC-EV) 33
6.2.1.1.6. Planned Value(MPC-PV) 33
6.2.1.1.7. Cost Performance Index(MPC-CPI) 33
6.2.1.1.8. Estimate At Completion(MPC-EAC) 33
6.2.1.1.9. Estimate To Complete(MPC-ETC) 34
6.2.1.1.10. Schedule Performance Index(MPC-SPI) 34
6.2.1.1.11. Schedule Variance(MPC-SV) 34
6.2.1.1.12. Cost Variance(MPC-CV) 34
6.2.1.2. Sviluppo 35
6.2.1.2.1. Requirement Stability Index(MPC-RSI) 35
6.2.1.2.2. Technical Debt Ratio(MPC-TDR) 35
6.2.2. Processi di supporto 35
6.2.2.1. Documentazione 35
6.2.2.1.1. Indice di Gulpease(MPC-IG) 35
6.2.2.1.2. Correttezza ortografica(MPC-CO) 35
6.2.2.2. Gestione qualità 36
6.2.2.2.1. Satisfaction of Quality Metrics(MPC-SQM) 36
6.2.2.3. Verifica 36
6.2.2.3.1. Code Coverage(MPC-CCO) 36
6.2.2.3.2. Test Superati in Percentuale(MPC-TSP) 36
6.2.3. Processi organizzativi 37
6.2.3.1. Gestione dei processi 37
6.2.3.1.1. Time Efficiency(MPC-TE) 37
6.3. Metriche per il prodotto 37
6.3.1. Funzionalità 37
6.3.1.1. Copertura Requisiti Obbligatori(MPD-RO) 37
6.3.1.2. Copertura Requisiti Opzionali(MPD-OP) 37
6.3.2. Affidabilità 37
6.3.2.1. Code Coverage(MPD-CC) 37
6.3.2.1.1. Branch Coverage(MPD-BC) 38
6.3.2.1.2. Statement Coverage(MPD-SC) 38
6.3.2.1.3. Failure Tolerance(MPD-FT) 38
6.3.2.1.4. Failure Frequency(MPD-FF) 38
6.3.2.1.5. Mean Time Between Failure(MPD-MTBF) 39
6.3.2.1.6. Disponibilità Sistema(MPD-DS) 39
6.3.3. Usabilità 39
6.3.3.0.1. Tempo di Apprendimento(MPD-TA) 39

6.3.3.0.2. Errori Utente/Azione(MPD-EUA) 39
6.3.3.0.3. Task Success Rate(MPD-TSR) 40
6.3.4. Efficienza 40
6.3.4.0.1. Tempo Risposta API(MPD-TRA) 40
6.3.4.0.2. Memoria Processo(MPD-MP) 40
6.3.4.0.3. Consumo Energetico(MPD-CE) 40
6.3.5. Manutenibilità 40
6.3.5.0.1. Complessità Ciclomatica(MPD-CC) 40
6.3.5.0.2. Debito Tecnico(MPD-DT) 41
6.3.5.0.3. Code Smell Density(MPD-CSD) 41
6.3.5.0.4. Tempo Fix Bug(MPD-TFB) 41
6.3.6. Sicurezza 41
6.3.6.0.1. Tasso di Autenticazione Fallita 41
6.3.6.0.2. Crittografia Dati 42

1. Introduzione

1.1. Scopo del documento

Questo documento ha lo scopo di delineare le principali fasi di sviluppo, i ruoli e le responsabilità dei membri del team CodeHex16*. Al suo interno viene fornita una guida completa per tutte le Practice* adottate dal gruppo e per il Way of Working*, garantendo un approccio strutturato e organizzato alle attività collaborative.

Il documento non si limita a fornire una panoramica iniziale ma si propone come un riferimento dinamico, soggetto a revisioni e aggiornamenti continui. Tale approccio incrementale assicura che il contenuto resti sempre aggiornato rispetto alle esigenze del progetto e alle best practices emergenti, consentendo al gruppo di adattarsi rapidamente a nuovi requisiti o cambiamenti contestuali.

1.2. Scopo del prodotto

Il progetto prevede lo sviluppo di un Chatbot* avanzato, basato su modelli linguistici LLM* (Large Language Models), pensato per migliorare la comunicazione tra aziende fornitrici e i loro clienti. Questo assistente virtuale* permetterà agli utenti di ottenere rapidamente e in modo intuitivo informazioni dettagliate su prodotti o servizi offerti, eliminando la necessità di contattare direttamente l'azienda.

Il sistema includerà anche un'interfaccia dedicata per le aziende fornitrici, offrendo strumenti per gestire i clienti e i documenti di riferimento che contengono le informazioni necessarie. Questi documenti saranno utilizzati dal modello linguistico per generare risposte personalizzate e accurate, garantendo un'esperienza utente ottimale. L'intero sistema sarà accessibile tramite una Webapp*, assicurando una gestione efficiente e una fruizione semplice per tutti gli utenti coinvolti.

1.3. Glossario

Per agevolare la comprensione del presente documento, è stato predisposto un glossario che spiega il significato dei termini specifici utilizzati nel contesto del progetto. Per facilitare la comprensione, questi termini avranno il seguente stile: Esempio*

Le definizioni sono disponibili nel documento Glossario.pdf e possono essere consultate anche tramite la seguente pagina web: `https://codehex16.github.io/glossario`

1.4. Riferimenti

1.4.1. Riferimenti normativi

- Capitolato C7 - Assistente Virtuale Ergon: `https://www.math.unipd.it/~tullio/IS-1/2024/Progetto/C7.pdf` *(ultima consultazione 03-03-2025)*;

1.4.2. Riferimenti informativi

- Sito del gruppo CodeHex16: `https://codehex16.github.io/` *(ultima consultazione 06-03-2025)*;
- Repository della documentazione del progetto: `https://github.com/CodeHex16/documentazione` *(ultima consultazione 06-03-2025)*;
- Valutazione capitolati: `https://codehex16.github.io/docs/1%20-%20candidatura/Valutazione-Capitolati.pdf` *(versione 0.3.0)*;
- Preventivo costi e impegni: `https://codehex16.github.io/docs/1%20-%20candidatura/Preventivo-Costi-e-Impegni.pdf` *(versione 0.2.0)*;
- Analisi dei requisiti: `https://codehex16.github.io/docs/2%20-%20RTB/Analisi-dei-Requisiti.pdf` *(versione 0.9.0)*;
- Piano di progetto: `https://codehex16.github.io/docs/2%20-%20RTB/Piano-di-Progetto.pdf` *(versione 0.7.0)*;
- Piano di Qualifica: `https://codehex16.github.io/docs/2%20-%20RTB/Piano-di-Qualifica.pdf` *(versione 1.0.0)*;
- Standard ISO/IEC 12207:1995: `https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf` *(ultima consultazione 06-03-2025)*;
- Glossario:

- Documento: `https://codehex16.github.io/docs/glossario/glossario.pdf` *(versione 1.0.0)*;
- Pagina web: `https://codehex16.github.io/glossario.html` *(ultima consultazione 06-03-2025)*;

2. Processi primari

2.1. Processo di fornitura

Il processo di fornitura è strutturato in conformità agli esiti previsti dalla clausola 5.2 dello Standard* ISO/IEC 12207:1997. Tale processo include la definizione di requisiti concordati, l'analisi dei rischi associati, e la pianificazione di tempi e costi.

2.1.1. Scopo e descrizione

Il processo di fornitura è finalizzato a garantire la realizzazione di un prodotto o servizio che soddisfi i requisiti concordati tra Proponente* e Committente*. L'accordo tra le parti deve definire in modo chiaro i requisiti, le tempistiche e i costi da rispettare. Prima di stipulare tale accordo, il Fornitore* avrà condotto un'analisi dettagliata del progetto proposto, identificando i rischi correlati e stabilendo le linee guida necessarie per gestirli efficacemente.

2.1.2. Attività

Il processo di fornitura prevede varie attività:
- **Analisi**: attività in cui il gruppo analizza le richieste e le aspettative del proponente considerando contemporaneamente i vincoli di vario genere a cui è sottoposto il team stesso(vincoli temporali, di organizzazione ...). Questa attività culmina con l'individuazione dei vari requisiti da parte del gruppo che vengono inseriti nel documento **Analisi dei Requisiti**;
- **Confronto**: attività in cui il gruppo e il proponente si incontrano per discutere e decidere definitivamente (eccetto casi particolari) i requisiti del prodotto finale;
- **Progettazione**: attività in cui il gruppo studia e definisce l'architettura alla base del progetto;
- **Realizzazione**: attività in cui il gruppo realizza ciò che è stato definito durante l'attività di progettazione;
- **Verifica e Revisione**: attività in cui il gruppo controlla e revisiona il lavoro svolto verificando principalmente che sia stato raggiunto lo standard di qualità previsto e che tutte le funzionalità del prodotto siano pronte e funzionanti;
- **Completamento**: attività in cui il gruppo presenta e consegna il lavoro svolto;

2.1.3. Rapporti con il proponente

Il gruppo CodeHex16 manterrà un dialogo attivo e regolare con il Proponente per tutta la durata del progetto didattico, con l'obiettivo di raccogliere il maggior numero possibile di Feedback* sulla correttezza e qualità del lavoro svolto. La comunicazione si articolerà in due modalità principali:

1 - Scritta (asincrona*) utilizzata per comunicazioni di breve durata, condivisione di verbali e materiale informativo e attività di coordinamento;

2 - Incontri online (sincrona*) utilizzati per chiarimenti sul capitolato, approfondimenti relativi ai casi d'uso e requisiti e feedback sul lavoro svolto;

Il formato testuale è chiaro, ma ci sono metodi di comunicazione che hanno maggiore fluidità e precisione:

I Meeting* saranno organizzati con cadenza variabile e fissati tramite e-mail in base alle necessità riscontrate durante lo sviluppo del progetto. Tutti i dettagli discussi durante questi incontri saranno documentati in verbali, con particolare attenzione alle decisioni prese. I verbali saranno disponibili al seguente link: `Verbali esterni - https://github.com/CodeHex16/documentazione/tree/main/2%20-%20RTB/verbali/esterni`

2.1.4. Documentazione prodotta

In questa sezione viene illustrata la documentazione prodotta dal gruppo nel processo di fornitura, che sarà messa a disposizione del Proponente, Ergon Informatica, e dei Committenti, i professori Tullio Vardanega e Riccardo Cardin.

2.1.4.1. Valutazione dei capitolati

Nel documento `Valutazione Capitolati`, il gruppo ha analizzato tutte le proposte di capitolato, fornendo per ciascuna una breve descrizione, una panoramica dello stack tecnologico previsto e una valutazione finale. La scelta del capitolato è stata effettuata considerando diversi criteri, tra cui l'interesse dei membri del gruppo per il progetto, la sua rilevanza nel contesto lavorativo e la fattibilità complessiva.

2.1.4.2. Preventivo dei costi

Nel documento `Preventivo Costi e Impegni` è stata stabilita una data di consegna stimata del progetto, definita in accordo con tutti i membri del gruppo. La pianificazione tiene conto degli impegni personali di ciascun membro e prevede una stima delle

ore settimanali da dedicare al progetto. Inoltre, dopo aver definito tutti i ruoli, è stata elaborata una tabella con la previsione delle ore che ogni membro deve svolgere per ciascun ruolo, garantendo una rotazione prestabilita per bilanciare equamente il carico di lavoro.

Probabilmente non va messa qui:

2.1.4.3. Analisi dei requisiti

Nel documento `Analisi dei Requisiti`, il gruppo ha definito tutti gli Use Case* e i requisiti, frutto di un'attenta analisi del capitolato e della comprensione dell'utilizzo finale del progetto. Il contenuto di questo documento è stato arricchito grazie alle riflessioni svolte con il referente dell'azienda proponente durante gli incontri svolti.

2.1.4.4. Piano di progetto

Nel documento `Piano di Progetto` è stato pianificato l'avanzamento del progetto suddiviso nei 3 periodi chiave (Candidatura, RTB e PB) con una particolare attenzione agli Sprint* settimanali effettuati. Per ognuno è stato descritto il lavoro svolto, il rendiconto delle ore e dei costi in base ai ruoli assegnati.
Inoltre è presente una retrospettiva per ogni periodo volto al raggiungimento di una milestone.

2.1.4.5. Piano di qualifica

Nel documento `Piano di Qualifica` vengono dichiarati gli obiettivi di qualità che il gruppo si prefigge di raggiungere durante lo sviluppo del progetto. Vengono inoltre descritte le metodologie di verifica e validazione adottate per garantire la qualità del prodotto finale, indicando anche le metriche di qualità utilizzate per misurare il grado di soddisfacimento degli obiettivi prefissati.

2.1.5. Strumenti utilizzati

Per lo svolgimento del progetto abbiamo utilizzato i seguenti strumenti:
- **Telegram*** per la comunicazione all'interno del gruppo;
- **Discord*** per gli incontri interni;
- **Zoom** per gli incontri esterni con il referente dell'azienda Ergon Informatica;
- **GitHub** per organizzare tutti i documenti e file sorgente del progetto tramite un repository;

- **GitHub Issue** per assegnare task* ad ogni membro avendo un rendiconto preciso dei ruoli e delle ore svolte per ogni sprint, con l'assegnazione di label e milestone specifiche;
- **GitHub Project** per visualizzare in modo più ordinato le issue e i loro dettagli;
- **Google Fogli** per organizzare incontri con la compilazione di un calendario settimanale e per fissare le ore svolte avendo una visione generale dell'andamento del progetto;
- **Typst** per la stesura di tutti i documenti e verbali;
- **Canva** per la realizzazione delle presentazioni per i Diari di Bordo settimanali;
- **Notion*** per organizzare appunti e documenti non ufficiali;

2.2. Sviluppo

2.2.1. Scopo e descrizione

Il processo di sviluppo è finalizzato alla realizzazione del prodotto software richiesto dal Proponente, seguendo le specifiche definite nel capitolato d'appalto. Questo processo include tutte le attività necessarie per la creazione del prodotto, dalla progettazione iniziale all'integrazione finale, garantendo che il software soddisfi i requisiti concordati e sia conforme alle aspettative del cliente.

2.2.2. Attività

Questo processo prevede le seguenti attività principali:
- **Analisi dei requisiti**;
- **Progettazione**;
- **Codifica**;
- **Testing**;

L'output atteso dal processo è un prodotto software funzionante, che soddisfi i requisiti concordati e che sia ampiamente testato. Perché rispetti i requisiti concordati la fase di codifica dovrà seguire le linee guida fissate durante l'analisi e di conseguenza durante la progettazione.

2.2.2.1. Analisi dei requisiti

Il processo di sviluppo inizia con l'analisi dei requisiti, durante la quale vengono identificati e definiti i requisiti del sistema. Il prodotto di questa attività è contenuto nel documento `Analisi dei Requisiti`. I requisiti descrivono:

- funzionalità richieste;
- funzionalità di supporto;
- funzionalità di sicurezza;
- funzionalità di interfaccia;

I **casi d'uso** descrivono le interazioni tra il prodotto e gli attori coinvolti, ed aiutano ad individuare ulteriori requisiti. Gli attori individuati sono:
- Amministratore;
- Cliente;
- Fornitore;
- Utente non registrato;

Ogni use case ha la seguente struttura:
- Diagramma del caso d'uso;
- Attori principali;
- Attori secondari se presenti;
- Descrizione;
- Precondizioni;
- Postcondizioni;
- Scenario principale;
- Generalizzazioni, estensioni e inclusioni, se presenti;

In questo modo si possono definire in modo dettagliato i requisiti funzionali, di qualità e di vincolo.

2.2.2.2. Progettazione

La progettazione del sistema è l'attività di definizione dell'architettura del software dal punto di vista logico; in questa fase si decide come soddisfare i requisiti identificati durante l'analisi.

In particolare vanno definiti i componenti software e le loro interazioni, prestando attenzione a mantenerli separati e indipendenti per garantire una maggiore manutenibilità e scalabilità del sistema; in questo passaggio è importante anche definire le unità architetturali.
Inoltre vanno definite le responsabilità che verranno applicate in fase di codifica assicurandosi di mantenere un livello di efficienza* e efficacia* il più alto possibile. L'approccio utilizzato in questa attività sarà sia top-down*, per scomporre il problema in sotto-problemi, sia bottom-up*, per ragionare sui singoli sotto-problemi e integrarli in una soluzione complessiva.

Al termine di questa attività ci si aspetta di avere un'architettura ben definita che preveda:
- Backend*:
  - API* per l'interazione con l'LLM;
  - API per l'interazione con il database;
  - backend per la gestione delle interfacce utente e di configurazione;
- Frontend*:
  - Interfaccia utente per il cliente;
  - Interfaccia per il fornitore;
- Database;

Più nello specifico questa attività si suddividerà in:
- **Progettazione dell'architettura**: in cui viene individuata l'architettura generale del sistema che soddisfi tutti i requisiti hardware e software individuati, inoltre vengono definiti i test di integrazione;
- **Progettazione di dettaglio**: in cui vengono definite in dettaglio le singole componenti del sistema definendo di pari passo i test di unità per garantire un certo standard di qualità e garantendo il corretto funzionamento di tali componenti;

2.2.2.3. Codifica e Testing

In questa attività i Programmatori traducono l'output della Progettazione in codice sorgente*, in modo da integrare ogni unità prevista dall'architettura. Inoltre ogni unità sarà documentata e testata per garantire che soddisfi i requisiti definiti in fase di analisi e progettazione. Nello specifico la documentazione dovrà prevedere la documentazione dedicata all'utente finale e quella dedicata al manutentore. Inoltre nella documentazione verranno integrati anche i dettagli relativi al testing eseguito sulle singole unità.

2.2.2.4. Regole di sviluppo del codice

Il gruppo ha definito norme precise per il processo di sviluppo del codice, volte a garantire uniformità, qualità e tracciabilità durante tutte le fasi di implementazione.

2.2.2.4.1. Naming delle branch

Ogni branch deve essere nominata seguendo la convenzione:
- **oggettoDellaBranch-utente-sprint**

dove l'oggetto rappresenta brevemente l'attività svolta, l'utente indica chi ha creato il branch, e sprint si riferisce allo sprint di appartenenza.

È stata inoltre creata una branch **develop** su cui confluiranno tutte le modifiche, sottoposte a verifica tramite action CI/CD, prima di essere integrate nel branch **main**.

2.2.2.4.2. Messaggi di commit

I messaggi di commit devono essere scritti in italiano e iniziare con uno dei seguenti prefissi standard:
- **fix:** correzioni di bug o malfunzionamenti;
- **feat:** introduzione o miglioramento di funzionalità;
- **refactor:** ristrutturazioni del codice senza modifica del comportamento esterno;
- **ci:** modifiche relative alla gestione della CI/CD;
- **test:** modifiche relative alla creazione o aggiornamento di test.

Ogni messaggio deve essere conciso e descrivere in modo chiaro il contenuto del commit.

2.2.2.4.3. Protezione dei branch

Come già avviene per il repository della documentazione, sono state attivate regole di **branch protection** su **main** e **develop**:
- È richiesto il superamento dei controlli automatici prima del merge;
- È obbligatoria almeno una review approvata.

2.2.2.4.4. Label dedicate

Nelle repository di sviluppo sono state introdotte nuove label per categorizzare rapidamente le issue:
- **enhancement** per miglioramenti o nuove feature;
- **bug** per segnalazioni di malfunzionamenti;
- **documentation** per documentazione tecnica;
- **test** per attività relative alla scrittura o revisione di test;
- **environment** per attività di configurazione ambienti o CI/CD.

2.2.2.4.5. Testing

La verifica del software sarà suddivisa in:
- **Testing automatico**, gestito tramite pipeline CI/CD, comprendente:
  - test di unità;
  - test di integrazione;
- **Testing manuale**, eseguito periodicamente per validare i comportamenti complessi non completamente automatizzabili.

2.2.2.4.6. Stile del codice

Lo stile dei sorgenti seguirà le convenzioni ufficiali delle principali guideline:
- Per il codice **Python**, si adotterà la nomenclatura indicata in [PEP8](`https://peps.python.org/pep-0008/`).
- Per il codice **TypeScript/JavaScript**, si seguiranno le convenzioni del [Google TypeScript Style Guide](`https://google.github.io/styleguide/tsguide.html#naming`).

La coerenza nello stile favorirà la leggibilità del codice e semplificherà i processi di revisione.

2.2.3. Strumenti usati

- **VS Code**: per la scrittura del codice;
- **Draw.io**: per elaborare i diagrammi UML* degli use case individuati durante la fase di analisi dei requisiti;

3. Processi di supporto

3.1. Documentazione

3.1.1. Scopo e descrizione

Lo scopo del processo di documentazione è quello di tracciare e quindi rendere immediatamente consultabile ogni attività e processo relativi al progetto. Inoltre, saranno riportate le decisioni prese e le norme scelte dal gruppo CodeHex16 che verranno rispettate da tutti i suoi membri al fine di procedere al meglio nello sviluppo del progetto.

3.1.2. Attività

Il processo di documentazione si suddivide in due attività principali:
- **Realizzazione**: attività in cui uno o più membri del gruppo redigono un determinato documento;
- **Verifica e approvazione**: attività che segue la realizzazione ed è necessaria per verificare che il documento in questione abbia rispettato i criteri di qualità e le norme scelte dal gruppo;
- **Aggiornamento**: attività necessaria per stabilire come agire nel momento in cui un documento debba subire degli aggiornamenti come modifiche, aggiunte o correzioni;

3.1.2.1. Realizzazione

Attività che si suddivide a sua volta nelle seguenti sottoattività:
- **Individuazione compito e creazione issue**: in questa fase viene identificato il compito da svolgere relativo ad un documento e viene creata una issue;
- **Assegnazione issue e creazione branch ausiliario**: la issue in questione viene assegnata ad uno dei membri del gruppo che avrà il compito di creare la branch secondaria su cui lavorare;
- **Redazione**: il membro del team può iniziare la stesura del documento;

3.1.2.2. Verifica e approvazione

Una volta terminata la stesura del documento si procede nel seguente modo:

- **Creazione pull request**: il membro che ha redatto il documento crea la pull request al fine di richiedere una verifica del lavoro svolto;
- **Approvazione**: viene eseguita la verifica e se il documento rispetta le norme e i criteri di qualità viene approvato e integrato con la documentazione principale;

3.1.2.3. Aggiornamento

Se un documento già redatto ha bisogno di essere modificato e/o corretto si entra nella fase di aggiornamento in cui vengono rieseguiti i passaggi delle attività di **Realizzazione** e **Verifica e approvazione**.

3.1.3. Struttura dei documenti

Ci sono diversi tipi di documenti e generalmente sono organizzati nelle seguenti sezioni:

3.1.3.1. Intestazione

La prima pagina è l'intestazione del documento ed è composta generalmente dalle seguenti informazioni:
- **Titolo**: Titolo del documento che è anche il nome del file;
- **E-mail**: E-mail del gruppo;
- **Logo**: Logo del gruppo;
- **Data**: La data in cui il gruppo si è incontrato e/o in cui il documento è stato redatto;
- **Versione**: La versione corrente del documento;
- **Sommario**: Una breve descrizione del contenuto del documento;
- **Ruolo**: I ruoli dei membri del gruppo in relazione al documento;

Per i verbali vengono aggiunte le seguenti informazioni:
- **Tipo**: Il verbale può essere di tipo interno o esterno;
- **Ora**: L'orario in cui è avvenuto l'incontro di gruppo;
- **Ordine del giorno**: L'elenco degli argomenti principali discussi durante l'incontro, che sostituisce il sommario;
- **Presenze**: Per ogni individuo presente durante l'incontro viene registrato il suo nome, cognome e il tempo per cui è stato presente durante l'incontro;

3.1.3.2. Registro delle modifiche

La seconda pagina riguarda il registro modifiche il cui contenuto è organizzato mediante una tabella in cui vengono riportate le seguente informazioni:

- **Versione**: Indica il numero della versione del documento, seguendo il formato definito nella sezione «Versionamento» 3.2.3;
- **Data**: Data della versione in cui è redatto il documento;
- **Autore**: Autore di quella versione del documento, cioè il membro del gruppo che ha apportato le modifiche;
- **Cambiamenti**: I cambiamenti principali di quella versione del documento;
- **Verificatore**: Membro del gruppo che ha verificato il documento per quella versione;

3.1.3.3. Indice

La terza pagina, e se necessario le seguenti, è riservata all'indice del documento che elenca le sezioni di cui è composto i documento.

3.1.3.4. Corpo del documento

Il corpo del documento è strutturato in capitoli, ciascuno dei quali può essere suddiviso in sottocapitoli, che a loro volta possono contenere ulteriori suddivisioni.

3.1.4. Documenti del progetto

Verranno prodotti i seguenti documenti:
- **Norme di Progetto***;
- **Piano di Progetto**;
- **Analisi dei Requisiti**;
- **Piano di Qualifica**;
- **Manuale Utente**;
- **Manuale Sviluppatore**;
- **Specifica tecnica***;
- **Glossario**;
- **Verbali esterni**;
- **Verbali interni**;

3.1.5. Elenchi puntati

Ogni voce di un elenco puntato finisce con «;».

3.1.6. Immagini use case

Per produrre i diagrammi uml degli use case il team ha usato il seguente sito : `draw.io`. Successivamente vengono inseriti nei documenti opportuni come immagini.

3.1.7. Formato delle date

Per le date viene utilizzato lo standard internazionale **ISO 8601** nella forma YYYY-MM-DD in cui:
- **YYYY**:Indica l'anno con 4 cifre;
- **MM**: Indica il mese con 2 cifre;
- **DD**: Indica il giorno con 2 cifre;

3.1.8. Composizione tipografica

Per la composizione tipografica dei documenti si è deciso di usare Typst* per i seguenti motivi:
- Semplicità degli strumenti utilizzati per la stesura;
- Sintassi semplice;
- Compilazione immediata;

Grazie a Typst si riesce facilmente a creare e mantenere un documento non lasciando il lavoro di controllo grafico al gruppo, infatti si possono scrivere i vari tipi di documenti partendo dai template che si possono trovare nella repository **`documentazione`**.

3.1.9. Strumenti

- **VS Code**: Editor di testo usato per scrivere i documenti;
- **Typst**: Linguaggio usato per la stesura dei documenti;
- **Github**: Servizio di hosting per il repository;

3.2. Gestione della configurazione

3.2.1. Scopo e descrizione

Il processo di gestione di configurazione identifica le norme adottate dal gruppo per garantire la tracciabilità della documentazione e del codice prodotto durante tutto l'arco di vita del progetto. Lo scopo principale è quello di organizzare la procedura di modifica della documentazione e del codice prodotto e di rendere immediatamente consultabili le varie modifiche apportate e i loro autori.

3.2.2. Attività

Le principali attività del processo di Gestione della configurazione sono:

- **Inizializzazione**: vengono definite le norme a cui ogni elemento prodotto dal gruppo durante l'intero arco del progetto deve sottostare e quindi rispettare;
- **Esecuzione**: viene svolto un determinato lavoro da uno o più membri del gruppo che producono un elemento (documento, codice o altro) utile al progetto;
- **Versionamento**: il lavoro svolto viene registrato così da risultare immediamente consultabile dagli altri membri del team, o membri esterni, così da sapere chi ha svolto quel lavoro e in che momento;
- **Verifica**: il lavoro svolto deve essere verificato da uno o più membri del team che svolgono il ruolo di verificatori con l'obiettivo di garantire che tale lavoro rispetti le norme predefinite;
- **Accettazione o Rifiuto**: se il lavoro svolto rispetta le norme predefinite ed è considerato corretto allora viene accettato e integrato con il lavoro principale, altrimenti viene rifiutato e dovranno essere eseguite delle modifiche;

3.2.3. Versionamento

In generale una versione ha una sintassi del tipo X.Y.Z in cui:
- **X**: E' il numero di versione principale che viene incrementato ogni qual volta il documento sia terminato e pronto per una revisione;
- **Y**: E' il numero di versione secondaria che viene incrementato ogni qual volta il documento sia stato modificato in modo significativo;
- **Z**: E' il numero di versione di correzione, incrementato ogni qual volta il documento subisca correzioni minori;

In particolare, per i verbali vengono considerati solo i numeri Y e Z per il fatto di essere documenti molto brevi, in cui:
- **Y**: E' il numero di versione principale che viene incrementato ogni qual volta il documento sia terminato o venga modificato con correzioni importanti;
- **Z**: E' il numero di versione secondaria che viene incrementato ogni qual volta il documento sia stato modificato con piccole correzioni;

3.2.4. Repository

3.2.4.1. Struttura della repository documentazione

Il contenuto pronto e convalidato del repository è presente nel branch* main in cui sono presenti tutti i PDF dei documenti prodotti. Nel branch main si possono trovare diverse cartelle che servono per organizzare e dividere i vari tipi di documenti e codice, in particolare:

- In **1 - candidatura** si trovano i documenti relativi alla candidatura per la gara d'appalto dei capitolati;
- In **2 - RTB** si trovano i documenti relativi alla fase di progetto **RTB (Requirement and Technology Baseline)**;
- In **3 - PB** si trovano i documenti relativi alla fase di progetto **PB (Product Baseline)**;
- In **diari-di-bordo** sono presenti tutti i diari di bordo prodotti;
- In **glossario** è presente il documento Glossario;
- In **template** sono presenti i template per i vari documenti.

3.2.5. Sincronizzazione

La sincronizzazione avviene tramite repository condivise su `github` in cui ogni attività da svolgere è tracciata da una issue con il/i membro/i assegnato/i a tale issue così da sapere sempre chi la segue, o l'ha seguita.

3.2.5.1. Branch

Per gestire al meglio le varie issue e la documentazione si è deciso di creare dei branch ogni volta che un membro svolge una o un insieme di attività correlate tra di loro. Quindi in generale la sintassi del nome di un branch è la seguente: **[NomeDocumento/Attività]-[NomeMembro]-[Sprint]**. Una volta finito il lavoro da parte di tutti i membri che operano su quel documento, questo viene verificato (tramite pull request*) e viene eseguito il merge sul branch main, mentre i branch ausiliari vengono eliminati appena si ha la sicurezza che questi non sono più necessari.
Altri branch degni di nota sono:
- **diario-di-bordo**: Utilizzato per inserire i diari di bordo nel repository;

3.2.5.2. Pull request

Per quanto riguarda le pull request si è deciso che per ogni documento redatto viene richiesta la verifica tramite pull request. A questo punto i verificatori a cui è stata assegnata la issue di verifica di quel documento, tramite questa pull request, eseguono la verifica e se è tutto corretto viene fatto il merge delle modifiche apportate nel main, altrimenti il verificatore può correggere direttamente il documento, oppure scrivere un commento con delle indicazioni per le correzioni da svolgere.

A questo punto i verificatori incaricati del sprint corrente (vengono create delle issue per la verifica assegnandole a chi di dovere), tramite questa pull request:
1. Esegue il controllo locale

```
# Clonare il repository
git checkout <pr-branch-name>
```

2. Eseguono la verifica e aggiungere proprio nome nel Registro dell Versioni
3. Se è tutto corretto viene effettuato il commit

```
git add .
git commit -m "Verifica <pr-branch-name>...."
git push origin <pr-branch-name>
```

4. Dopo di che viene fatto il merge delle modifiche apportate nel main sulla pagina Pull Request del Github
5. Altrimenti il verificatore può correggere direttamente il documento, oppure scrivere un commento con delle indicazioni per le correzioni da svolgere.

In generale, le pull request vengono effettuate quando vi è una modifica interna al repository.

3.2.6. Strumenti usati

- **Git**: Software usato per il controllo della versione dei documenti e del codice;
- **Github**: Servizio di hosting per progetti software usato dal gruppo per coordinarsi sulle operazioni di versionamento e usato come **Issue Tracking System**;

3.3. Gestione qualità

3.3.1. Scopo e descrizione

Il processo di gestione della qualità ha come obiettivo quello di garantire che il software, la documentazione e tutto ciò che il team produce sia conforme ai requisiti di qualità specificati e richiesti. Processi utili a garantire un certo grado di qualità sono sicuramente i processi di verifica e validazione. Gli obiettivi e gli standard di qualità richiesti e che devono essere soddisfatti sono indicati nel documento **Piano di Qualifica**.

3.3.2. Attività

Seguendo lo standard ISO/IEC 12207:1995, le attività previste per questo processo sono:

- **Implementazione**: in cui vengono realizzati gli strumenti necessari, per esempio i test, per poter attuare correttamente l'attività successiva di verifica, rivelando le varie problematiche e i metodi per mitigarle;

- **Verifica**: in cui deve essere controllata l'efficacia dei processi, dei requisiti rispetto alla consistenza ed esaustività degli stessi, della documentazione prodotta, della progettazione rispetto ai requisiti individuati durante la fase di analisi e riportati nel documento **Analisi dei Requisiti**, del codice e dell'integrazione delle varie componenti del sistema tra loro.

Quindi è l'attività che mette in pratica ciò che è stato individuato e realizzato durante l'attività di **Implementazione**;

3.3.3. Verifica

3.3.3.1. Scopo e descrizione

La verifica è un'attività affidata ai verificatori e inizia quando la fase iniziale di progettazione viene avviata. Questo processo ha come obiettivo quello di garantire un certo grado di qualità di tutto quello che viene prodotto dal gruppo (documentazione, codice sorgente, test, ecc..) e di conformità rispetto alle aspettative. Quindi tramite tecniche di analisi e test tale processo mira a stabilire se ciò che viene prodotto dal team soddisfa i requisiti richiesti. Il documento che rispecchia questo processo è il **Piano di Qualifica** e definisce gli obiettivi da raggiungere, i criteri di accettazione e i metodi che verranno usati per eseguire la verifica in modo completo ed efficiente.

3.3.3.2. Analisi statica

L'analisi statica è un tipo di verifica che non richiede l'esecuzione del prodotto e ha come obiettivo la revisione critica del codice e della documentazione al fine di garantire conformità ai vincoli, assenza di difetti e presenza delle funzionalità e proprietà richieste. Questo tipo di analisi adotta, tipicamente, due metodi di lettura: l'**inspection** e il **walkthrough**. L'inspection, preferibile al walkthrough per velocità ed efficienza, consente di individuare tempestivamente potenziali difetti, errori e problemi. Il walkthrough mette in collaborazione il verificatore con l'autore del prodotto preso in analisi e ne prevede una lettura a pettine.

3.3.3.3. Analisi dinamica

L'analisi dinamica è un tipo di verifica che richiede l'esecuzione del sistema e delle sue componenti così da individuare difetti, problemi ed errori nel funzionamento al fine di garantire un certo grado di qualità nel prodotto finale. Il gruppo, per garantire tutto ciò, utilizzerà un insieme di test ripetibili e automatizzati, anche se sarà necessario l'uso di test manuali. Tali test si suddividono nei seguenti tipi:

3.3.3.3.1. Test

Test di unità
I test di unità sono test effettuati su singole componenti autonome del sistema e tali test possono essere:
- **Test funzionali** che verificano che l'output prodotto sia uguale a quello atteso;
- **Test strutturali** che verificano tutti i possibili cammini del codice;

Test di sistema
I test di sistema sono impiegati per verificare il corretto funzionamento del sistema e, in particolare, che tutti i requisiti richiesti siano soddisfatti.

Test di integrazione
I test d'integrazione verificano la corretta integrazione tra le varie componenti del sistema che sono già state testate singolarmente tramite i test di unità. E' possibile seguire due approcci per i test d'integrazione:
- **Bottom up**: si testano per prime le componenti che hanno meno dipendenze e maggior valore interno, cioè quelle più nascoste all'utente;
- **Top down**: si testano per prime le componenti che hanno il maggior numero di dipendenze e maggiormente visibili da parte dell'utente così da avere disponibilità immediata di tali componenti;

Test di regressione
I test di regressione vengono impiegati per assicurare che la correzione o la modifica delle componenti non causi problemi al livello di sistema. Tali test sono necessari per garantire che le modifiche non compromettano le funzionalità già testate e funzionanti, evitando quindi la comparsa di regressioni nel sistema.

Test di accettazione
I test di accettazione devono essere eseguiti insieme al committente al fine di verificare che il prodotto finale rispetti tutti i requisiti richiesti.

3.3.4. Validazione

La validazione è la verifica ultima per garantire che il prodotto sia in linea con le aspettative e che rispetti i requisiti richiesti e per questo è una fase molto importante nello sviluppo del progetto. Questo processo segue il processo di verifica e si sofferma su alcuni aspetti quali:

- Il prodotto finale deve funzionare correttamente ed essere conforme con la logica di progettazione;
- Il prodotto deve soddisfare completamente i requisiti specificati;
- Il prodotto deve essere intuitivo e di facile comprensione e utilizzo, cioè deve essere usabile;
- Il prodotto deve essere efficace nel soddisfare le necessità del cliente;

3.3.5. Strumenti usati

- **Github Actions**;
- **Github Pull Request**;

4. Processi organizzativi

4.1. Gestione dei processi

4.1.1. Scopo e descrizione

Il processo di gestione ha lo scopo di identificare le attività e i compiti che ogni membro del gruppo dovrà eseguire per proseguire nel progetto.

4.1.2. Attività

Questo processo prevede di seguire le seguenti attività:
- **Individuazione**: in cui vengono individuati il compito da svolgere e l'impegno necessario per portarlo a termine;
- **Pianificazione**: una volta individuato il compito da svolgere bisogna capire quanto costa in termini economici e temporali per svolgerlo e a chi affidarlo;
- **Esecuzione**: una volta compreso il compito e averlo affidato ad uno dei membri del gruppo questo deve svolgerlo e portarlo a termine;
- **Verifica**: quando il compito è concluso questo deve essere verificato da uno o più membri del gruppo (verificatori) al fine di garantire che il lavoro svolto soddisfi determinati vincoli di qualità;
- **Chiusura**: se il lavoro svolto soddisfa tutti i vincoli di qualità fissati allora questo può essere integrato con il lavoro principale, quindi il lavoro svolto viene approvato tramite una pull request apposita;

4.1.3. Ruoli

I ruoli svolti, a rotazione, dai membri del gruppo sono:
- **Responsabile**: Ha il compito primario di coordinare i membri del gruppo, inoltre deve:
  - Determinare le attività da svolgere, assegnarle e verificarne l'avanzamento;
  - Gestire i rapporti tra i membri del gruppo e i soggetti esterni;
  - Redigere i verbali sia interni che esterni;
  - Redigere il documento **Piano di Progetto**;

  Il responsabile è una figura che sarà presente durante tutto l'arco del progetto.
- **Amministratore**: Ha il compito primario di controllare e gestire l'ambiente di lavoro, inoltre deve:

- Stabilire gli strumenti necessari da usare durante il progetto;
- Gestire i processi e risolverne gli eventuali problemi;
- Redigere il documento **Norme di Progetto**;

L'amministratore è una figura che sarà presente durante tutto l'arco del progetto.

- **Progettista**: Ha lo scopo principale di determinare le scelte realizzative del progetto, in particolare deve:
  - Trovare l'architettura adeguata per gestire il progetto;
  - Redigere il documento **Specifica Tecnica**;

  Il progettista è una figura che sarà, principalmente, presente durante la parte di sviluppo del progetto.
- **Analista**: Ha il compito principale di trovare i requisiti che il progetto dovrà soddisfare e riportarli nel documento **Analisi dei Requisiti**, quindi deve:
  - Studiare i bisogni dei committenti;
  - Studiare i requisiti definendone la complessità;
  - Redigere il documento **Analisi dei Requisiti**;

  L'analista è una figura che sarà presente principalmente durante la prima parte del progetto in cui verrà analizzato e compreso appieno il capitolato. Solo in casi straordinari, cioè se il gruppo dovesse cambiare i requisiti del progetto, allora l'Analista dovrà essere interpellato per apportare delle modifiche ai requisiti in questione.
- **Programmatore**: Ha il compito primario di svolgere l'attività di codifica e sviluppare l'architettura individuata dal Progettista, in particolare deve:
  - Scrivere codice mantenibile che rispetti le **Norme di Progetto**;
  - Creare test per la verifica e validazione del codice;
  - Redige il documento **Manuale Utente**;

  Il programmatore è una figura che sarà presente durante la parte di sviluppo del progetto.
- **Verificatore**: Ha il compito principale di controllare e validare la documentazione e il codice prodotto, in particolare deve:
  - Controllare ogni documento a lui assegnato, verificarlo e in caso correggerlo o notificare il redattore, specificare cosa non è corretto e richiederne la correzione;
  - Controllare che tutto ciò che viene prodotto rispetti le **Norme di Progetto**;
  - Redige il documento **Piano di Qualifica**;

  Il Verificatore è una figura che sarà presente durante tutto il progetto.

Ogni membro, in un dato momento, può svolgere un solo ruolo alla volta, ma durante lo sprint può assumere più ruoli.

4.1.4. Ticketing (Issue Tracking System)

Il gruppo sfrutta l'ITS* offerto da Github per gestire le attività da svolgere, cioè le **Issue**. Creare le issue è molto semplice e veloce; quando viene individuata un'attività specifica da svolgere viene creata una issue e viene assegnata, in modo coerente, ad un membro del gruppo. Quando viene creata una issue, questa sarà composta da:

- **Titolo**: Il nome della issue che identifica l'attività da svolgere;
- **Assegnatario/i**: Il/I membro/i del gruppo a cui è affidata la issue;
- **Etichetta/e**: Identifica il tipo di issue, come ad esempio i documenti su cui lavorare o la macro categoria delle attività da svolgere;
- **Stato**: Avanzamento della issue, nello specifico:
  - **Todo**: La issue è stata creata ma non ancora svolta;
  - **In corso**: La issue è in fase di svolgimento;
  - **Completato**: La issue è stata completata e chiusa;
- **Priorità**: Identifica l'urgenza con cui svolgere la issue;
- **Peso**: Stima del carico di lavoro relativo alla issue;
- **Sprint**: Lo sprint in cui questa issue deve, idealmente, essere svolta;
- **Ruolo**: Identifica il ruolo associato allo svolgimento di tale issue;
- **Ore**: Il numero di ore impiegate per svolgere la issue;
- **Repository**: Repository a cui è associata la issue;
- **Milestone**: Identifica la milestone a cui è assegnata la issue;

Più in particolare quando viene individuato un compito da svolgere vengono eseguiti i seguenti passi:

1. Viene individuato il compito da svolgere e viene creata la issue relativa;
2. La issue viene assegnata ad uno o più membri del gruppo e vengono inserite le informazioni della issue scritte appena sopra.
3. La issue viene completata e chiusa, oppure se è richiesta la verifica del lavoro svolto allora il verificatore, che ha una issue di verifica parallela, controlla il lavoro svolto:
   a) se corretto vengono confermate le modifiche e la issue principale e quella del verificatore vengono chiuse;
   b) altrimenti le issue rimangono aperte e il verificatore suggerisce dei cambiamenti e/o correzioni a carico dell'assegnatario, una volta apportate tali modifiche si torna al punto 3;

4.1.5. Strumenti usati

- **Github**: per gestire tutta la documentazione e il codice per il progetto in un repository;;

- **GitHub Issue** per assegnare i compiti ad ogni membro avendo un rendiconto preciso dei ruoli e delle ore svolte per ogni sprint;
- **GitHub Project** per visualizzare in modo più ordinato le issue e i loro dettagli;

4.2. Coordinamento

4.2.1. Attività

Le principali attività del processo di coordinamento sono:
- **Pianificazione generale**: decisione del giorno e dell'orario in cui ritrovarsi (di solito viene deciso durante la riunione precedente, o tramite un documento apposito in **Google Fogli**);
- **Pianificazione specifica**: decisione degli argomenti principali da discutere durante la riunione;
- **Riunione**: momento in cui il gruppo si ritrova per controllare il lavoro svolto, risolvere eventuali dubbi e definire le nuove attività e quindi i prossimi obiettivi;
- **Redazione verbale**: Momento successivo alla riunione in cui viene redatto il verbale relativo alla riunione che verrà successivamente verificato e integrato con la documentazione principale;

4.2.2. Comunicazioni e Riunioni

Le comunicazioni principali che avvengono durante lo svolgimento del progetto sono di due tipi:
- **Comunicazioni interne**: Il gruppo utilizza **Telegram** e **Discord** per le comunicazioni principali interne, in particolare Telegram viene usato per messaggi brevi, veloci e informali, mentre Discord viene usato per discussioni e riunioni a distanza.
  Inoltre, in caso di problemi su Telegram, il gruppo può spostarsi su Discord per comunicare anche in modo non formale;
- **Comunicazioni esterne**: Per le comunicazioni esterne vengono usate la mail di gruppo **unipd.codehex16@gmail.com** e **Zoom** per chiarire dubbi e porre domande;

Anche le riunioni svolte durante l'arco del progetto sono di due tipi:
- **Riunioni interne**: I membri del gruppo si riuniscono su **Discord** dove si fa il punto della situazione in quel momento e se necessario si introducono nuove attività da svolgere e/o si procede con quelle già indicate;
- **Riunioni esterne**: I membri del gruppo si riuniscono insieme al proponente, questa riunione di solito è richiesta dal gruppo per trattare problemi/dubbi di una certa

importanza al fine di avere una migliore comprensione delle attività da svolgere e procedere con il progetto;

Sarà compito del **Responsabile** riassumere in un **verbale interno** o un **verbale esterno** quello che si è discusso durante la riunione in questione;

4.2.3. Verbali

I verbali redatti sono di due tipi:
- **Verbali interni**: Questo tipo di verbale è la trascrizione dei punti salienti di una riunione interna e, generalmente, l'obiettivo principale è quello di discutere delle attività svolte e delle attività da svolgere, stabilendo quindi nuove issue, se necessario;
- **Verbali esterni**: Questo tipo di verbale è la trascrizione dei punti più importanti riscontrati durante una riunione esterna, il cui obiettivo principale è quello di risolvere dubbi/problemi riscontrati durante l'avanzamento del progetto;

Nel caso in cui il Responsabile non fosse presente durante le riunioni, il verbale corrispondente verrà redatto dall'**Amministratore**; se neanche l'amministratore fosse presente allora sarà uno dei membri presenti alla riunione ad avere l'incarico di redigere il verbale.

4.2.4. Strumenti usati

- **Zoom**: per gli incontri esterni;
- **Discord**: usato per le riunioni interne;
- **Telegram**: usato per le comunicazioni interne;
- **Google Mail**: per le comunicazioni esterne;
- **Notion**: per organizzare appunti e documenti in modo non ufficiale;
- **Google Fogli**: per organizzare gli orari per le riunioni interne;

4.3. Miglioramento

4.3.1. Scopo e descrizione

Il miglioramento è un processo sempre attivo che durerà per tutto il progetto il cui obiettivo è quello di controllare e migliorare tutto quello che viene prodotto mantenendo un elevato grado di qualità. In particolare si cerca e si cercherà di ruotare ruoli, identificare attività idonee ai membri del gruppo così da risolvere il maggior numero di problemi che si potrebbero venire a creare e/o colmare tempestivamente eventuali lacune mantenendo il lavoro del team elastico e flessibile.

4.3.2. Attività

Le attività principali del processo di miglioramento sono:
- **Inizializzazione**: stabilire le norme che ogni membro del team dovrà rispettare durante lo svolgimento del progetto;
- **Controllo**: stabilendo le norme ora, basandosi su queste si possono stabilire alcune metriche da valutare per controllare e verificare l'efficacia e l'efficienza di ogni processo svolto;
- **Attivazione**: se risulta che un processo non rispetta le norme e/o non è efficiente e/o efficace allora tale processo deve essere migliorato e quindi è compito del team stesso trovare delle soluzioni per migliorare questo processo;

4.3.3. Strumenti usati

- **Github**: usato come **Issue Tracking System** per gestire e monitorare il lavoro di ogni membro del gruppo e dell'andamento generale del progetto;
- **Notion**: per organizzare appunti e documenti in modo non ufficiale;

4.4. Formazione

4.4.1. Scopo e descrizione

Lo scopo principale del processo di formazione è quello di tenere aggiornati e preparati i membri del gruppo così da reagire prontamente ad eventuali modifiche o problemi che possono sorgere durante lo svolgimento del progetto.

4.4.2. Attività

Le principali attività del processo di formazione sono:
- **Inizializzazione**: vengono definite le procedure per formare i membri del gruppo;
- **Sviluppo materiale**: viene sviluppato il materiale necessario alla formazione e aggiornamento per tutti i membri del gruppo;
- **Attivazione**: quando necessario i membri del gruppo attivano i meccanismi di formazione. In particolare vengono attivati, durante gli sprint, dei momenti e delle issue relative allo studio delle tecnologie usate durante il progetto. Inoltre, se necessario, i membri del gruppo possono incontrarsi per discutere problemi specifici già affrontati e risolti da altri, favorendo così la condivisione delle conoscenze tra colleghi. In ogni caso tutti i membri del team possono accedere alle repository riguardanti il codice

e la documentazione per ottenere informazioni in modo veloce e risolvere eventuali dubbi;

4.4.3. Strumenti usati

- **Discord**: per gli incontri formali;
- **Github**: per gestire tutta la documentazione e il codice per il progetto in un repository;
- **Notion**: per organizzare appunti e documenti in modo non ufficiale;
- **Telegram**: per comunicare in modo veloce con gli altri membri del team;

5. Metriche e standard per la qualità

Per migliorare e avere uno standard di qualità da cui attingere il gruppo ha deciso di utilizzare degli standard riconosciuti a livello internazionale. Tra quelli disponibili è stato scelto lo standard ISO/IEC 12207:1995 per quanto riguarda la qualità dei processi principali, organizzativi e di supporto. Mentre per gli standard di qualità del software il gruppo adotterà lo standard ISO/IEC 25010:2023. Le principali caratteristiche prese in considerazione sono:

- **Funzionalità**;
- **Performance**;
- **Affidabilità**;
- **Usabilità**;
- **Portabilità**;
- **Manutenibilità**;

5.1. Funzionalità

La funzionalità riguarda e misura il grado di soddisfacibilità delle esigenze del proponente rispetto al prodotto finale. In particolare vengono misurate:

- **Completezza**: il software prodotto deve soddisfare tutti i requisiti e le funzionalità previste;
- **Correttezza**: il funzionamento del software deve rispettare le specifiche dichiarate;
- **Adeguatezza**: il software prodotto deve essere idoneo allo scopo per cui è stato pensato e al contesto in cui viene usato;
- **Conformità**: il software prodotto deve rispettare le norme predefinite e il grado di qualità previsto;

5.2. Performance

La performance misura l'efficienza di un prodotto e la capacità del prodotto stesso di gestire le risorse in modo direttamente proporzionale alle prestazioni che vengono fornite. In particolare vengono misurate:

- **Risorse**: l'uso delle risorse deve essere efficiente;
- **Tempo**: il software prodotto deve dare una risposta in tempi consoni alle richieste che gli vengono fatte;
- **Conformità**: il prodotto deve rispettare dei vincoli di qualità;
- **Capacità**: il prodotto deve saper gestire i carichi di lavoro attesi senza compromettere le prestazioni previste;

5.3. Affidabilità

L'affidabilità è il parametro per misurare se un prodotto reagisce ai problemi senza compromettere le prestazioni. In particolare vengono misurate:

- **Tolleranza ai guasti e agli errori**: il prodotto deve saper gestire gli errori e rispondere a malfunzionamenti o guasti senza che si causino interruzioni gravi nel prodotto e allo stesso tempo mantenere un certo grado di prestazioni;
- **Maturità**: il prodotto deve garantire affidabilità e stabilità cercando di evitare che si vengano a generare errori o malfunzionamenti;
- **Recuperabilità**: a seguito di un qualsiasi tipo di errore, guasto o malfunzionamento il prodotto deve essere in grado di tornare alle sue prestazioni standard rispristinando le sue funzionalità;
- **Disponibilità**: il prodotto deve essere accessibile e correttamente funzionante, quindi operativo ogni qual volta sia necessario;
- **Aderenza**: il prodotto deve rispettare gli standard di qualità prefissati;

5.4. Sicurezza

La sicurezza di un prodotto misura il suo grado di protezione da minacce e vulnerabilità, garantendo che i dati e le funzionalità rimangano integri, disponibili e riservati.

- **Riservatezza**: il software deve garantire la protezione dei dati sensibili;
- **Integrità**: il software deve garantire che i dati siano completi, accurati e sicuri;
- **Autenticazione**: il software deve controllare e verificare le credenziali degli utenti e limitare l'accesso ai soli utenti autorizzati;
- **Autenticità**: il prodotto deve permettere di verificare la provenienza dei dati;

5.5. Usabilità

L'usabilità ci permette di calcolare e comprendere quando e in quanto tempo l'utente finale riesca ad apprendere le modalità di utilizzo di un prodotto. In particolare vengono misurate:

- **Apprendibilità**: la semplicità con cui l'utente riesce ad apprendere le funzionalità del prodotto;
- **Comprensibilità**: la facilità con cui l'utente comprende il funzionamento del prodotto e riesce ad utilizzarlo in modo appropriato;
- **Riconoscibilità**: il prodotto deve fornire un'interfaccia utente intuitiva e di facile comprensione;
- **Estetica**: l'interfaccia del prodotto deve essere gradevole;

- **Operabilità**: quanto è semplice per l'utente usare il prodotto in maniera corretta;

5.6. Portabilità

La portabilità è la capacità di un prodotto di essere facilmente spostato da un ambiente di esecuzione ad un altro senza insorgere in problemi. In particolare vengono misurate:
- **Adattabilità**: il software deve essere capace di funzionare in ambienti di esecuzione diversi in modo corretto;
- **Installabilità**: il software deve poter essere installato e configurato facilmente e rapidamente;
- **Sostituibilità**: il prodotto deve poter funzionare correttamente nel momento in cui venga aggiornato e quindi sostituito da nuove versioni;

5.7. Manutenibilità

La manutenibilità di un prodotto misura la facilità con cui può essere modificato, corretto e migliorato nel tempo. In particolare vengono misurate:
- **Analizzabilità**: la facilità con cui il codice può essere controllato al fine di risolvere eventuali errori e problemi;
- **Modificabilità**: il software deve poter essere modificato e migliorato in maniera semplice;
- **Testabilità**: la semplicità con cui il prodotto può essere testato;
- **Riutilizzabilità**: le varie parti del software devono poter essere riutilizzate in progetti o ambienti differenti;
- **Stabilità**: capacità del prodotto di continuare a funzionare senza gravi problemi a seguito di modifiche sbagliate;

6. Metriche di qualità

6.1. Nomenclatura delle metriche

Per identificare le metriche relative ai processi e quelle relative ai prodotti vengono usate, come prefisso, le seguenti sigle:
- **MPC***: sigla per le metriche per la qualità dei processi;
- **MPD**: sigla per le metriche per la qualità del prodotto;

Quindi una metrica avrà come sigla : **MPC/MPD-AcronimoMetrica**.

6.2. Metriche per i processi

6.2.1. Processi primari

6.2.1.1. Fornitura

6.2.1.1.1. Budget At Completion(MPC-BAC)

- **Codice**: MPC-BAC;
- **Descrizione**: costo totale del progetto preventivato per il suo completamento;

6.2.1.1.2. Estimated Cost(MPC-EC)

- **Descrizione**: costo stimato calcolando le ore necessarie per lo sviluppo del progetto;

6.2.1.1.3. Completion Cost(MPC-CC)

- **Descrizione**: costo finale raggiunto alla fine del progetto. Idealmente non deve superare quello stimato durante le fasi iniziali;
- **Valore ottimo**: ≤ 100% EC;
- **Valore accettabile**: ≤ 105% EC;

6.2.1.1.4. Actual Cost(MPC-AC)

- **Descrizione**: budget utilizzato fino a quel determinato momento. Indicatore utile per monitorare l'andamento del progetto e valutare se i costi rispettano le aspettative;
- **Valore ottimo**: ≤ EAC;
- **Valore accettabile**: ≥ 0%;

6.2.1.1.5. Earned Value(MPC-EV)

- **Descrizione**: valore ottenuto fino a quel dato momento, si basa sui progressi del completamento delle attività. In particolare viene quantificato il valore del lavoro effettivamente completato rispetto al budget complessivo. Questo indicatore permette di monitorare e valutare l'andamento del progetto, offrendo una misura concreta dello stato di avanzamento rispetto a quanto pianificato;
- **Come calcolarlo**: $\text{Earned Value} = \frac{\text{Budget at Completion}}{\%\text{ lavoro completato}}$;
- **Valore ottimo**: ≤ EAC;
- **Valore accettabile**: ≥ 0;

6.2.1.1.6. Planned Value(MPC-PV)

- **Descrizione**: rappresenta il valore del lavoro che dovrebbe essere completato. Si basa sulla programmazione delle attività del progetto e riflette il valore del lavoro che si intende portare a termine. Questo indicatore fornisce una base di riferimento per confrontare il progresso reale del progetto con le aspettative;
- **Come calcolarlo**: $\text{Planned Value} = \frac{\text{Budget at Completion}}{\%\text{ lavoro da completare}}$;
- **Valore ottimo**: ≤ BAC;
- **Valore accettabile**: ≥ 0;

6.2.1.1.7. Cost Performance Index(MPC-CPI)

- **Descrizione**: indicatore che misura l'efficienza del costo del lavoro realizzato rispetto al costo pianificato. Il Cost Performance Index rappresenta il rapporto tra il valore del lavoro effettivamente completate e il budget utilizzato per portarlo a termine;
- **Come calcolarlo**: $\text{Cost Performance Index} = \frac{\text{Earned Value}}{\text{Actual Cost}}$;
- **Valore ottimo**: ≥ 1;
- **Valore accettabile**: ≥ 0.8;

6.2.1.1.8. Estimate At Completion(MPC-EAC)

- **Descrizione**: stima del costo totale del progetto al momento del suo completamento, considerando i costi sostenuti fino ad ora e una stima aggiornata dei costi rimanenti;
- **Come calcolarlo**: $\text{Estimate At Completion} = \frac{\text{Budget at Completion}}{\text{Cost Performance Index}}$;
- **Valore ottimo**: = BAC;
- **Valore accettabile**: ± 5% BAC;

6.2.1.1.9. Estimate To Complete(MPC-ETC)

- **Descrizione**: stima del costo necessario per completare il progetto.
- **Come calcolarlo**: Estimate To Complete = Estimate At Completion - Actual Cost;
- **Valore ottimo**: ≤ EAC;
- **Valore accettabile**: ≥ 0%;

6.2.1.1.10. Schedule Performance Index(MPC-SPI)

- **Descrizione**: è una metrica che misura quanto il progetto sta procedendo rispetto alla sua pianificazione iniziale tramite il rapporto tra il costo preventivato del lavoro completato(EV) e il costo preventivato del lavoro ancora da svolgere(PV);
- **Come calcolarlo**: $\text{Schedule Performance Index} = \frac{\text{Earned Value}}{\text{Planned Value}}$;
- **Valore ottimo**: ≥ 1;
- **Valore accettabile**: ≥ 0.8;

6.2.1.1.11. Schedule Variance(MPC-SV)

- **Descrizione**: varianza rispetto a quanto previsto inteso come anticipo o ritardo sui tempi delle attività svolte e da svolgere. Rappresenta la differenza tra il valore del lavoro completato e il valore del lavoro pianificato. In pratica, misura se un progetto è in anticipo, in ritardo o in linea con la pianificazione effettuata inizialmente;
- **Come calcolarlo**: Schedule Variance = Earned Value - Planned Value;
- **Valore ottimo**: ≥ 0;
- **Valore accettabile**: ≥ −10%;

6.2.1.1.12. Cost Variance(MPC-CV)

- **Descrizione**: valore che misura la differenza tra il budget disponibile e quello usato effettivamente fino a quel momento. Rappresenta la differenza tra il valore del lavoro completato e il budget utilizzato per completarlo. È un indicatore fondamentale per valutare la performance finanziaria di un progetto, rivelando se si sta spendendo più o meno di quanto previsto dal budget inizialmente preventivato.
- **Come calcolarlo**: Cost Variance = Earned Value - Actual Cost;
- **Valore ottimo**: ≥ 0;
- **Valore accettabile**: ≥ −5%;

6.2.1.2. Sviluppo

6.2.1.2.1. Requirement Stability Index(MPC-RSI)

- **Descrizione**: indice di stabilità dei requisiti. Indica la percentuale di requisiti che sono stati modificati rispetto al totale dei requisiti. Un valore alto indica che i requisiti sono stabili e non soggetti a modifiche frequenti;
- **Come calcolarlo**: $\text{Requirement Stability Index} = \left(\frac{\text{TNOR + NCR + NAR + NDR}}{\text{TNOR}}\right) * 100$ dove:
  - **TNOR**: Total Number of Original Requirements = numero iniziale di requisiti;
  - **NCR**: Number of Changed Requirements = numero di requisiti modificati ;
  - **NAR**: Number of Added Requirements = numero di requisiti aggiunti;
  - **NDR**: Number of Deleted Requirements = numero di requisiti cancellati;
- **Valore ottimo**: 100%;
- **Valore accettabile**: ≥ 80%;

6.2.1.2.2. Technical Debt Ratio(MPC-TDR)

- **Descrizione**: rapporto tra il tempo necessario per risolvere i problemi tecnici e il tempo necessario per sviluppare nuove funzionali. Quindi calcola quanto costa correggere e mantenere il codice, rispetto a quanto è costato inizialmente svilupparlo;
- **Valore ottimo**: ≤ 5%;
- **Valore accettabile**: ≤ 15%;

6.2.2. Processi di supporto

6.2.2.1. Documentazione

6.2.2.1.1. Indice di Gulpease(MPC-IG)

- **Descrizione**: Indica la complessità nella lettura di una frase o documento. Considera come variabili il numero di parole, di frasi e di lettere;
- **Come calcolarlo**: $\text{Indice di Gulpease} = 89 + \frac{(300*\text{numero di frasi}) - (10*\text{numero di lettere})}{\text{numero di parole}}$;
- **Valore ottimo**: ≥ 60;
- **Valore accettabile**: ≥ 40;

6.2.2.1.2. Correttezza ortografica(MPC-CO)

- **Descrizione**: Indica il numero di errori ortografici presenti nella documentazione;
- **Valore ottimo**: 0;

- **Valore accettabile**: 3;

6.2.2.2. Gestione qualità

6.2.2.2.1. Satisfaction of Quality Metrics(MPC-SQM)

- **Descrizione**: misura della quantità di metriche soddisfatte. Più in particolare viene misurato il grado di soddisfazione dell'utente finale rispetto alla qualità di un prodotto. Ciò aiuta a comprendere se le aspettative del cliente sono state soddisfatte o meno, e consentono di identificare eventuali migliorie;
- **Come calcolarlo**: $\text{Satisfaction of Quality Metrics} = \frac{\text{Numero totale di metriche soddisfatte}}{\text{Numero totale di metriche}}$;
- **Valore ottimo**: 100%;
- **Valore accettabile**: ≥ 85%;

6.2.2.3. Verifica

6.2.2.3.1. Code Coverage(MPC-CCO)

- **Descrizione**: Quantità di codice eseguito durante i test. Viene utilizzato per valutare la qualità dei test e garantire che il codice sia stato adeguatamente testato. Un alto livello indica che il codice è stato eseguito in molti contesti e scenari diversi con diverse parti di codice. Quindi indica quanto codice è stato sottoposto ai test;
- **Come calcolarlo**: $\text{Code Coverage} = \frac{\text{Linee di Codice Eseguite}}{\text{Linee di Codice Totali}} * 100$;
- **Valore ottimo**: 100%;
- **Valore accettabile**: ≥ 90%;

6.2.2.3.2. Test Superati in Percentuale(MPC-TSP)

- **Descrizione**: Indica la proporzione di test automatizzati o manuali che sono stati eseguiti con successo rispetto al totale dei test previsti. Viene espressa come una percentuale e serve a misurare quanto dell'applicazione in fase di sviluppo è stato verificato con successo tramite i test. Una percentuale alta di test superati indica che il sistema è stabile e che la maggior parte delle funzionalità funzionano come previsto. Quindi indica quanti test sono stati superati;
- **Come calcolarlo**: $\text{Test Superati in Percentuale} = \frac{\text{Numero di Test Superati}}{\text{Numero Totale di Test}} * 100$;
- **Valore ottimo**: 100%;
- **Valore accettabile**: 100%;

6.2.3. Processi organizzativi

6.2.3.1. Gestione dei processi

6.2.3.1.1. Time Efficiency(MPC-TE)

- **Descrizione**: misura il rapporto tra ore utilizzate e ore produttive;
- **Come calcolarlo**: $\text{Time Efficiency} = \frac{\text{Ore Produttive}}{\text{Ore Totali}} * 100$;
- **Valore ottimo**: 1;
- **Valore accettabile**: 3;

6.3. Metriche per il prodotto

6.3.1. Funzionalità

6.3.1.1. Copertura Requisiti Obbligatori(MPD-RO)

- **Descrizione**: indica la percentuale di requisiti obbligatori coperti dal prodotto. Un valore del 100% indica che tutti i requisiti obbligatori sono stati implementati;
- **Come calcolarlo**: $\text{Copertura Requisiti Obbligatori} = \frac{\text{Numero Requisiti Obbligatori implementati}}{\text{Numero Requisiti Obbligatori totali}}$;
- **Valore ottimo**: 100%;
- **Valore accettabile**: 100%;

6.3.1.2. Copertura Requisiti Opzionali(MPD-OP)

- **Descrizione**: indica la percentuale di requisiti opzionali coperti dal prodotto. Un valore del 100% indica che tutti i requisiti opzionali sono stati implementati;
- **Come calcolarlo**: $\text{Copertura Requisiti Opzionali} = \frac{\text{Numero Requisiti Opzionali implementati}}{\text{Numero Requisiti Opzionali totali}}$;
- **Valore ottimo**: 100%;
- **Valore accettabile**: $\geq$ 50%;

6.3.2. Affidabilità

6.3.2.1. Code Coverage(MPD-CC)

- **Descrizione**: indica la percentuale di codice coperto dai test. Un valore alto indica che il codice è stato testato in modo approfondito e che è meno probabile che contenga errori;

- **Come calcolarlo**: $\text{Code Coverage} = \frac{\text{righe di codice testate}}{\text{righe di codice totali}} * 100$;
- **Valore ottimo**: 100%;
- **Valore accettabile**: ≥ 80%;

6.3.2.1.1. Branch Coverage(MPD-BC)

- **Descrizione**: è un sottoinsieme del code coverage e misura la percentuale di rami delle condizioni che sono stati eseguiti durante i test. Un valore alto indica che il codice è stato testato in modo approfondito e che è meno probabile che contenga errori;
- **Come calcolarlo**: $\text{Branch Coverage} = \frac{\text{Branch eseguiti}}{\text{Branch totali}} * 100$;
- **Valore ottimo**: ≥ 80%;
- **Valore accettabile**: ≥ 50%;

6.3.2.1.2. Statement Coverage(MPD-SC)

- **Descrizione**: è un sottoinsieme di code coverage e misura la percentuale di istruzioni che sono state eseguite durante i test. Un valore alto indica che il codice è stato testato in modo approfondito e che è meno probabile che contenga errori;
- **Come calcolarlo**: $\text{Statement Coverage} = \cdot * 100$;
- **Valore ottimo**: ≥ 80%;
- **Valore accettabile**: ≥ 60%;

6.3.2.1.3. Failure Tolerance(MPD-FT)

- **Descrizione**: misura la capacità del prodotto di mantenere un livello di prestazioni accettabile anche in caso di guasti o malfunzionamenti. Un valore alto indica che il prodotto è in grado di gestire i guasti senza compromettere le funzionalità principali;
- **Valore ottimo**: 100%;
- **Valore accettabile**: 100%;

6.3.2.1.4. Failure Frequency(MPD-FF)

- **Descrizione**: misura la frequenza con cui si verificano guasti o malfunzionamenti nel prodotto. Un valore basso indica che il prodotto è affidabile e presenta pochi problemi;
- **Come calcolarlo**: $\text{Failure Frequency} = \frac{\text{Numero di malfunzionamenti}}{\text{Tempo totale}}$;
- **Valore ottimo**: 0;
- **Valore accettabile**: 0;

6.3.2.1.5. Mean Time Between Failure(MPD-MTBF)

- **Descrizione**: misura il tempo medio tra un guasto e il successivo. Un valore alto indica che il prodotto è affidabile e presenta pochi guasti;
- **Come calcolarlo**: $\text{Mean Time Between Failure} = \frac{\text{Tempo di attività totale}}{\text{Numero di incidenti}}$ dove:
  - Il tempo di attività totale è il tempo totale per cui il sistema rimane in funzione senza errori;
  - Il numero totale di guasti è il numero degli errori di sistema che si sono verificati durante il periodo specificato;
- **Valore ottimo**: ≥ 72h;
- **Valore accettabile**: ≥ 48h;

6.3.2.1.6. Disponibilità Sistema(MPD-DS)

- **Descrizione**: indica la percentuale di tempo in cui il sistema è operativo. Un valore alto indica che il sistema è affidabile e che è disponibile per l'utente;
- **Come calcolarlo**: $\text{Disponibilità Sistema} = \frac{\text{MTBF}}{(\text{MTBF}+\text{MTTR})}$ dove:
  - **MTBF** = Mean Time Between Failure = tempo medio tra un guasto e il successivo;
  - **MTTR** = Mean Time To Repair = tempo medio di riparazione;
- **Valore ottimo**: ≥ 99.9%;
- **Valore accettabile**: ≥ 90%;

6.3.3. Usabilità

6.3.3.0.1. Tempo di Apprendimento(MPD-TA)

- **Descrizione**: indica il tempo necessario per un utente base per apprendere come utilizzare il prodotto. Un valore basso indica che il prodotto è facile da usare e richiede poco tempo per essere appreso. Viene calcolato con sessioni di test con utenti;
- **Valore ottimo**: ≤5 min;
- **Valore accettabile**: ≤15 min (utente base);

6.3.3.0.2. Errori Utente/Azione(MPD-EUA)

- **Descrizione**:indica il numero di errori commessi dagli utenti durante l'utilizzo del prodotto. Un valore basso indica che il prodotto è intuitivo e facile da usare. Viene calcolato tramite log delle interazioni;
- **Valore ottimo**: 0;
- **Valore accettabile**: ≤0.5 errori/azione;

6.3.3.0.3. Task Success Rate(MPD-TSR)

- **Descrizione**: indica la percentuale di task completati con successo dagli utenti. Un valore alto indica che il prodotto è facile da usare e che gli utenti riescono a completare le azioni richieste. Viene calcolato con sessioni di test con utenti;
- **Valore ottimo**: 100%;
- **Valore accettabile**: ≥ 75%;

6.3.4. Efficienza

6.3.4.0.1. Tempo Risposta API(MPD-TRA)

- **Descrizione**: misura il tempo di risposta delle API per il 90% delle richieste. Un valore basso indica che il sistema risponde velocemente alle richieste degli utenti;
- **Valore ottimo**: ≤ 200 ms;
- **Valore accettabile**: ≤ 500 ms;

6.3.4.0.2. Memoria Processo(MPD-MP)

- **Descrizione**: indica l'utilizzo della memoria da parte del sistema. Un valore basso indica che il sistema utilizza in modo efficiente le risorse disponibili;
- **Valore ottimo**: ≤ 256 MB;
- **Valore accettabile**: ≤ 512 MB;

6.3.4.0.3. Consumo Energetico(MPD-CE)

- **Descrizione**: indica il consumo energetico del sistema. Un valore basso indica che il sistema consuma poca energia;
- **Valore ottimo**: ≤ 1% batteria/min;
- **Valore accettabile**: ≤ 2% batteria/min;

6.3.5. Manutenibilità

6.3.5.0.1. Complessità Ciclomatica(MPD-CC)

- **Descrizione**: misura la complessità del codice. Un valore basso indica che il codice è semplice e facile da mantenere;
- **Come calcolarlo**: $\mathrm{Complessità\ Ciclomatica} = \mathrm{E} - \mathrm{N} + \mathrm{P}$ dove:
  - E = numero di archi nel grafo di controllo;
  - N = numero di nodi nel grafo di controllo;

- P = numero di componenti connesse da ogni arco;
- **Valore ottimo**: ≤ 10;
- **Valore accettabile**: ≤ 15 per modulo;

6.3.5.0.2. Debito Tecnico(MPD-DT)

- **Descrizione**: misura la percentuale di debito tecnico rispetto al codice totale (spesso correlato alla presenza di debito tecnico). Un valore basso indica che il codice è ben strutturato e non presenta problemi tecnici;
- **Come calcolarlo**: $\text{Debito Tecnico} = \frac{\text{Costo di rimozione del debito}}{\text{Costo totale di sviluppo}} * 100$;
- **Valore ottimo**: ≤ 5%;
- **Valore accettabile**: ≤ 15%;

6.3.5.0.3. Code Smell Density(MPD-CSD)

- **Descrizione**: indica il numero di «code smells» (cattive pratiche di codifica) per 100 righe di codice. Un valore basso indica che il codice è ben strutturato e non presenta problemi tecnici;
- **Valore ottimo**: 0 smell;
- **Valore accettabile**: ≤ $\frac{5 \text{ smell}}{100 \text{ righe}}$;

6.3.5.0.4. Tempo Fix Bug(MPD-TFB)

- **Descrizione**: misura il tempo medio per risolvere un bug critico. Un valore basso indica che il team è in grado di risolvere i bug in modo rapido ed efficiente;
- **Come calcolarlo**:$\text{Tempo Fix Bug} = \frac{\text{Tempo totale di riparazione bug}}{\text{Numero totale di bug riparati}}$.
- **Valore ottimo**: ≤ 2 ore;
- **Valore accettabile**: ≤ 4 ore (critico);

6.3.6. Sicurezza

6.3.6.0.1. Tasso di Autenticazione Fallita

- **Descrizione**: misura la percentuale di tentativi di autenticazione falliti. Un valore basso indica che il sistema è sicuro e che è difficile per gli utenti non autorizzati accedere al sistema;
- **Valore ottimo**: ≤ 1%;
- **Valore accettabile**: ≤ 5%;

6.3.6.0.2. Crittografia Dati

- **Descrizione**: misura il livello di crittografia dei dati sensibili. Un valore alto indica che i dati sono protetti e che è difficile per gli utenti non autorizzati accedere ai dati sensibili;
- **Valore ottimo**: 100% dati sensibili;
- **Valore accettabile**: 100% dati sensibili;