



BugBusters

Email: bugbusters.unipd@gmail.com

Gruppo: 4

Università degli Studi di Padova

Laurea in Informatica

CORSO: Ingegneria del Software

Anno Accademico: 2025/2026

Norme di Progetto_G

Versione 1.0.0

Stato	Approvato per RTB _G
Verificatore _G	Alberto Pignat, Luca Slongo
Redattore	Alberto Autiero
Destinatari	BugBusters, Prof. Tullio Vardanega, Prof. Riccardo Cardin

Abstract

Documento contenente le Norme di Progetto_G adottate dal team BugBusters per lo sviluppo del progetto_G Nexum proposto dall'azienda Eggon. Il documento include metodologie di lavoro, standard di codifica, processi di sviluppo e gestione del progetto_G.

Registro delle modifiche

Versione	Data	Descrizione	Redatto	Verificato	Approvato
1.0.0	30/01/2025	Approvazione documento	-		Luca Slongo
0.2.0	30/01/2025	Verifica documento	-	Alberto Pignat	-
0.1.7	29/01/2026	Aggiunti ai termini presenti nel Glossario _g la G	Luca Slongo	-	-
0.1.6	19/1/2026	Terminate metriche finali	Linor Sadè	-	-
0.1.5	18/1/2026	Aggiornamento sezione qualità _g , aggiunti dettagli su metriche di processo	Linor Sadè	-	-
0.1.4	13/1/2026	continuo sezione qualità _g , aggiunti: standard e modelli, principi di qualità _g del processo, metriche di processo	Linor Sadè	-	-
0.1.3	12/1/2026	Inizio sezione qualità _g , modificato inizio precedente	Linor Sadè	-	-
0.1.2	4/1/2026	Aggiunte parti mancanti relative al Piano di Qualifica _g . Ulteriori correzioni al contenuto precedente.	Linor Sadè	-	-
0.1.1	3/1/2026	Correzioni minime nel documento e aggiunte alcune parti mancanti	Linor Sadè	-	-
0.1.0	30/12/2025	Verificato	-	Luca Slongo	-
0.0.10	22/12/2025	Aggiunto Piano di Progetto _g e di Qualifica	Marco Piro	-	-
0.0.9	18/12/2025	Eliminato i Processi di acquisizione ed aggiunto le lettere di Candidatura _g e Presentazione	Marco Piro	-	-
0.0.8	04/12/2025	Conclusione momentanea della sezione relativa ai processi di supporto	Alberto Pignat	-	-
0.0.7	02/12/2025	Avanzamento processi di supporto: Configurazioni e Qualifica	Alberto Pignat	-	-
0.0.6	28/11/2025	Iniziati processi di supporto: documentazione	Marco Favero	-	-
0.0.5	27/11/2025	Correzione strutturale e sistemazione lavoro svolto	Marco Favero	-	-
0.0.4	20/11/2025	Finita la parte di Documentazione prodotta in processi di fornitura	Marco Favero	-	-

0.0.3	19/11/2025	Avanzamento, parte di documentazione prodotta	Marco Favero	-	-
0.0.2	16/11/2025	Introduzione, processi di acquisizione e iniziati processi di fornitura	Marco Favero	-	-
0.0.1	04/11/2025	Prima stesura della struttura del documento dopo l'aggiudicazione dell'appalto per il capitolato C5 Nexum dell'azienda Eggon	Alberto Autiero	-	-

Indice

1 Introduzione	6
1.1 Scopo del documento	6
1.2 Scopo del prodotto _G	6
1.3 Glossario _G	6
1.3.1 Riferimenti normativi	7
1.3.2 Riferimenti informativi	7
2 Processi Primari	8
2.1 Processi di fornitura	8
2.1.1 Scopo	8
2.1.2 Attività	8
2.1.3 Strumenti di Supporto	9
2.1.4 Comunicazione e Organizzazione	9
2.1.5 Documentazione Prodotta	9
2.1.5.1 Glossario _G	9
2.1.5.2 Dichiarazione degli impegni	10
2.1.5.3 Valutazione dei capitolati	10
2.1.5.4 Analisi dei Requisiti _G	11
2.1.5.5 Norme di Progetto _G	11
2.1.5.6 Lettera di Candidatura _G	12
2.1.5.7 Lettera di Presentazione	12
2.1.5.8 Verbali	13
2.1.5.9 Piano di Progetto _G	13
2.1.5.10 Piano di Qualifica _G	13
2.2 Processi di sviluppo	14
2.2.1 Attività previste	14
2.2.2 Analisi dei Requisiti _G	15
2.2.2.1 Casi d'uso _G	15
2.2.2.2 Requisiti _G	16
3 Processi di Supporto	16
3.1 Documentazione	16
3.1.1 Scopo	16
3.1.2 Strumenti Utilizzati	17
3.1.3 Documenti Prodotti	17
3.1.4 Struttura di un documento	17
3.1.4.1 Struttura dei verbali	18
3.1.4.2 Struttura dei Diari di Bordo	18
3.1.5 Denominazione dei documenti	19
3.1.6 Produzione	19
3.2 Gestione della configurazione	20
3.2.1 Strumenti utilizzati	20
3.2.2 Controllo della configurazione	20
3.2.3 Registrazione stato di configurazione	21
3.3 Qualifica	21
3.3.1 Verifica _G	21
3.3.1.1 Attività di verifica _G	21
3.3.1.2 Analisi Statica	21
3.3.1.3 Analisi Dinamica	22

3.3.1.4	Test _G di Unità	23
3.3.1.5	Test _G di Regressione	23
3.3.1.6	Test _G di Integrazione	23
3.3.1.7	Test _G di Sistema	23
3.3.2	Validazione _G	24
3.3.2.1	Processo di Validazione _G	24
4	Processi Organizzativi	24
4.1	Descrizione e Scopo	24
4.2	Attività	24
4.3	Gestione degli Sprint _G	24
4.4	Ruoli	25
4.4.1	Responsabile _G	25
4.4.2	Ammiriatore _G	25
4.4.3	Analista _G	25
4.4.4	Progettista _G	25
4.4.4.1	Programmatore _G	26
4.4.5	Verificatore _G	26
4.5	Tracciamento Ore	26
4.6	Tracciamento Azioni	26
4.7	Comunicazione	27
4.7.1	Comunicazione Interna	27
4.7.2	Comunicazione Esterna	27
4.8	Riunioni	27
4.8.1	Riunioni Interne	27
4.8.2	Riunioni Esterne	28
5	Qualità_G del Software	28
5.1	Standard di riferimento e modelli	28
5.2	Processo di Valutazione della Qualità _G	29
5.3	Principi della Qualità _G del processo	29
5.4	Metriche per la qualità _G	30
5.4.1	Qualità _G del Processo	30
5.4.1.1	Processi primari	30
5.4.1.2	Processi di supporto	32
5.4.1.3	Processi organizzativi	32
5.4.2	Qualità _G del Prodotto _G	33
5.4.2.1	Adeguatezza funzionale	33
5.4.2.2	Affidabilità _G	33
5.4.2.3	Efficienza _G	33
5.4.2.4	Usabilità	33
5.4.2.5	Manutenibilità _G	34
5.4.2.6	Portabilità	34

1 Introduzione

1.1 Scopo del documento

Questo documento definisce le Norme di Progetto_g adottate dal team BugBusters per lo sviluppo del progetto_g Nexum, proposto dall'azienda Eggon. Le Norme di Progetto_g includono metodologie di lavoro, standard di codifica, processi di sviluppo e gestione del progetto_g, al fine di garantire un approccio strutturato e coerente durante l'intero ciclo di vita del progetto_g.

Per strutturare il nostro way of working_g, faremo riferimento alle best practice_g suggerite dallo standard ISO/IEC 12207:1995 adattandole alle esigenze specifiche del nostro team e del progetto_g Nexum. In particolare lo standard identifica tre tipologie di processi:

- Processi primari: processi direttamente coinvolti nella creazione del prodotto_g software
- Processi di supporto: processi che supportano i processi primari
- Processi organizzativi: processi che gestiscono e coordinano le attività del team

La combinazione di questi processi ci permetterà di gestire in modo efficace lo sviluppo del progetto_g Nexum. La stesura di questo documento mira a fornire una guida chiara e condivisa per tutti i membri del team, assicurando che le attività di sviluppo siano svolte in modo efficiente e conforme agli standard di qualità_g previsti. Il documento è redatto in maniera incrementale: verrà aggiornato e ampliato progressivamente durante lo sviluppo del progetto_g per riflettere decisioni, modifiche e miglioramenti adottati dal team.

1.2 Scopo del prodotto_g

Nexum è una piattaforma con un modulo_g dedicato alle comunicazioni interne, alla raccolta di feedback e alla timbratura digitale. Include una messaggistica top-down con tracciamento delle letture, un builder per survey con logiche di ramificazione e dashboard_g in tempo reale; inoltre un sistema di timbratura via badge o dispositivi mobili con regole automatiche per il controllo e l'aggregazione delle ore. Le anagrafiche centralizzate, con ruoli e permessi, permettono una gestione granulare degli accessi e l'integrazione dei moduli, garantendo un flusso informativo coerente, tracciabile e adattabile alle esigenze operative. Il nostro progetto_g mira alla creazione di un'applicazione stand-alone_g (rispetto all'applicazione Nexum già esistente) che implementa un modulo_g di AI_g assistant generativo per la scrittura di comunicazioni e l'AI Co-Pilot_g per i Cdl_g. In particolare quest'ultimo deve essere in grado di riconoscere i documenti caricati dagli utenti, estrarne le informazioni rilevanti, capire la tipologia, destinatari e consegnarli in modo massivo. Per ulteriori informazioni sul progetto_g si rimanda all'Analisi dei Requisiti_g al link (AGGIUNGERE LINK). Il nostro obiettivo è realizzare questo progetto_g entro il 21 marzo 2026 con un budget di 12.790 euro.

1.3 Glossario_g

Il glossario_g raccoglie e definisce i termini, gli acronimi e le abbreviazioni impiegati nel documento e nel progetto_g Nexum. L'obiettivo è fornire definizioni univoche per ridurre ambiguità, garantire coerenza terminologica tra i membri del team e facilitare l'onboarding di nuovi partecipanti.

Per i termini tecnici e specifici utilizzati in questo documento, si fa riferimento al glossario_g disponibile al seguente [link](#). Per maggiore usabilità e facilità di consultazione, il glossario_g è accessibile anche aprendo dal nostro sito web i vari documenti con il viewer pdf da noi sviluppato.

1.3.1 Riferimenti normativi

- **Capitolato_gd'appalto C5: Nexum - Piattaforma di consulenza e documentazione previdenziale**
<https://www.math.unipd.it/~tullio/IS-1/2025/Progetto/C5.pdf>

1.3.2 Riferimenti informativi

- **Glossario_g:**
<https://github.com/BugBustersUnipd/DocumentazioneSWE/blob/main//RTB/GLOSSARIO/Glossario.pdf>

2 Processi Primari

L'obiettivo principale di BugBusters è la realizzazione di un prodotto_G software di alta qualità_G che soddisfi le esigenze del cliente e degli utenti finali. Per raggiungere questo obiettivo, è indispensabile basarsi su un modello di riferimento che definisca processi chiari da seguire.

L'adozione di un simile framework metodologico, che indirizza ad esempio le fasi di acquisizione e di costruzione del software, è ciò che permette di passare da un semplice programma funzionante a un prodotto_G di valore e di lunga durata. Basandosi dunque sullo standard ISO/IEC 12207:1995, il team BugBusters ha deciso di adottare i seguenti processi primari:

- Processi di fornitura
- Processi di sviluppo

2.1 Processi di fornitura

2.1.1 Scopo

Lo scopo del processo di fornitura è definire e regolamentare l'insieme delle attività preliminari necessarie ad avviare il progetto_G in modo controllato, condiviso e verificabile. In particolare, tale processo ha l'obiettivo di chiarire i requisiti_G richiesti dalla proponente_G, identificare vincoli e risorse, pianificare le modalità operative, stabilire gli strumenti di lavoro e formalizzare la documentazione necessaria, così da garantire una corretta base metodologica e contrattuale per le successive fasi di sviluppo.

2.1.2 Attività

La fornitura prevede varie attività, in particolare:

- **Analisi del capitolato_G proposto:** raccolta di informazioni, requisiti_G, tecnologie e vincoli presenti nel capitolato_G d'appalto. In questa fase si pongono domande alla proponente_G per chiarire eventuali dubbi o ambiguità, processo utile alla scelta finale.
- **Stima delle risorse:** definizione delle tempistiche, dei costi totali e delle risorse necessarie per la realizzazione del progetto_G.
- **Analisi dei Requisiti_G e contrattazione con la proponente_G:** identificazione e documentazione dei requisiti_G funzionali e non funzionali del software, analisi delle aspettative del proponente_G e negoziazione di eventuali modifiche o aggiunte. Definizione del Minimum Viable Product (MVP).
- **Pianificazione del progetto_G:** suddivisione del progetto_G in fasi, definizione delle milestone_G aggiuntive (oltre a RTB_G e PB_G), assegnazione dei compiti ai membri del team e pianificazione delle attività. Individuazione degli strumenti di lavoro e delle metodologie da adottare; definizione del Proof of Concept (PoC)_G.
- **Documentazione:** redazione di tutti i documenti necessari per formalizzare la fornitura, come il Piano di Progetto_G, l'Analisi dei Requisiti_G, Norme di Progetto_G e altri documenti di supporto. La documentazione prodotta sarà utilizzata sia come strumento di lavoro interno al team sia come riferimento e strumento di controllo da parte della proponente_G.
- **Comunicazione con la proponente_G:** stabilire canali di comunicazione efficaci per garantire un flusso informativo continuo e trasparente. Prevedere incontri regolari per aggiornamenti sullo stato del progetto_G, discussione di eventuali problemi e raccolta di feedback.

- **Revisione e approvazione_G:** sottoporre tutta la documentazione prodotta alla revisione e approvazione_G della proponente_G e del committente_G, assicurando l'allineamento sugli obiettivi e le aspettative prima di procedere con le fasi successive del progetto_G.
- **Consegna e chiusura della fase di fornitura:** consegna di quanto prodotto_G durante la fase di fornitura al proponente_G, garantendo che tutti i documenti siano completi e corretti. Completata la consegna, si procede alla chiusura formale della fase.

2.1.3 Strumenti di Supporto

Per supportare le attività di fornitura, il team BugBusters usufruisce di vari strumenti:

- **GitHub_G:** Per la documentazione collaborativa, il versionamento dei documenti, la produzione asincrona, il sistema di ticketing e delle project board.
- **GitLab:** Per il codice già esistente fornito dalla proponente_G, utile all'implementazione_G di un prodotto_G che possa essere facilmente integrabile alla piattaforma già esistente.
- **Discord_G:** Per le riunioni di team.
- **WhatsApp:** Per comunicazioni rapide e aggiornamenti interni al gruppo.
- **Google Calendar:** Per la pianificazione delle riunioni e delle scadenze.
- **Telegram e Gmail:** Per la comunicazione con la proponente_G e per inviare documenti ufficiali.

2.1.4 Comunicazione e Organizzazione

Le comunicazioni con la proponente_G avvengono solitamente a cadenza bisettimanale mercoledì alle 15:00, salvo imprevisti o esigenze particolari. I verbali relativi agli incontri con la proponente_G vengono condivisi e archiviati nel repository_G della documentazione per garantire trasparenza e reperibilità. Tutte le decisioni rilevanti emerse negli incontri con la proponente_G devono essere formalmente riportate e documentate con il sistema di ticketing su GitHub_G e nella documentazione ufficiale. Eventuali problemi critici o dubbi vengono comunicati immediatamente sui canali concordati con la proponente_G, in modo da garantirne la tempestività nelle azioni correttive.

2.1.5 Documentazione Prodotta

Durante la fase di fornitura, il team BugBusters produce e mantiene aggiornata la seguente documentazione: AGGIUNGERE I LINK DEI DOCUMENTI QUI SOTTO.

- [Glossario_G](#)
- [Analisi dei Requisiti_G](#)
- [Piano di Progetto_G](#)
- [Piano di Qualifica_G](#)

2.1.5.1 Glossario_G

Per facilitare la lettura e la comprensione dei documenti di progetto_G, viene redatto un glossario_G che raccoglie e definisce i termini tecnici, gli acronimi e le abbreviazioni utilizzate. Questo strumento è fondamentale per garantire una comunicazione chiara e univoca tra tutti i membri del team,

con la proponente_G, docenti (il committente_G) e con i lettori esterni. Per una consultazione rapida durante la visualizzazione dei documenti, il glossario_G è accessibile anche tramite il viewer PDF sviluppato dal team che si trova nella repository_G web ufficiale della documentazione accessibile al seguente [link](#).

Campo	Dettaglio
Redattore_G	Amministratore _G
Destinatari	BugBusters, Eggon, Prof. Vardanega, Prof. Cardin
Uso	Interno ed Esterno

2.1.5.2 Dichiarazione degli impegni

La Dichiarazione degli Impegni definisce i ruoli, il monte ore previsto, la data di consegna prevista e il costo orario di ciascun membro del team BugBusters per la realizzazione del progetto_G Nexum, impegnandosi a rispettare tali condizioni durante l'intero ciclo di vita del progetto_G.

Campo	Dettaglio
Redattore_G	Responsabile _G
Destinatari	BugBusters, Eggon, Prof. Vardanega, Prof. Cardin
Uso	Esterno

2.1.5.3 Valutazione dei capitolati

Con questo documento si intende valutare i vari capitolati d'appalto proposti per il progetto_G di Ingegneria del Software, analizzandone punti di forza, debolezze e opportunità in modo da scegliere il capitolato_G più adatto alle competenze e agli interessi del team BugBusters. Per ogni capitolato_G vengono esaminati vari aspetti:

- Descrizione breve
- Caratteristiche funzionali
- Tecnologie proposte
- Chiarimenti e colloqui con l'azienda
- Interesse del team
- Punti di forza e debolezza

Campo	Dettaglio
Redattore_G	Responsabile _G
Destinatari	BugBusters, Prof. Vardanega, Prof. Cardin
Uso	Esterno

2.1.5.4 Analisi dei Requisiti_G

Il documento di Analisi dei Requisiti_G descrive in dettaglio le funzionalità_G e i vincoli. Questo documento fornisce:

- la definizione del contesto, degli stakeholder_G e degli attori coinvolti;
- l'elenco dei requisiti_G funzionali_G e non funzionali_G, classificati e dotati di codifica univoca e criteri di accettazione;
- casi d'uso_G e scenari principali con flussi e attori associati;
- vincoli tecnici, normativi e di integrazione con sistemi esterni;
- la prioritizzazione dei requisiti_G e l'identificazione del Minimum Viable Product (MVP);
- la strategia di tracciabilità_G e gestione delle modifiche (issue_G-tracker, versionamento e mappatura requisiti_G-test_G);
- i criteri e i metodi per la verifica_G e la validazione_G dei requisiti_G.

Campo	Dettaglio
Redattore_G	Analista _G
Destinatari	Eggon, Prof. Vardanega, Prof. Cardin
Uso	Esterno

2.1.5.5 Norme di Progetto_G

Il documento delle Norme di Progetto_G definisce le metodologie, gli standard e i processi che il team BugBusters adotterà durante lo sviluppo del progetto_G Nexum.

Campo	Dettaglio
Redattore_G	Amministratore _G
Destinatari	BugBusters, Prof. Vardanega, Prof. Cardin
Uso	Interno

2.1.5.6 Lettera di Candidatura_G

La Lettera di Candidatura_G rappresenta il documento formale con cui il gruppo BugBusters ha ufficializzato la propria candidatura_G per il capitolato_G proposto dall'azienda Eggon. Al suo interno sono sintetizzate le motivazioni che hanno determinato la scelta di questo specifico progetto_G, evidenziando i punti di interesse tecnologico e formativo, oltre a fornire i riferimenti al documento di dichiarazione degli impegni e valutazione dei capitolati.

Campo	Dettaglio
Redattore_G	Responsabile _G
Destinatari	BugBusters, Prof. Vardanega, Prof. Cardin
Uso	Esterno

2.1.5.7 Lettera di Presentazione

Il documento viene redatto in due distinte versioni, corrispondenti alle principali scadenze del progetto_G:

- Lettera di Presentazione per la **Requirements and Technology Baseline (RTB)_G**
- Lettera di Presentazione per la **Product Baseline (PB)_G**

Campo	Dettaglio
Redattore_G	Responsabile _G
Destinatari	BugBusters, Prof. Vardanega, Prof. Cardin
Uso	Esterno

2.1.5.8 Verbal

I verbali sono documenti di lavoro indispensabili per tracciare le decisioni, le discussioni e le azioni concordate durante le riunioni interne del team e le riunioni con la proponente_G Eggon. Si dividono in verbali interni, prodotti durante le riunioni del team, e verbali esterni, redatti dopo gli incontri con la proponente_G.

Campo	Dettaglio interni	Dettaglio esterni
Redattore_G	Responsabile _G	Responsabile _G
Destinatari	BugBusters, Prof. Vardanega, Prof. Cardin	BugBusters, Eggon, Prof. Vardanega, Prof. Cardin
Uso	Interno	Esterno

2.1.5.9 Piano di Progetto_G

Il Piano di Progetto_G rappresenta il documento di riferimento per la pianificazione strategica e operativa delle attività del gruppo BugBusters. Esso definisce la scansione temporale del lavoro suddiviso in sprint_G, l'allocazione delle risorse umane e strumentali, nonché l'analisi preventiva dei rischi_G con le relative strategie di mitigazione_G. Il documento ha inoltre la funzione fondamentale di monitorare l'andamento del progetto_G, riportando a ogni avanzamento il consuntivo_G delle ore produttive e dei costi sostenuti rispetto a quanto preventivato, garantendo così il pieno controllo sulle risorse impiegate.

Campo	Dettaglio interni	Dettaglio esterni
Redattore_G	Responsabile _G	Responsabile _G
Destinatari	BugBusters, Prof. Vardanega, Prof. Cardin	BugBusters, Eggon, Prof. Vardanega, Prof. Cardin
Uso	Interno	Esterno

2.1.5.10 Piano di Qualifica_G

Il Piano di Qualifica_G documenta la strategia adottata dal gruppo BugBusters per garantire la qualità_G del prodotto_G software che il gruppo deve realizzare e dei processi correlati. Esso definisce nel dettaglio le metodologie di verifica_G e validazione_G, gli standard di qualità_G di riferimento e le metriche utilizzate per il monitoraggio. Il documento riporta inoltre la pianificazione e gli esiti dei test_G effettuati, con l'obiettivo specifico di assicurare il rispetto dei requisiti_G e mantenere una copertura del codice (code coverage_G) pari o superiore all'80%.

Campo	Dettaglio interni	Dettaglio esterni
Redattore_G	Responsabile _G	Responsabile _G
Destinatari	BugBusters, Prof. Vardanega, Prof. Cardin	BugBusters, Eggon, Prof. Vardanega, Prof. Cardin
Uso	Interno	Esterno

2.2 Processi di sviluppo

Il processo di sviluppo definisce le attività tecniche e operative necessarie per la realizzazione del prodotto_G software che il gruppo deve realizzare, in conformità ai requisiti_G stabiliti durante la fase di fornitura. Questo processo include la progettazione, l'implementazione_G, il testing e la documentazione del software, garantendo che ogni fase sia eseguita secondo standard di qualità_G elevati e best practice_G del settore.

2.2.1 Attività previste

- **Implementazione_G del processo:** se non definito prima, il fornitore dovrebbe definire o scegliere un modello di ciclo di vita del software appropriato alla complessità del progetto_G;
- **Analisi dei Requisiti_G di sistema:** l'uso previsto del sistema da sviluppare deve essere analizzato per definire i requisiti_G di sistema. La specifica dei requisiti_G di sistema deve descrivere le funzioni e le capacità del sistema, i requisiti_G di business, organizzativi e degli utenti, nonché i requisiti_G relativi a sicurezza_G, protezione, fattori umani (ergonomia), interfacce, operatività e manutenzione. Devono inoltre essere inclusi i vincoli di progettazione e i requisiti_G di qualificazione;
- **Design architetturale del sistema:** deve essere definita un'architettura di alto livello del sistema, che identifichi gli elementi hardware, software e le operazioni manuali. Tutti i requisiti_G di sistema devono essere correttamente assegnati a tali elementi. A partire da questi, devono essere individuati gli elementi di configurazione hardware, software e le operazioni manuali;
- **Analisi dei Requisiti_G del software:** il fornitore deve stabilire e documentare i requisiti_G del software, includendo caratteristiche di qualità_G: come le caratteristiche funzionali, i requisiti_G di sicurezza_G e di interfaccia_G esterna del software;
- **Design architetturale del software:** il fornitore deve tradurre i requisiti_G dell'elemento software in un'architettura che ne descriva la struttura di alto livello e identifichi i componenti software. Tutti i requisiti_G devono essere assegnati ai componenti software e ulteriormente dettagliati per supportare la progettazione di dettaglio;
- **Design di dettaglio del software:** il fornitore deve sviluppare il progetto_G di dettaglio per ciascun componente software. I componenti devono essere ulteriormente suddivisi in unità software implementabili, compilabili e testabili. Tutti i requisiti_G software devono essere correttamente assegnati dalle componenti alle unità software;

- **Scrittura e test_g del codice:** il fornitore deve scrivere il codice per tutte le componenti individuate, e testarle esaustivamente;
- **Integrazione del software:** il fornitore deve sviluppare un piano di integrazione per assemblare le unità e i componenti software nel prodotto_gsoftware finale. Il piano deve includere i requisiti_gdi test_g, le procedure, i dati necessari, le responsabilità e la pianificazione delle attività;
- **Test_g di qualifica del software:** il fornitore deve eseguire i test_gdi qualificazione in conformità ai requisiti_gdi qualificazione del software. Deve essere verificato che l'implementazione_gdi ogni requisito_gsoftware sia testata per garantirne la conformità;
- **Integrazione del sistema:** gli elementi di configurazione software devono essere integrati con quelli hardware, con le operazioni manuali e con altri sistemi, se necessario, all'interno del sistema complessivo. Gli insiemi risultanti devono essere testati progressivamente rispetto ai requisiti_gprevisti;
- **Testi di qualifica del sistema:** devono essere eseguiti i test_gdi qualificazione del sistema in conformità ai requisiti_gdi qualificazione definiti. Deve essere verificato che l'implementazione_gdi ogni requisito_gsistema sia testata per garantirne la conformità e che il sistema sia pronto per la consegna;
- **Installazione del software:** il fornitore deve predisporre un piano per l'installazione del prodotto_gsoftware nell'ambiente di destinazione previsto dal contratto. Devono essere definiti e resi disponibili le risorse e le informazioni necessarie all'installazione. Secondo quanto stabilito dal contratto, il fornitore deve supportare il committente_gnelle attività di configurazione iniziale. Qualora il nuovo software sostituisca un sistema esistente, il fornitore deve supportare l'esercizio in parallelo, se richiesto;
- **Supporto per l'approvazione_gdel software:** il fornitore deve supportare il committente_gnella revisione e nei test_gdi accettazione del prodotto_gsoftware. Le attività di accettazione devono tenere conto dei risultati delle revisioni congiunte, degli audit, dei test_gdi qualificazione del software e, se eseguiti, dei test_gdi qualificazione del sistema;

2.2.2 Analisi dei Requisiti_g

È un'attività fondamentale della Requirements and Technology Baseline (RTB)_ge consiste nella raccolta, analisi e documentazione di tutti i requisiti_gche il sistema sviluppato dovrà soddisfare. Tale attività è documentata nell'apposito documento di Analisi dei Requisiti_g(TODO inserire link), che servirà poi come riferimento per guidare tutte le fasi successive di progettazione, implementazione_ge testing del software. Nella sezione seguente viene illustrato il sistema di nomenclatura adottato per identificare in modo univoco i casi d'uso_g.

2.2.2.1 Casi d'uso_g

UC-SezionePrincipale.Secondario

dove:

- **UC:** sta per Use Case_g(Caso d'Uso_g);
- **SezionePrincipale:** abbiamo identificato quattro moduli principali identificati da un numero da 0 a 3;

- **Principale:** è identificato da una lettera dell'alfabeto, e rappresenta un caso d'uso primario non correlato ad altri casi d'uso_G, se non tramite inclusione;
- **Secondario:** è identificato da un numero, ed è correlato esclusivamente a un caso d'uso principale;

Ad esempio, il caso d'uso "UC-1A.1" rappresenta un caso d'uso secondario (1) correlato al caso d'uso principale "UC-1A", che a sua volta è associato alla sezione principale 1 del sistema. Ogni caso d'uso viene anche descritto da un nome che lo riassume e da una descrizione dettagliata.

2.2.2.2 Requisiti_G

I requisiti_G vengono individuati dopo aver identificato i casi d'uso_G, e sono classificati nel modo seguente:

Tipologia-Numero

dove:

- **Tipologia:** rappresenta la categoria del requisito_G, e può essere funzionale (RF), requisiti_G di qualità_G (RQ), requisiti_G di vincolo_G (RV) o requisiti_G prestazionali_G (RP);
- **Numero:** è un numero progressivo che identifica in modo univoco il requisito_G all'interno della sua tipologia.

Ad ogni requisito_G viene poi associata una descrizione dettagliata, la fonte del requisito_G (interna, di capitolato_G, da colloquio con la proponente_G, o da Use Case_G), e la priorità_G (Obbligatorio_G, Desiderabile_G, Opzionale_G).

3 Processi di Supporto

L'obiettivo dei processi di supporto è fornire le risorse, gli strumenti e le metodologie necessarie per supportare efficacemente i processi primari di sviluppo e fornitura del software. I processi di supporto sono essenziali per garantire che il team segua pratiche di lavoro coerenti e sia sempre allineato agli obiettivi che si prefigge di raggiungere. Per esempio la documentazione svolge un ruolo cruciale nel mantenere la tracciabilità_G delle decisioni, eliminare qualsiasi ambiguità nella comprensione dei requisiti_G tra i membri del team, e assicurare che tutte le informazioni rilevanti siano facilmente accessibili e aggiornate.

3.1 Documentazione

3.1.1 Scopo

Lo scopo del processo di documentazione è definire le linee guida e le metodologie per la creazione, gestione e manutenzione della documentazione di progetto_G. Questo processo mira a garantire che tutta la documentazione prodotta sia chiara, coerente, accessibile e aggiornata, facilitando la comunicazione tra i membri del team, la proponente_G e gli stakeholder_G esterni. Una documentazione ben strutturata supporta l'efficace gestione del progetto_G, la tracciabilità_G delle decisioni e la conformità agli standard di qualità_G previsti.

3.1.2 Strumenti Utilizzati

Per supportare le attività di documentazione, il team BugBusters utilizza i seguenti strumenti:

- **GitHub_g**: Per la gestione collaborativa della documentazione, il versionamento dei documenti e la produzione asincrona. Vengono utilizzate le funzionalità_g di issue_g tracking e project board per tracciare le attività di documentazione. Inoltre BugBusters fa ampio uso dei branch per garantire che le modifiche alla documentazione siano revisionate prima di essere integrate nella versione principale.
- **LaTeX_g**: Per la stesura di documenti tecnici e formali, garantendo un formato professionale e coerente. È stata stabilita un'identità visiva comune per tutti i documenti, inclusi template_g, stili e convenzioni di formattazione.
- **Viewer PDF sviluppato dal team**: Per facilitare la consultazione della documentazione prodotta.

3.1.3 Documenti Prodotti

Di seguito l'elenco dei documenti obbligatori, coerenti con le indicazioni del capitolo_g C4. Ogni documento verrà mantenuto aggiornato e versionato sul repository_g del progetto_g.

1. **Norme di Progetto_g**: Il presente documento, dove vengono definite linee guida metodologiche, standard di lavoro e convenzioni adottate dal team.
2. **Piano di Progetto_g**: Documento dettagliato con allocazione delle risorse umane e materiali per le milestone_g e le consegne.
3. **Piano di Qualifica_g**: Strategia e metriche di collaudo; obiettivo minimo di qualità_g: copertura dei test_g pari o superiore al 80%.
4. **Analisi dei Requisiti_g**: Specifica dei requisiti_g funzionali_g e non funzionali_g, casi d'uso_g principali e criteri di accettazione.
5. **Glossario_g**: Elenco e definizioni dei termini tecnici e degli acronimi usati nel progetto_g.
6. **Lettera di Presentazione**: Documento di presentazione formale rivolto a RTB_g, completo e firmato, da utilizzare per la consegna iniziale.
7. **Verbali (interni/esterni)**: Registrazione degli incontri e delle decisioni, sia per le riunioni interne sia per i confronti con la proponente_g.

Tutti i documenti saranno sottoposti a revisione interna e tracciati tramite GitHub_g (issue_g e pull request) per garantirne la tracciabilità_g e la storicizzazione.

3.1.4 Struttura di un documento

Un documento generalmente, esclusi verbali e diario di bordo, ha questa struttura:

- **Copertina**: Logo e informazioni del team, titolo del documento, redattori, verificatori, versione, data di ultima modifica, destinatari.
- **Registro delle modifiche**: Tabella che traccia le versioni del documento, le modifiche apportate, le date e i responsabili.
- **Indice**: Elenco delle sezioni e sottosezioni con i numeri di pagina.

- **Introduzione:** Scopo del documento, scopo del prodotto_g, glossario_ge riferimenti.
- **Corpo del documento:** Contenuto principale, suddiviso in sezioni e sottosezioni.
- **Appendici:** Materiale supplementare, come diagrammi, tabelle o esempi.

3.1.4.1 Struttura dei verbali

I verbali seguono una struttura semplificata:

- **Copertina:** Logo e informazioni del team, titolo del documento, redattore_g, verificatore_g, versione, data, uso e destinatari.
- **Abstract:** Elenco degli argomenti da trattare durante la riunione.
- **Indice:** Elenco delle sezioni e sottosezioni con i numeri di pagina.
- **Informazioni Generali:** Data, ora, luogo, partecipanti e assenti.
- **Ordine del Giorno:** Pianificazione degli argomenti da discutere.
- **Svolgimento:** Dettagli delle discussioni ed eventuali argomenti fuori programma.
- **Tabella delle decisioni e delle azioni:** elenco strutturato delle decisioni prese, con descrizione e incaricato/responsabile.
- **Esito Riunione:** Sintesi dei risultati e delle conclusioni.

Da questo schema è possibile notare che i verbali non hanno il registro delle modifiche. Questo perché sono considerati documenti di lavoro e non documenti formali che richiedono una tracciabilità_g delle versioni.

Dal momento che il ruolo del responsabile_g costituisce il maggior dispendio di budget all'interno del progetto_g e che la redazione dei verbali è una delle sue principali responsabilità, è stato deciso di adottare una struttura standardizzata per i verbali al fine di ottimizzare il tempo impiegato nella loro redazione, introducendo anche delle macro per LaTeX_g che permettono di velocizzare ulteriormente la creazione dei verbali.

3.1.4.2 Struttura dei Diari di Bordo

I diari di bordo sono diapositive presentate durante la lezione settimanale di Ingegneria del Software dedicata alla discussione e condivisione delle difficoltà e dubbi incontrati durante lo sviluppo del progetto_g. Per questo motivo i diari di bordo hanno pochi punti:

- **Copertina:** Logo e informazioni del team, titolo del documento.
- **Difficoltà incontrate:** Lista delle difficoltà incontrate durante la settimana.
- **Dubbi:** Elenco di domande e dubbi da porre ai docenti.

Visto che i diari di bordo sono da presentare in pochi minuti non hanno bisogno di una struttura complessa, bensì di essere sintetici e diretti per permettere a docenti e colleghi di capire velocemente quali sono i problemi riscontrati.

3.1.5 Denominazione dei documenti

I documenti prodotti dal team BugBusters seguono una convenzione di denominazione standardizzata per garantire coerenza, facilità di identificazione e tracciabilità_G.

La struttura del nome dei **verbali** è la seguente:

`<TIPO>_<DATA>.pdf`

Dove <TIPO> può essere:

- **VI**: Verbale Interno_G
- **VE**: Verbale Esterno_G

E <DATA> è la data della riunione nel formato GG-MM-AAAA.

Esempio: `VE_15-01-2026.pdf` rappresenta il verbale esterno_G della riunione del 15 gennaio 2026.

La struttura del nome dei **diari di bordo** è la seguente:

`DB-<DATA>.pdf`

Dove <DATA> è la data di presentazione del diario nel formato GG-MM-AAAA.

Esempio: `DB-22-01-2026.pdf` rappresenta il diario di bordo presentato il 22 gennaio 2026.

Per gli altri **documenti formali** vengono utilizzati nomi descrittivi chiari e concisi del documento stesso, ad esempio Norme di Progetto_G.pdf, Piano di Progetto_G.pdf, Analisi dei Requisiti_G.pdf ecc.

3.1.6 Produzione

La produzione della documentazione segue un processo strutturato per garantire qualità_G, coerenza e tracciabilità_G.

- **Pianificazione**: A seconda del documento, viene pianificata la sua creazione in base alle milestone_G del progetto_G e alle esigenze del team. Secondo il tipo di documento, viene assegnato un redattore_G e un verificatore_G secondo i ruoli definiti nel way of working_G (vedi sezione 2.1.5).
- **Creazione del branch di lavoro**: Per ogni documento viene creato un branch dedicato nel repository_G GitHub_G per permettere la collaborazione e il versionamento. Per verbali e diari di bordo non vengono creati branch per ogni singolo documento bensì sono stati creati dei branch "verbali" e "diari-di-bordo" che raccolgono rispettivamente tutti i verbali e tutti i diari di bordo.
- **Redazione**: Il redattore_G incaricato crea la bozza del documento seguendo le linee guida stabilite nelle Norme di Progetto_G, utilizzando gli strumenti appropriati (ad esempio LaTeX_G per documenti formali).
- **Revisione**: Una volta completata la bozza, il documento viene sottoposto al verificatore_G designato che controlla la correttezza, la coerenza e la conformità agli standard di qualità_G. Il processo è asincrono e avviene tramite pull request su GitHub_G. Se ci sono modifiche vengono tracciate e discusse all'interno della pull request. La verifica_G verrà tracciata all'interno del registro delle modifiche del documento.

- **Approvazione_G e integrazione:** Dopo la revisione, il documento viene approvato dal responsabile_G e integrato nel branch principale del repository_G attraverso una pull request. Viene aggiornato il registro delle modifiche per riflettere la versione finale.
- **Modifiche:** Qualsiasi modifica successiva al documento segue lo stesso processo di redazione, revisione e approvazione_G, garantendo che tutte le versioni siano tracciate e documentate.

3.2 Gestione della configurazione

La gestione delle configurazioni è l'insieme di attività e strumenti che servono a controllare, tracciare e organizzare tutte le modifiche fatte a un qualsiasi elemento del progetto_G.

Basandosi sullo standard ISO/IEC 12207:1995, la gestione della configurazione deve occuparsi di monitorare le modifiche e le versioni degli elementi del sistema, tenere traccia del loro stato e delle richieste di cambiamento, assicurare che gli elementi siano completi, coerenti e corretti, e controllare come vengono archiviati, spostati e consegnati.

3.2.1 Strumenti utilizzati

Per ognuna delle attività previste, BugBusters utilizza:

- **Git:** come sistema di controllo di versione distribuito per tracciare le modifiche al codice sorgente e ai documenti.
- **GitHub_G:** servizio di hosting per repository_G Git utilizzato per il versionamento di codice e documentazione. Fornisce strumenti per la collaborazione (branching, pull request e code review), per il tracciamento delle attività (issues, project board) e per l'integrazione continua (Actions). Su GitHub_G vengono centralizzate le risorse di progetto_G e registrate tutte le modifiche, garantendo controllo degli accessi e tracciabilità_G.

3.2.2 Controllo della configurazione

Il controllo della configurazione consiste nell'amministrare le richieste di modifica che vengono poi approvate o meno.

Per tracciare le modifiche da approvare BugBusters utilizza le **issue_G** la **board** e le **pull request** predisposte da GitHub_G nel modo seguente:

- **Issue_G:** ogni modifica da apportare viene documentata mediante l'apertura di una issue_G relativa, quest'ultima viene assegnata a un componente che la prenderà in carico e procederà con le modifiche al relativo documento o codice.

Ogni issue_G è identificata da un codice univoco dato dalla fase del progetto_G a cui essa è relativa, e da un numero incrementale (es: RTB1, RTB2, RTB3 ecc...).

Per velocizzare il processo di gestione delle issue_G, BugBusters utilizza una GitHub action che, se in qualsiasi commit viene scritto: *chiudi <NOME ISSUE>*, la issue_G corrispondente verrà immediatamente chiusa, qualsiasi sia il branch in cui si trova.

- **Board:** Serve per definire lo stato in cui si trova una issue_G, ovvero se: è ancora da iniziare, è in sviluppo o è terminata. Il team ha scelto il modello Kanban per la gestione delle issue_G.
- **Pull Request:** serve per richiedere la verifica_G o approvazione_G di una modifica prima di fonderla con un ramo della repository_G. Il team BugBusters, oltre che per un corretto workflow_G di verifica_G, usa le pull request come unico modo di modifica in produzione, in quanto si vuole che il branch main sia modificato solo a seguito di una pull request con 2 review positive.

3.2.3 Registrazione stato di configurazione

Per poter tenere traccia dei cambiamenti di ogni documento e codice, è previsto il sistema di versionamento seguente:

X.Y.Z

Dove:

- **X**: subisce un incremento solo quando il file viene approvato;
- **Y**: subisce un incremento solo quando il file viene verificato;
- **Z**: subisce un incremento quando viene fatto un qualsiasi tipo di modifica al file;

3.3 Qualifica

3.3.1 Verifica_g

Il processo di Verifica_g ha come obiettivo principale quello di verificare che quanto prodotto_g sia a regola d'arte, ovvero conforme con i requisiti_g richiesti.

L'obiettivo della verifica_g è poter rispondere alla domanda «Did I build the system right?» gli esiti di tale processo sono riportati nel documento di Piano di Qualifica_g nella sezione dedicata ai test_g.

Notare che al momento della condivisione di questo documento al pubblico, il team BugBusters avrà solamente raggiunto l'RTB_g, per questo motivo lo stato dei test_gsara "Non Implementato". Al raggiungimento della PB_g, questa sezione verrà aggiornata con i risultati dei test_geseguiti.

3.3.1.1 Attività di verifica_g

La verifica_gagisce su singoli segmenti di sviluppo, accertando che l'esecuzione di essi non abbia introdotto errori. In questa prospettiva, un'attività di verifica_gè il controllo che il prodotto_gottenuto al termine di una fase sia congruente con il semilavorato avuto come punto di partenza di quella fase.

Il team BugBusters si è principalmente occupato della verifica_grelativa alla documentazione, controllando la correttezza grammaticale, sintattica e la correttezza del contenuto di ogni documento.

Relativamente alle verifiche sul codice, sarà un argomento che verrà trattato più in dettaglio dopo il raggiungimento della *Requirements and Technology Baseline_g*.

In ogni caso tutte le informazioni relative alla verifica_g, verranno riportate nel Piano di Qualifica_g (INSERIRE LINK FUTURO E SEZIONE CORRISPONENTE). Il processo di verifica_g verrà realizzata in due modi: tramite Analisi Statica e Analisi Dinamica.

3.3.1.2 Analisi Statica

L'analisi statica non richiede l'esecuzione dell'oggetto di cui si fa la verifica_g, ciò permette di applicarla già prima della fine della codifica.

Può essere eseguita mediante **metodi formali** (ad esempio prove matematiche) o mediante **metodi di lettura**. Tra i **metodi di lettura** ne troviamo due significativi:

- **Walkthrough**: si esegue un esame privo di assunzioni o presupposti, non si sa esattamente dove è più probabile che vi siano difetti, dunque si guarda ovunque. Si percorre il codice

simulandone possibili esecuzioni e si studia ogni parte di documento come farebbe un compilatore, verificando che le regole siano soddisfatte. È un metodo di verifica_G costoso e non automatizzabile;

- **Inspection:** si esegue un esame focalizzato su presupposti, si sa già dove cercare, i verificatori dunque rilevano la presenza di difetti eseguendo una lettura mirata dell'oggetto di verifica_G. Non è esaustiva come il Walkthrough, ma permette di creare una lista di controllo apposita per ogni oggetto di verifica_G, così da individuare potenziali problemi;

Ogni passo documenta attività svolte e risultanze. Gli errori identificati vengono discussi tra verificatore_G e autore mentre le modifiche vengono apportate generalmente solo dall'autore.

3.3.1.3 Analisi Dinamica

L'analisi dinamica è chiamata così perché per essere eseguita necessita l'esecuzione dell'oggetto da verificare.

Serve per verificare se sono presenti comportamenti inattesi, e dunque rimuovere o modificare le parti che ne causano il fault.

Per raggiungere tale obiettivo, si usufruisce di Test_G, che devono essere ripetibili e automatizzabili.

- **Ripetibili:** poiché se si presenta un failure nel codice, e vengono corretti i fault, posso eseguire lo stesso test_G nelle stesse condizioni per verificare l'effettiva correzione del failure. Perché un test_G sia ripetibile, l'ambiente di esecuzione (e quindi lo stato iniziale) deve essere sempre lo stesso;
- **Automatizzabili:** poiché i test_G possono essere centinaia, è necessario automatizzarli mediante driver (che serve per pilotare il test_G), stub (che simula i moduli necessari al test_G ma che non ne sono oggetto) e logger (che registra ciò che avviene durante l'esecuzione).

Le principali tipologie di Test_G sono:

- **Test_G di Unità_G;**
- **Test_G di Regressione;**
- **Test_G di Integrazione_G;**
- **Test_G di Sistema_G;**

I test_G eseguiti da BugBusters reperibili nella sezione 3 Metodi di Testing del Piano di Qualifica_G (INSERIRE LINK Piano di Qualifica_GE IN CASO AGGIORNARE NOME) sono descritti nella seguente maniera:

- **Codice:** un codice univoco che garantisce la traccibilità del test_G (quindi la sua identificazione univoca);
- **Descrizione:** una breve descrizione dell'obiettivo del test_G;
- **Requisito_G:** codice univoco del requisito_G a cui il test_G fa riferimento. Tale codice è reperibile nell'Analisi dei Requisiti_G nella sezione 3.1 Requisiti_G funzionali_G (INSERIRE LINK Analisi dei Requisiti_GE IN CASO AGGIORNARE NOME);
- **Stato:** lo stato del test_G, che può essere:

- **Non Implementato:** il test_g non è ancora stato implementato;
- **Superato:** il test_g è stato implementato ed eseguito;
- **Non Superato:** il test_g è stato implementato ed eseguito, ma non ha raggiunto i criteri di accettazione;
- **Da Eseguire:** il test_g è stato implementato, ma non è ancora stato eseguito.

3.3.1.4 Test_g di Unità

I test_g di unità_g hanno lo scopo di verificare le singole unità di un software, definite durante la progettazione di dettaglio. Per unità si intende la più piccola quantità di Software che sia utilmente sottoponibile a verifica_g come oggetto singolo.

I test_g di unità possono essere funzionali (black-box), cioè basati sulle specifiche dell'unità. In questo caso, vengono verificati gli input e output dati dal sistema, ma non viene verificata la logica che fornisce tali risultati.

Tuttavia, i test_g funzionali da soli non sono sufficienti per verificare la correttezza della logica interna del modulo_g. Per questo motivo vengono affiancati dai test_g strutturali (white-box), che analizzano la logica interna dell'unità cercando di percorrere tutti i cammini di esecuzione al suo interno, raggiungendo una copertura massima.

L'esecuzione di questi test_g può essere facilitata dall'uso del debugger.

Il testing di unità si considera completo quando tutte le unità del sistema sono state verificate.

3.3.1.5 Test_g di Regressione

I Test_g di regressione sono necessari affinchè delle modifiche effettuate per aggiunta, correzione o rimozione non pregiudichino le funzionalità già verificate.

Per far ciò il test_g di regressione comprende tutti i test_g necessari ad accertare che la modifica di una parte $P \in S$, non causi errori in P o in alcuna altra parte di S esterna a P .

3.3.1.6 Test_g di Integrazione

I Test_g di integrazione_g verificano il corretto funzionamento di più unità software combinate tra loro, controllando che le loro interazioni avvengano come previsto.

L'obiettivo è assicurarsi che le unità già testate singolarmente comunichino e collaborino correttamente tra loro.

Ci sono diverse strategie per implementare tali test_g:

- **Bottom-up:** si parte dalle componenti più basse (con meno dipendenze) e si integrano progressivamente verso l'alto, usando driver per simulare moduli superiori mancanti;
- **Top-down:** si dalle componenti di alto livello (con più dipendenze) e si scende verso quelli inferiori, usando stub per simulare moduli non ancora sviluppati;

3.3.1.7 Test_g di Sistema

I Test_g di sistema_g verificano il comportamento dell'intero software nel suo complesso, controllando che il sistema soddisfi i requisiti_g funzionali_g e non funzionali_g specificati.

3.3.2 Validazione_g

La **validazione_g** è un processo per confermare in modo definitivo che le caratteristiche del software siano conformi ai bisogni dell'utente finale e all'uso previsto, risponde alla domanda: «Did i build the right system?»

3.3.2.1 Processo di Validazione_g

BugBusters ha raccolto tutte le richieste di **Eggon** e le ha raccolte nell'Analisi dei Requisiti_g (LINK DA INSERIRE QUANDO Analisi dei Requisiti_gè PRONTA), in modo tale da poter tracciare correttamente i requisiti_g da poter eseguire i test_gdi accettazione per controllare che quanto sviluppato corrisponda alle richieste di **Eggon**.

I risultati della validazione_g verranno riportati nel documento di Piano di Qualifica_gnella sezione Test_gdi accettazione (LINK DA INSERIRE QUANDO Piano di Qualifica_gè PRONTO).

Si fa notare che al momento della condivisione di questo documento al pubblico, il team Bug-Busters avrà solamente raggiunto l'RTB_g, per questo motivo lo stato dei test_gsara "In Attesa". Al raggiungimento della PB_g, questa sezione verrà aggiornata con i risultati dei test_g. (MODIFICARE EVENTUALMENTE VALORE DI STATO/ESITO)

4 Processi Organizzativi

4.1 Descrizione e Scopo

Secondo lo standard ISO/IEC 12207:1997, il processo di gestione include tutte le attività necessarie a portare a termine un progetto_gsoftware garantendo il conseguimento degli obiettivi prefissati nel rispetto dei requisiti_g, dei tempi e dei costi.

Per raggiungere questi risultati il processo si basa su:

- una pianificazione accurata delle attività, delle milestone_ge delle risorse;
- il monitoraggio continuo dell'avanzamento mediante indicatori e report periodici;
- la gestione dei rischi_ge delle modifiche attraverso procedure di controllo e di escalation;
- l'adozione tempestiva di azioni correttive ogni volta che gli scostamenti rispetto al piano lo richiedono;
- la definizione chiara di ruoli, responsabilità e canali di comunicazione fra gli stakeholder_g.

In sintesi, il processo di gestione assicura che il progetto_gsia eseguito in modo controllato e tracciabile, consentendo decisioni informate e interventi rapidi per mantenere il progetto_gallineato ai vincoli di qualità_g, tempi e costi.

4.2 Attività

4.3 Gestione degli Sprint_g

Ogni due settimane viene pianificato uno sprint_gin cui vengono assegnate le attività da svolgere ai vari membri del team. Viene fatto un meeting di pianificazione in cui si scelgono le attività da svolgere e si assegnano ai vari membri. Al termine dello sprint_gviene fatta una review in cui si mostrano i risultati ottenuti e si discutono le difficoltà incontrate. Viene anche fatto un

meeting di retrospettiva_G in cui si discute su cosa è andato bene e cosa può essere migliorato per il prossimo sprint_G. Durante lo sprint_G vengono definiti i vari ruoli che i membri del team devono svolgere.

4.4 Ruoli

4.4.1 Responsabile_G

Il responsabile_G ha il compito di coordinare il team, pianificare le attività, gestire le risorse e comunicare con la proponente_G e i docenti. In particolare dovrà assegnare i compiti, redigere i documenti:

- Verbali Esterni ed Interni, con assegnazione dei compiti previsti nelle riunioni;
- Diario di Bordo
- Avanzamento Piano di Progetto_G
- redigere il documento di Piano di Qualifica_G;

4.4.2 Amministratore_G

È la figura incaricata di sviluppare, mantenere e migliorare gli strumenti, le risorse e i processi che garantiscono il regolare avanzamento del progetto_G. Si occupa di supervisionare la configurazione degli ambienti di lavoro e di monitorare le scadenze di carattere amministrativo, offrendo supporto al team nelle varie attività. Tra le sue responsabilità rientrano:

- predisporre e gestire gli ambienti di lavoro;
- controllare e aggiornare gli strumenti utilizzati dal gruppo per collaborare e comunicare;
- occuparsi del versionamento dei documenti;
- redigere e mantenere aggiornato il documento Norme di Progetto_G.

4.4.3 Analista_G

L'analista_G ha il compito di raccogliere, analizzare e documentare i requisiti_G del progetto_G, assicurandosi che siano chiari, completi e coerenti con le esigenze del proponente_G. Inoltre, collabora con il team per tradurre i requisiti_G in specifiche tecniche utilizzabili durante le fasi di progettazione e sviluppo. In particolare dovrà occuparsi di:

- redigere il documento di Analisi dei Requisiti_G;
- gestire le richieste di chiarimento e modifica dei requisiti_G con la proponente_G;
- collaborare con il team per garantire che i requisiti_G siano compresi e implementati correttamente.

4.4.4 Progettista_G

Il progettista_G ha il compito di definire l'architettura e la struttura del sistema, assicurandosi che soddisfi i requisiti_G funzionali_G e non funzionali_G stabiliti. Collabora con il team per tradurre i requisiti_G in soluzioni tecniche efficienti e scalabili. In particolare dovrà occuparsi di:

- Determinare le tecnologie e gli strumenti più adatti per lo sviluppo del progetto_g;
- definire l'architettura del sistema e le sue componenti principali;
- Supervisionare lo sviluppo tecnico e garantire la coerenza con le specifiche progettuali;

4.4.4.1 Programmatore_g

Il programmatore_g ha il compito di implementare le funzionalità_g del sistema secondo le specifiche tecniche fornite dal progettista_g. Collabora con il team per garantire che il codice sia di alta qualità_g, efficiente e mantenibile. In particolare dovrà occuparsi di:

- scrivere il codice sorgente seguendo le linee guida e gli standard di programmazione stabiliti, traducendo le specifiche tecniche definite dal progettista_g in soluzioni funzionanti;
- Occuparsi del testing del codice per garantire la correttezza e l'affidabilità_g delle funzionalità_g implementate;
- collaborare con il team per risolvere problemi tecnici e implementare nuove funzionalità_g.

4.4.5 Verificatore_g

Il verificatore_g ha il compito di assicurare che il prodotto_g sviluppato soddisfi i requisiti_g stabiliti e che sia di alta qualità_g. Collabora con il team per identificare e risolvere problemi, garantendo che il sistema sia affidabile, efficiente e conforme agli standard. In particolare dovrà occuparsi di:

- pianificare e condurre attività di verifica_g, come test_g funzionali, test_g di integrazione e test_g di sistema;
- revisionare la documentazione prodotta per garantire la correttezza e la completezza;
- documentare i risultati delle attività di verifica_g e segnalare eventuali difetti o non conformità riscontrate;
- identificare possibili miglioramenti o punti critici nel processo di sviluppo e proporre soluzioni per aumentarne l'efficacia_g.

4.5 Tracciamento Ore

Per garantire una corretta gestione del tempo e delle risorse, ogni membro del team BugBusters è tenuto a registrare le ore lavorate su ciascuna attività. Il tracciamento avviene tramite un foglio di calcolo condiviso su Google Sheets, dove ogni membro del team inserisce le ore dedicate alle varie attività e alle relative issue_g di progetto_g. Questo approccio centralizzato consente di monitorare l'avanzamento effettivo rispetto alla pianificazione, identificare tempestivamente discrepanze tra stima e consuntivo_g e facilitare decisioni di riallocazione delle risorse, se necessario.

Il foglio di calcolo contiene inoltre la tabella di stima iniziale, permettendo una comparazione immediata tra preventivo_g e consuntivo_g e garantendo visibilità sull'andamento complessivo del progetto_g. Ogni membro del team è responsabile_g dell'accuratezza e della propria registrazione.

4.6 Tracciamento Azioni

Le azioni sono tracciate nei verbali delle riunioni interne ed esterne. Dopo ogni riunione il responsabile_g redige il verbale_g in cui vengono riportate tutte le decisioni prese e le azioni da

svolgere. Ogni azione viene assegnata a un membro del team tramite issue_Gtracking su GitHub_G, dunque nei verbali è presente il collegamento con il ticket aperto sulla piattaforma. Nel caso ci fossero problemi o dubbi riguardo l'azione assegnata, il membro del team può aprire una discussione all'interno della issue_Gper chiedere chiarimenti o segnalare difficoltà. Come riportato dalla sezione 3.2, il team utilizza automazioni per rendere più fluido il processo di tracciamento delle azioni, garantendo maggior tracciamento possibile.

4.7 Comunicazione

4.7.1 Comunicazione Interna

Le comunicazioni interne al team BugBusters avvengono principalmente tramite:

- **WhatsApp**: utilizzato per comunicazioni rapide, discussioni tecniche e coordinamento quotidiano tra i membri del team.
- **Discussioni GitHub_G**: utilizzate per comunicazioni formali relative a issue_G, pull request e documentazione, garantendo tracciabilità_Ge storicizzazione delle decisioni. Utili in quanto permettono di comunicare solo con i membri interessati alla discussione.

In caso di problemi tecnici o necessità di confronto più approfondito, il team può organizzare riunioni virtuali tramite piattaforme Discord_G. Per ulteriori dettagli sulle riunioni, si veda la sezione 4.8.

4.7.2 Comunicazione Esterna

Le comunicazioni esterne con la proponente_Ge i docenti avvengono principalmente tramite:

- **Email**: utilizzata per comunicazioni formali, invio di documenti e aggiornamenti sullo stato del progetto_G, usando la mail bugbusters.unipd@gmail.com.
- **Telegram**: utilizzato per comunicazioni rapide e coordinamento con la proponente_G.

Analogamente per quanto riguarda le comunicazioni interne, in caso di necessità di confronto più approfondito, il team può organizzare riunioni virtuali tramite piattaforma Google Meet_G. Per ulteriori dettagli sulle riunioni, si veda la sezione 4.8.

4.8 Riunioni

4.8.1 Riunioni Interne

Le riunioni interne al team BugBusters sono pianificate regolarmente per garantire un coordinamento efficace e un avanzamento costante del progetto_G.

E' previsto un incontro settimanale, solitamente il lunedì, in cui si discute lo stato di avanzamento del progetto_G, si pianificano le attività della settimana e si affrontano eventuali problematiche riscontrate.

Inoltre, vengono organizzate riunioni ad hoc in caso di necessità, ad esempio per discutere questioni specifiche o per risolvere problemi urgenti.

Le riunioni si svolgono principalmente tramite piattaforma Discord_G, ma possono essere organizzate anche in presenza se necessario.

Al termine di ogni riunione, viene redatto un verbale_Gche riassume le decisioni prese e le azioni da intraprendere, garantendo tracciabilità_Ge responsabilità all'interno del team.

4.8.2 Riunioni Esterne

Le riunioni esterne con la proponente_g sono pianificate per garantire un allineamento continuo sugli obiettivi del progetto_g e per ricevere feedback preziosi.

Queste riunioni si svolgono generalmente ogni due settimane, il mercoledì alle ore 15:00, ma possono essere organizzate con maggiore frequenza in caso di necessità.

Questi incontri avvengono tramite piattaforma Google Meet_g. Al termine di ogni riunione, viene redatto un verbale_g che riassume le decisioni prese e le azioni da intraprendere, il quale verrà condiviso con la proponente_g, se in linea con quanto discusso, firmato.

5 Qualità_g del Software

Per qualità_g si intende l'insieme delle caratteristiche di un'entità che ne determinano la capacità di soddisfare esigenze sia espresse che implicite. La qualità_g si misura su due aspetti principali:

1. la **qualità_g del prodotto_g**;
2. la **qualità_g del processo**.

La qualità_g del prodotto_g finale è una conseguenza diretta della qualità_g del processo utilizzato per crearlo. Perché il software risulti di qualità_g è necessario che venga pianificato, e non solo controllato, un buon processo di valutazione: questo permette di lavorare seguendo una modalità proattiva (valutazione continua) piuttosto che reattiva (correzione di bug).

Diventa quindi fondamentale seguire uno standard di riferimento per poter definire, misurare e valutare la qualità_g del software in modo oggettivo e coerente.

5.1 Standard di riferimento e modelli

Storicamente definite dagli standard ISO/IEC 9126 e ISO/IEC 14598, queste sono oggi unificate e aggiornate nella serie di norme ISO/IEC 25010, nota anche come SQuaRE (Systems and software Quality Requirements and Evaluation). Questo standard internazionale organizza la qualità_g in 2 diverse categorie:

- Qualità_g del Prodotto_g, che analizza le seguenti proprietà intrinseche del software:
 1. **Adeguatezza funzionale**;
 2. **Efficienza_g prestazionale**;
 3. **Compatibilità**;
 4. **Usabilità**;
 5. **Affidabilità_g**;
 6. **Sicurezza_g**;
 7. **Manutenibilità_g**;
 8. **Portabilità**;
- Qualità_g in Uso, che valuta le seguenti caratteristiche dal punto di vista dell'utente finale:
 1. **Efficacia_g**;
 2. **Efficienza_g**;
 3. **Soddisfazione**;

4. **Libertà da rischi_G;**
5. **Contesto di copertura.**

Noi ci concentriamo sulla qualità_G del prodotto_G.

Per quanto riguarda, invece, la qualità_G del processo, lo standard di riferimento è l'**ISO/IEC 9001** e il **modello CMMI (Capability Maturity Model Integration)**. Questi standard forniscono linee guida per la gestione della qualità_G nei processi di sviluppo software, includendo aspetti come la pianificazione, il controllo e il miglioramento continuo dei processi stessi.

5.2 Processo di Valutazione della Qualità_G

Il processo di valutazione della qualità_G del software si articola in quattro fasi principali:

1. **Fase preventiva di definizione di metriche, interpretazione delle misure e criteri di accettazione:** sia nel contesto del prodotto_G che del processo, vengono identificate le metriche e come queste devono essere interpretate per valutare la qualità_G e raggiungere l'obiettivo di accettazione.
2. **Misurazione:** la raccolta dei dati necessari per calcolare le metriche selezionate. Questa fase dovrebbe essere automatizzata attraverso strumenti di analisi statica e dinamica del codice, test_G automatici e monitoraggio delle prestazioni.
3. **Valutazione:** l'analisi dei dati raccolti per determinare se gli obiettivi di qualità_G sono stati raggiunti. Questa fase può includere la revisione dei risultati da parte di un team di verifica_G e validazione_G, nonché la documentazione dei risultati e delle eventuali azioni correttive da intraprendere.
4. **Accettazione:** la decisione finale su se il software soddisfa i requisiti_G di qualità_G stabiliti. Se le metriche sono soddisfatte, il software può essere considerato pronto.

Questo intero processo è documentato all'interno del [Piano di Qualifica_G](#).

5.3 Principi della Qualità_G del processo

Secondo lo standard ISO/IEC 9001, la qualità_G del processo di sviluppo software si basa su diversi principi fondamentali:

- **Responsabilità della direzione:** il responsabile_G deve definire una politica per la qualità_G, stabilire obiettivi e garantire che tutti i membri del team siano consapevoli delle loro responsabilità.
- **Gestione delle risorse:** è essenziale assicurarsi che il team abbia gli strumenti giusti (es. IDE, versionamento con git) e che i membri abbiano la formazione necessaria.
- **Realizzazione del prodotto_G:** il processo di sviluppo richiede che le fasi di documentazione dei requisiti_G, progettazione e sviluppo siano sottocontrollo e tracciabili.
- **Misurazione, analisi e miglioramento:** è fondamentale monitorare continuamente il processo di sviluppo per identificare aree di miglioramento e implementare azioni correttive quando necessario.

Ciò viene fatto in questo progetto_G attraverso una chiara definizione degli strumenti e delle metodologie di lavoro (specificate in questo documento), un monitoraggio continuo dell'avanzamento del progetto_G (tramite issue_G tracking e riunioni periodiche documentate), un'attenta analisi sulle

richieste della proponente_g (specificate nel documento di [Analisi dei Requisiti_g](#)) e una valutazione regolare della qualità_g del prodotto_g con un'analisi sul miglioramento (tramite il processo di verifica_g e validazione_g descritto nel [Piano di Qualifica_g](#)).

5.4 Metrie per la qualità_g

Come descritto nella sezione precedente, la qualità_g del software viene misurata attraverso l'uso di metriche specifiche. Queste metriche forniscono dati quantitativi che aiutano a valutare diversi aspetti della qualità_g del prodotto_g e del processo di sviluppo.

Le metriche selezionate per questo progetto_g vengono definite qui e misurate durante il processo di valutazione della qualità_g descritto nel [Piano di Qualifica_g](#).

Le metriche sono suddivise in due categorie principali:

- Metriche per la qualità_g del prodotto_g:** queste metriche valutano aspetti come la funzionalità_g, l'affidabilità_g, l'efficienza_g, l'usabilità, la manutenibilità_g e la portabilità del software.
- Metriche per la qualità_g del processo:** queste metriche monitorano l'efficacia_g del processo di sviluppo, inclusi tempi di consegna, tassi di difetti, copertura dei test_g e conformità agli standard di qualità_g.

5.4.1 Qualità_g del Processo

5.4.1.1 Processi primari

Fornitura

MPC01 Earned Value (EV)_g: misura il valore del lavoro effettivamente completato rispetto al costo preventivato.

$$EV_g = BAC \times \text{Percentuale di completamento}$$

dove *BAC* (Budget at Completion) è il budget totale previsto per il progetto_g.

MPC02 Planned Value (PV)_g: misura il valore del lavoro pianificato fino a una certa data.

$$PV_g = BAC \times \text{Percentuale di pianificazione}$$

Interpretazione: Risponde alla domanda: «Quanto lavoro avremmo dovuto completare fino a questa data?».

Valore ideale: per ogni sprint_g il valore deve essere $\leq EAC_g$

MPC03 Actual Cost (AC)_g: misura il costo effettivo sostenuto per il lavoro completato fino a una certa data (che coincide con quella di fine sprint_g).

MPC04 Cost Performance Index (CPI)_g: indica l'efficienza_g dei costi del progetto_g.

$$CPI_g = \frac{EV_g}{AC_g}$$

Interpretazione: Risponde alla domanda: «Quanto lavoro è stato completato rispetto a quanto è stato speso?».

Se $CPI_g > 1$, il progetto_g è sotto budget (positivo).

MPC05 Schedule Performance Index (SPI)_g: indica l'efficienza_g del tempo del progetto_g.

$$SPI_g = \frac{EV_g}{PV_g}$$

Interpretazione: Risponde alla domanda: «Quanto lavoro è stato effettivamente completato rispetto a quanto pianificato?».

Se $SPI_g > 1$, il progetto_g è in anticipo rispetto alla pianificazione (positivo).

MPC06 Estimated at Completion (EAC_g): stima il costo totale del progetto_g alla sua conclusione, basandosi sulle prestazioni attuali.

$$EAC_g = \frac{BAC}{CPI_g}$$

Interpretazione: Risponde alla domanda: «Quanto costerà il progetto_g se le attuali prestazioni di costo continuano?».

Valore ideale = BAC

MPC07 Estimate to Complete (ETC)_g: stima il costo necessario per completare il lavoro rimanente del progetto_g.

$$ETC_g = EAC_g - AC_g$$

Interpretazione: Risponde alla domanda: «Quanto costerà completare il progetto_g?».
Valore ideale $\leq EAC_g$

MPC08 Time Estimate At Completion (TEAC): stima il tempo totale necessario per completare il progetto_g.

$$TEAC = \frac{\text{Durata Pianificata}}{SPI_g}$$

dove SPI_g (Schedule Performance Index_g) è $\frac{EV_g}{PV_g}$. *Interpretazione:* Risponde alla domanda: «Quanto tempo ci vorrà per completare il progetto_g se le attuali prestazioni di tempo continuano?».

Valore ideale \leq Durata pianificata.

Sviluppo

MPC09 Requirements Stability Index (RSI)_g: misura la stabilità dei requisiti_g durante il ciclo di vita del progetto_g.

$$RSI_g = \left(1 - \frac{N_{agg} + N_{mod} + N_{canc}}{N_{iniz}} \right) \times 100$$

dove:

- N_{agg} è il numero di nuovi requisiti_g emersi dopo l'inizio del periodo;
- N_{mod} è il numero di requisiti_g modificati;
- N_{canc} è il numero di requisiti_g rimossi;
- N_{iniz} è il numero di requisiti_g definiti all'inizio del periodo.

Interpretazione: un valore più alto indica una maggiore stabilità dei requisiti_g.

5.4.1.2 Processi di supporto

Documentazione

MPC10 Indice di Gulpease_g: misura la leggibilità di un testo in lingua italiana, utile per valutare la documentazione tecnica.

$$\text{Indice di Gulpease}_g = 89 + \frac{300 \times \text{Numero di frasi} - 10 \times \text{Numero di lettere}}{\text{Numero di parole}}$$

Interpretazione: Un valore più alto indica una maggiore leggibilità del testo

Verifica_g

MPC11 Code Coverage_g: misura la percentuale di codice sorgente coperto dai test_g.

$$\text{Code Coverage}_g = \frac{\text{Linee di codice testate}}{\text{Linee di codice totali}} \times 100\%$$

Interpretazione: Un valore più alto indica una maggiore copertura dei test_g, riducendo il rischio_g di bug non rilevati.

MPC12 Test_g Success Rate: misura la percentuale di test_g superati con successo.

$$\text{Test}_g \text{ Success Rate} = \frac{\text{Numero di test}_g \text{ superati}}{\text{Numero totale di test}_g \text{ eseguiti}} \times 100\%$$

Interpretazione: Un valore più alto indica una maggiore affidabilità_g del software.

Gestione della qualità_g

MPC13 Quality metrics satisfied: misura la percentuale di metriche di qualità_g soddisfatte rispetto a quelle definite nel Piano di Qualifica_g.

$$\text{Quality metrics satisfied} = \frac{\text{Numero di metriche soddisfatte}}{\text{Numero totale di metriche}} \times 100\%$$

Interpretazione: Un valore più alto indica una maggiore conformità agli standard di qualità_g stabiliti.

5.4.1.3 Processi organizzativi

Gestione dei processi

MPC14 Time efficiency: misura l'efficienza_g nella gestione del tempo rispetto alla pianificazione.

$$\text{Time efficiency} = \frac{\text{Ore produttive}}{\text{Ore totali}} \times 100\%$$

Interpretazione: Un valore più alto indica una migliore gestione del tempo e rispetto delle scadenze.

5.4.2 Qualità_g del Prodotto_g

5.4.2.1 Adeguatezza funzionale

MPD01 Requisiti_g obbligatori soddisfatti.

Valore ideale: 100

MPD02 Requisiti_g desiderabili soddisfatti.

Valore ideale: 100 (*accettato 0*)

MPD03 Requisiti_g opzionali soddisfatti.

Valore ideale: 100 (*accettato 0*)

MPD04 AI_g Acceptance Rate (rating $\geq 3/5$).

Valore ideale: $\geq 80\%$ (*accettato $\geq 60\%$*)

5.4.2.2 Affidabilità_g

MPD05 Branch Coverage_g.

Valore di accettazione: $\geq 70\%$

Valore ideale: $\geq 85\%$

MPD06 Defect Density.

Valore di accettazione: ≤ 3 difetti / KLOC

Valore ideale: ≤ 1 difetto / KLOC

5.4.2.3 Efficienza_g

MPD07 UI Response Time (Interfaccia_g).

Valore di accettazione: ≤ 2 sec

Valore ideale: ≤ 0.5 sec

MPD08 Code Response Time - Ai Generativo testo.

Valore di accettazione: ≤ 5 sec

Valore ideale: ≤ 3 sec

MPD09 Code Response Time - Ai Generativo immagini.

Valore di accettazione: ≤ 10 sec

Valore ideale: ≤ 5 sec

MPD10 Code Response Time - Ai Co-Pilot_g.

Valore di accettazione: ≤ 10 sec

Valore ideale: ≤ 5 sec

5.4.2.4 Usabilità

MPD11 Click Count (Funzioni principali). Misura il numero di click necessari per accedere alle funzioni principali.

Valore di accettazione: ≤ 5 click

Valore ideale: ≤ 3 click

MPD12 User Error Rate (Errori di validazione_g).

Valore di accettazione: $\leq 10\%$

Valore ideale: $\leq 5\%$

5.4.2.5 Manutenibilità_g

MPD13 Blocker Code Smells. Misura la presenza di "code smells" di tipo blocker, ovvero problemi nel codice che possono causare gravi malfunzionamenti o difficoltà di manutenzione.

Valore di accettazione: 0

Valore ideale: 0

MPD14 Cyclomatic Complexity_g(per metodo). Misura la complessità ciclometrica_g di un metodo, ovvero il numero di percorsi indipendenti nel codice.

Valore di accettazione: ≤ 15

Valore ideale: ≤ 10

MPD15 Comment Intensity.

Valore di accettazione: $\geq 10\%$

Valore ideale: $\geq 20\%$

5.4.2.6 Portabilità

MPD16 Supported Browsers (Test_g passati).

Valore di accettazione: 100% (Desktop)

Valore ideale: 100% (All devices)