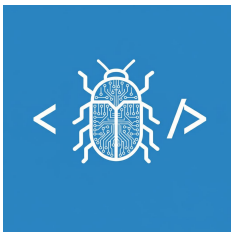

BugBusters

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

Versione 0.0.8

Destinatari	BugBusters, Prof. Tullio Vardanega, Prof. Riccardo Cardin
Data ultima modifica	17/11/2025

Abstract

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

Registro delle modifiche

Versione	Data	Descrizione	Redatto	Verificato	Approvato
0.0.10	22/12/2025	Aggiunto Piano di Progetto e di Qualifica	Marco Piro	-	-
0.0.9	18/12/2025	Eliminato i Processi di acquisizione ed aggiunto le lettere di Candidatura 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

1.1 Scopo del documento

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

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 Nexum. In particolare identifica tre tipologie di processi:

- Processi primari$_G$: processi direttamente coinvolti nella creazione del prodotto software
- Processi di supporto$_G$: processi che supportano i processi primari$_G$
- Processi organizzativi$_G$: 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 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 previsti. Il documento è redatto in maniera incrementale: verrà aggiornato e ampliato progressivamente durante lo sviluppo del progetto per riflettere decisioni, modifiche e miglioramenti adottati dal team.

1.2 Scopo del prodotto

Nexum è una piattaforma con un modulo 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 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 mira a estendere la piattaforma Nexum con un AI assistant generativo per la scrittura di comunicazioni e con un AI Co-Pilot$_G$ per i Cdl. 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.
Il nostro obiettivo é realizzare questo progetto entro il 21 marzo 2026 con un budget di 12.790 euro.

1.3 Glossario

Il glossario raccoglie e definisce i termini, gli acronimi e le abbreviazioni impiegati nel documento e nel progetto 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 disponibile al seguente link. Per maggiore usabilità e facilitá di consultazione, il glossario è accessibile anche aprendo dal nostro sito web i vari documenti con il viewer pdf da noi sviluppato.

1.3.1 Riferimenti normativi

- **Capitolato$_G$ d'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 software di alta qualità 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 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 in modo controllato, condiviso e verificabile. In particolare, tale processo ha l'obiettivo di chiarire i requisiti richiesti dal proponente, 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 Attivitá

La fornitura prevede varie attitá, in particolare:

- **Analisi del capitolato proposto**: raccolta di informazioni, requisiti, tecnologie e vincoli presenti nel capitolato d'appalto. In questa fase si pongono domande al proponente per chiarire eventuali dubbi o ambiguità.
- **Stima delle risorse**: definizione delle tempistiche, dei costi totali e delle risorse necessarie per la realizzazione del progetto.
- **Analisi dei requisiti e contrattazione con il proponente**: identificazione e documentazione dei requisiti funzionali e non funzionali del software, analisi delle aspettative del proponente e negoziazione di eventuali modifiche o aggiunte. Definizione del Minimum Viable Product (MVP).
- **Pianificazione del progetto**: suddivisione del progetto in fasi, definizione delle milestone (RTB e PB), 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).
- **Documentazione**: redazione di tutti i documenti necessari per formalizzare la fornitura, come il Piano di Progetto, il Analisi dei requisiti, Norme di Progetto 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 del proponente.
- **Comunicazione con il proponente**: stabilire canali di comunicazione efficaci per garantire un flusso informativo continuo e trasparente. Prevedere incontri regolari per aggiornamenti sullo stato del progetto, discussione di eventuali problemi e raccolta di feedback.

- **Revisione e approvazione**: sottoporre tutta la documentazione prodotta alla revisione e approvazione del proponente, assicurando l'allineamento sugli obiettivi e le aspettative prima di procedere con le fasi successive del progetto.
- **Consegna e chiusura della fase di fornitura**: consegna di quanto prodotto durante la fase di fornitura al proponente, 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 usofruisce di vari strumenti:

- **GitHub**: 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 dal proponente e per l'implementazioe delle nuove feature.
- **Discord**: Per le riunioni di team.
- **WhatsApp**: Per comunicazioni rapide e aggiornamenti interni al gruppo.
- **Calendar Google**: Per la pianificazione delle riunioni e delle scadenze.
- **Telegram e Gmail**: Per la comunicazione con il proponente e per inviare documenti ufficiali.

2.1.4 Comunicazione e Organizzazione

Le comunicazioni con il proponente avvengono inizialmente con cadenza settimanale il mercoledì alle 15:00 e, successivamente, a progetto avviato, con cadenza bisettimanale. I verbali relativi agli incontri con il proponente vengono condivisi e archiviati nel repository della documentazione per garantire trasparenza e reperibilità. Tutte le decisioni rilevanti emerse negli incontri con il proponente devono essere formalmente riportate e documentate su GitHub e nella documentazione ufficiale. Eventuali problemi critici segnalati dal proponente vengono comunicati immediatamente sul canale concordato, in modo da garantirne la tracciabilità e 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:

2.1.5.1 Glossario

Per facilitare la lettura e la comprensione dei documenti di progetto, viene redatto un glossario che raccoglie e definisce i termini tecnici, gli acronimi e le abbreviazioni utilizzati. Questo strumento è fondamentale per garantire una comunicazione chiara e univoca tra tutti i membri del team, con il proponente, docenti e con i lettori esterni. Per una consultazione rapida durante la visualizzazione dei documenti, il glossario è accessibile anche tramite il viewer PDF sviluppato dal team.

Campo	Dettaglio
Redattore	Amministratore
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 previso e il costo orario di ciascun membro del team BugBusters per la realizzazione del progetto Nexum e la data di consegna prevista, impegnandosi a rispettare tali condizioni durante l'intero ciclo di vita del progetto.

Campo	Dettaglio
Redattore	Responsabile
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 di Ingegneria del Software, analizzandone punti di forza, debolezze e opportunità in modo da scegliere il capitolato più adatto alle competenze e agli interessi del team BugBusters. Per ogni capitolato 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	Responsabile
Destinatari	BugBusters, Prof. Vardanega, Prof. Cardin
Uso	Esterno

2.1.5.4 Analisi dei Requisiti

Il documento di Analisi dei Requisiti descrive in dettaglio le funzionalità, i vincoli e le proprietà di qualità che il sistema Nexum dovrà soddisfare. Questo capitolo fornisce:

- la definizione del contesto, degli stakeholder e degli attori coinvolti;
- l'elenco dei requisiti funzionali e non funzionali, classificati e dotati di codifica univoca e criteri di accettazione;
- casi d'uso e scenari principali con flussi e attori associati;
- vincoli tecnici, normativi e di integrazione con sistemi esterni;
- la prioritizzazione dei requisiti e l'identificazione del Minimum Viable Product (MVP);
- la strategia di tracciabilità e gestione delle modifiche (issue tracker, versionamento e mappatura requisiti–test);
- i criteri e i metodi per la verifica e la validazione dei requisiti.

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

2.1.5.5 Norme di Progetto

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

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

2.1.5.6 Lettera di Candidatura

La Lettera di Candidatura rappresenta il documento formale con cui il gruppo BugBusters ha ufficializzato la propria candidatura per il capitolato proposto dall'azienda Eggon. Al suo interno sono sintetizzate le motivazioni che hanno determinato la scelta di questo specifico progetto, evidenziando i punti di interesse tecnologico e formativo, oltre a fornire i riferimenti ai repository contenenti l'intera documentazione di progetto.

Campo	Dettaglio
Redattore	Responsabile
Destinatari	BugBusters, Eggon, 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:

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

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

2.1.5.8 Verbali

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

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

2.1.5.9 Piano di Progetto

Il Piano di Progetto 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, l'allocazione delle risorse umane e strumentali, nonché l'analisi preventiva dei rischi con le relative strategie di mitigazione. Il documento ha inoltre la funzione fondamentale di monitorare l'andamento del progetto, riportando a ogni avanzamento il consuntivo 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	Responsabile	Responsabile
Destinatari	BugBusters, Prof. Vardanega, Prof. Cardin	BugBusters, Eggon, Prof. Vardanega, Prof. Cardin
Uso	Interno	Esterno

2.1.5.10 Piano di Qualifica

Il Piano di Qualifica documenta la strategia adottata dal gruppo BugBusters per garantire la qualità del prodotto software Nexum e dei processi correlati. Esso definisce nel dettaglio le metodologie di verifica e validazione, gli standard di qualità di riferimento e le metriche utilizzate per il monitoraggio. Il documento riporta inoltre la pianificazione e gli esiti dei test effettuati, con l'obiettivo specifico di assicurare il rispetto dei requisiti e mantenere una copertura del codice (code coverage) pari o superiore all'80

Campo	Dettaglio interni	Dettaglio esterni
Redattore	Responsabile	Responsabile
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 software Nexum, in conformità ai requisiti stabiliti durante la fase di fornitura. Questo processo include la progettazione, l'implementazione, il testing e la documentazione del software, garantendo che ogni fase sia eseguita secondo standard di qualità elevati e best practice del settore.

2.2.0.1 attività previste

DA FARE

3 Processi di Supporto

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. Questo processo mira a garantire che tutta la documentazione prodotta sia chiara, coerente, accessibile e aggiornata, facilitando la comunicazione tra i membri del team, il proponente e gli stakeholder esterni. Una documentazione ben strutturata supporta l'efficace gestione del progetto, la tracciabilità delle decisioni e la conformità agli standard di qualità previsti.

3.1.2 Strumenti Utilizzati

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

- **GitHub**: Per la gestione collaborativa della documentazione, il versionamento dei documenti e la produzione asincrona. Vengono utilizzate le funzionalità di issue 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**: 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, 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 capitolato C4. Ogni documento verrà mantenuto aggiornato e versionato sul repository del progetto.

1. **Norme di Progetto**: Il presente documento: linee guida metodologiche, standard di lavoro e convenzioni adottate dal team.
2. **Piano di Progetto**: Cronoprogramma dettagliato e allocazione delle risorse umane e materiali per le milestone e le consegne.
3. **Piano di Qualifica**: Strategia e metriche di collaudo; obiettivo minimo di qualità: copertura dei test pari o superiore al 80%.
4. **Analisi dei Requisiti**: Specifica dei requisiti funzionali e non funzionali, casi d'uso principali e criteri di accettazione.
5. **Glossario**: Elenco e definizioni dei termini tecnici e degli acronimi usati nel progetto.
6. **Lettera di Presentazione**: Documento di presentazione formale rivolto a RTB, 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 Sync Lab e il proponente.

Tutti i documenti saranno sottoposti a revisione interna e tracciati tramite GitHub (issue e pull request) per garantirne la tracciabilità 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, glossario e 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, verificatore, 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 é 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 delle versioni.

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. 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.

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

`<TIPO>_<DATA>.pdf`

Dove <TIPO> può essere:

- **VI**: Verbale Interno
- **VE**: Verbale Esterno

E <DATA> è la data della riunione nel formato `GG-MM-AAAA`.
Esempio: `VE_15-01-2026.pdf` rappresenta il verbale esterno 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.pdf, Piano di Progetto.pdf, Analisi dei Requisiti.pdf ecc.

3.1.6 Produzione

La produzione della documentazione segue un processo strutturato per garantire qualità, coerenza e tracciabilità.

- **Pianificazione**: A seconda del documento, viene pianificata la sua creazione in base alle milestone del progetto e alle esigenze del team. Secondo il tipo di documento, viene assegnato un redattore e un verificatore secondo i ruoli definiti nel way of working (vedi sezione 2.1.5).
- **Creazione del branch di lavoro**: Per ogni documento viene creato un branch dedicato nel repository GitHub per permettere la collaborazione e il versionamento. Per verbali e diari di bordo non vengono creati branch per ogni singolo documenti 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 incaricato crea la bozza del documento seguendo le linee guida stabilite nelle norme di progetto, utilizzando gli strumenti appropriati (ad esempio LaTeX per documenti formali).
- **Revisione**: Una volta completata la bozza, il documento viene sottoposto al verificatore designato che controlla la correttezza, la coerenza e la conformità agli standard di qualità. Il processo è asincrono e avviene tramite pull request su GitHub. Se ci sono modifiche vengono tracciate e discusse all'interno della pull request. La verifica verrà tracciata all'interno del registro delle modifiche del documento.
- **Approvazione e integrazione**: Dopo la revisione, il documento viene approvato dal responsabile e integrato nel branch principale del repository 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, 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**$_G$: come sistema di controllo di versione distribuito per tracciare le modifiche al codice sorgente e ai documenti.
- **GitHub**$_G$: servizio di hosting per repository 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 vengono centralizzate le risorse di progetto e registrate tutte le modifiche, garantendo controllo degli accessi e tracciabilità.

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 nel modo seguente:

- **Issue**: 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...).
Una issue$_G$ viene chiusa solo nel momento in cui è stata verificata. 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 ci 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.
- **Pull Request**: serve per richiedere la verifica$_G$ o approvazione di una modifica prima di fonderla con un ramo della repository$_G$. Il team bugbuster, oltre che per un corretto workflow di verifica, 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$_G$ seguente:

X.Y.Z

Dove:

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

3.3 Qualifica

3.3.1 Verifica

Il processo della Verifica$_G$ ha come obiettivo principale quello di verificare che quanto prodotto sia a regola d'arte, ovvero conforme con i requisiti 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 (Inserire Parte Relativa Nel Piano Di Qualifica).

3.3.1.1 Attività di verifica

Il team BugBusters si è principalmente occupato della verifica relativa 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 al raggiungimento della *Requirements and Technology Baseline*$_G$.
In ogni caso tutte le informazioni relative alla verifica, verranno riportate nel Piano di Qualifica (Inserire Link Futuro alla sezione del Piano relativa ai test). 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 dunque il codice simulandone possibili esecuzioni e si studia ogni parte di documento come farebbe un compilatore. 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;

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.
Automatizzabili mediante driver (che serve per pilotare il test), 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 di Unità**$_G$;
- **Test di Regressione**;
- **Test di Integrazione**$_G$;
- **Test di Sistema**$_G$;

I test eseguiti da BugBusters (reperibili nel INSERIRE LINK PIANO DI QUALIFICA DOVE SONO PRESENTI I TEST) sono descritti nella seguente maniera: DOBBIAMO ANCORA FARLI, SUCCESSIVAMENTE QUA INSERIRE NOMENCLATURA TEST.

3.3.1.4 Test di Unità$_G$

I **test 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 di unità$_G$ 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. 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 di Regressione

I **Test di regressione** sono necessari affinchè delle modifiche effettuate per aggiunta, correzione o rimozione non pregiudichino le funzionalità già verificate.
Per far ciò il test di regressione$_G$ 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 di Integrazione$_G$

I **Test 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 di Sistema$_G$

I **Test di sistema**$_G$ verificano il comportamento dell'intero software nel suo complesso, controllando che il sistema soddisfi i requisiti funzionali e non funzionali specificati.

3.3.2 Validazione

La **validazione** 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

BugBusters ha raccolto tutte le richieste di **Eggon** e le ha raccolte nell'Analisi dei Requisiti$_G$ (LINK DA INSERIRE QUANDO ANALISI DEI REQUISITI è PRONTA), in modo tale da poter tracciare correttamente i requisiti e da poter eseguire i test di accettazione$_G$ per controllare che quanto sviluppato corrisponda alle richieste di **Eggon**.

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 software garantendo il conseguimento degli obiettivi prefissati nel rispetto dei requisiti, dei tempi e dei costi.
Per raggiungere questi risultati il processo si basa su:

- una pianificazione accurata delle attività, delle milestone e delle risorse;
- il monitoraggio continuo dell'avanzamento mediante indicatori e report periodici;
- la gestione dei rischi e 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.

In sintesi, il processo di gestione assicura che il progetto sia eseguito in modo controllato e tracciabile, consentendo decisioni informate e interventi rapidi per mantenere il progetto allineato ai vincoli di qualità, tempi e costi.

4.2 Attività

4.3 Gestione degli Sprint

Ogni due settimane viene pianificato uno sprint in 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 viene fatta una review in cui si mostrano i risultati ottenuti e si discutono le difficoltà incontrate. Viene anche fatto un meeting di retrospettiva in cui si discute su cosa è andato bene e cosa può essere migliorato per il prossimo sprint. Durante lo sprint vengono definiti i vari ruoli che i membri del team devono svolgere.

4.4 Ruoli

4.4.1 Responsabile

Il responsabile ha il compito di coordinare il team, pianificare le attività, gestire le risorse e comunicare con il proponente 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

4.4.2 Amministratore

È la figura incaricata di sviluppare, mantenere e migliorare gli strumenti, le risorse e i processi che garantiscono il regolare avanzamento del progetto. 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.

4.4.3 Analista

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

- redigere il documento di Analisi dei Requisiti;
- redigere il documento di Piano di Qualifica;
- gestire le richieste di chiarimento e modifica dei requisiti con il proponente;
- collaborare con il team per garantire che i requisiti siano compresi e implementati correttamente.

4.4.4 Progettista

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

- Determinare le tecnologie e gli strumenti più adatti per lo sviluppo del progetto;
- 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

Il programmatore ha il compito di implementare le funzionalità del sistema secondo le specifiche tecniche fornite dal progettista. Collabora con il team per garantire che il codice sia di alta qualità, 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 in soluzioni funzionanti;
- Occuparsi del testing del codice per garantire la correttezza e l'affidabilità delle funzionalità implementate;
- collaborare con il team per risolvere problemi tecnici e implementare nuove funzionalità.

4.4.5 Verificatore

Il verificatore ha il compito di assicurare che il prodotto sviluppato soddisfi i requisiti stabiliti e che sia di alta qualità. 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, come test funzionali, test di integrazione e test di sistema;
- revisionare la documentazione prodotta per garantire la correttezza e la completezza;
- documentare i risultati delle attività di verifica 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.

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 di progetto. Questo approccio centralizzato consente di monitorare l'avanzamento effettivo rispetto alla pianificazione, identificare tempestivamente discrepanze tra stima e consuntivo 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 e consuntivo e garantendo visibilità sull'andamento complessivo del progetto. Ogni membro del team è responsabile 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 redige il verbale in cui vengono riportate tutte le decisioni prese e le azioni da svolgere. Ogni azione viene assegnata a un membro del team tramite issue tracking su GitHub, 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 per 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**: utilizzate per comunicazioni formali relative a issue, pull request e documentazione, garantendo tracciabilità e 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. Per ulteriori dettagli sulle riunioni, si veda la sezione 4.8.

4.7.2 Comunicazione Esterna

Le comunicazioni esterne con il proponente e i docenti avvengono principalmente tramite:

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

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. 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.
E' previsto un incontro settimanale, solitamente il lunedì, in cui si discute lo stato di avanzamento del progetto, 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, ma possono essere organizzate anche in presenza se necessario.
Al termine di ogni riunione, viene redatto un verbale che riassume le decisioni prese e le azioni da intraprendere, garantendo tracciabilità e responsabilità all'interno del team.

4.8.2 Riunioni Esterne

Le riunioni esterne con il proponente sono pianificate per garantire un allineamento continuo sugli obiettivi del progetto 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. Al termine di ogni riunione, viene

redatto un verbale che riassume le decisioni prese e le azioni da intraprendere, il quale verrà condiviso con il proponente e, se in linea con quanto discusso, firmato.

5 Standard di progetto

5.1 Standard ISO/IEC 9126

Lo standard ISO/IEC 9126 fornisce delle linee guida e normative, in modo da poter fornire un modello univoco e riconosciuto globalmente che misura la qualità del software.
Questo modello è definito da sei caratteristiche generali:

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

5.1.1 Funzionalità

La funzionalità verifica se sono soddisfatti i requisiti funzionali definiti con la proponente.
Nello specifico misura:

- **Appropriatezza**: definisce se il prodotto software offre le funzioni necessarie per svolgere i compiti fissati dal proponente.
- **Accuratezza**: è la capacità del prodotto software di produrre i precisi risultati richiesti.
- **Interoperabilità**: è la capacità del prodotto software di poter interagire e operare con altri sistemi.
- **Conformità**: è la capacità del prodotto software di essere conforme a standard, convenzioni e regolamentazioni che sono di rilievo per il settore nel quale deve operare.
- **Sicurezza**: è la capacità del prodotto software di tutelare le informazioni, assicurando che ciascun utente possa accedere esclusivamente ai dati per i quali possiede le necessarie autorizzazioni.

5.1.2 Affidabilità

Consiste nella capacità con cui il software svolge i suoi compiti mantenendo un certo livello di prestazioni in qualsiasi momento, anche in periodi d'uso intensi.
Viene misurato nello specifico:

- **Maturità**: è la capacità di un prodotto software di evitare che si verifichino errori, malfunzionamenti o che siano prodotti risultati errati.
- **Tolleranza agli errori**: è la capacità di mantenere livelli costanti di prestazioni anche in caso di malfunzionamenti o usi scorretti del prodotto.

- **Recuperabilità**: indica la capacità di un prodotto software di ripristinare un adeguato livello di funzionamento e recuperare le informazioni necessarie dopo un guasto o un malfunzionamento.
- **Aderenza**: indica la capacità di aderire a standard o convenzioni inerenti all'affidabilità.

5.1.3 Efficienza

L'efficienza misura la capacità di fornire appropriate prestazioni in confronto alla quantità di risorse utilizzate.
Viene misurato nello specifico:

- **Comportamento rispetto al tempo**: è la capacità di fornire adeguati tempi di risposta ed elaborazione, sotto determinate condizioni.
- **Utilizzo delle risorse**: è la capacità di utilizzare quantità e tipo di risorse in maniera adeguata.
- **Conformità**: è la capacità del prodotto ad aderire a standard relativi all'efficienza.

5.1.4 Usabilità

L'usabilità è la capacità del prodotto software di essere compreso e utilizzato dall'utente finale.
Viene misurato nello specifico:

- **Comprensibilità**: misura la facilità di comprensione dei concetti del prodotto.
- **Apprendibilità**: misura la facilità nell'apprendere l'uso delle funzionalità del prodotto.
- **Operabilità**: definisce se l'utilizzo del prodotto da parte degli utenti risulta semplice o no.
- **Attrattiva**: è la capacità del software di risultare piacevole all'utente quando viene utilizzato.
- **Conformità**: è la capacità del prodotto ad aderire a standard relativi all'usabilità.

5.1.5 Manutenibilità

La manutenibilità è la capacità del software di essere modificato, includendo correzioni, adattamenti o miglioramenti.
Viene misurato nello specifico:

- **Analizzabilità**: misura la capacità con la quale è possibile analizzare il codice per trovare un errore in esso presente.
- **Modificabilità**: misura la capacità del prodotto software di permettere l'implementazione di modifiche specifiche.
- **Stabilità**: misura la capacità del prodotto software di evitare effetti inaspettati derivanti da modifiche errate.
- **Testabilità**: definisce la capacità del prodotto software di essere facilmente testato per validare le modifiche apportate.

5.1.6 Portabilità

La portabilità definisce la capacità del software di essere trasportato da un ambiente di lavoro ad un altro.
Viene misurato nello specifico:

- **Adattabilità**: è la capacità del software di essere adattato per differenti ambienti operativi senza dover apportare modifiche.
- **Installabilità**: è la capacità del software di essere installato in uno specifico ambiente.
- **Conformità**: è la capacità del prodotto software di aderire a standard relativi alla portabilità.
- **Sostituibilità**: è la capacità del software di essere utilizzato al posto di un altro software per svolgere gli stessi compiti di quest'ultimo nello stesso ambiente.