



UNIVERSITÀ DEGLI STUDI DI PADOVA  
CORSO DI LAUREA IN INFORMATICA (L-31)

Corso di Ingegneria del Software  
Anno Accademico 2025/2026

# Norme di Progetto

**Gruppo: NightPRO**

[swe.nightpro@gmail.com](mailto:swe.nightpro@gmail.com)

Data: 2026-01-28

Versione: 1.4

## Tabella delle Versioni

Versione	Data	Autore/i	Descrizione delle Modifiche	Verificatore
1.4	2026-01-28	D. Biasuzzi	Completate sezioni Analisi dei Requisiti e Codifica; integrato Ciclo di Deming (PDCA) e metriche EVM nella sezione Ottimizzazione dei processi, Gestione dei Rischi, DdB e spiegazione progettazione UI per MVP	Francesco Zanella
1.3	2025-12-03	M. Ogniben	Chiariti compiti Responsabile, aggiunta descrizione nuova sezione verbali interni, sistematica descrizione organizzazione repository	Giovanni Ponso
1.2	2025-11-13	D. Biasuzzi	Aggiunta struttura iniziale paragrafo relativo alla fornitura (sezioni "Analisi Req". e "Codifica" da completare)	Michele Ogniben
1.1	2025-11-11	M. Ogniben	Aggiunto paragrafo relativo ai Processi Organizzativi	Davide Biasuzzi
1.0	2025-11-05	D. Biasuzzi	Modifica struttura del file e verifica	Francesco Zanella
0.1	2025-11-05	G. Ponso; D. Biasuzzi	Creazione documento + redazione processi di supporto	Francesco Zanella

## Indice

<b>Tabella delle Versioni</b>	<b>2</b>
<b>1 Informazioni Generali</b>	<b>5</b>
1.1 Componenti del Gruppo . . . . .	5
<b>2 Introduzione</b>	<b>6</b>
2.1 Scopo del documento . . . . .	6
2.2 Glossario . . . . .	6
<b>3 Processi primari</b>	<b>7</b>
3.1 Fornitura . . . . .	7
3.1.1 Descrizione . . . . .	7
3.1.2 Gestione della Comunicazione . . . . .	7
3.1.3 Strumenti . . . . .	7
3.2 Sviluppo . . . . .	7
3.2.1 Scopo . . . . .	7
3.2.2 Descrizione . . . . .	8
3.2.3 Risultato Atteso . . . . .	8
3.2.4 Analisi dei RequisitiG . . . . .	8
3.2.5 Progettazione . . . . .	10
3.2.6 Codifica (MVP) . . . . .	10
<b>4 Processi di Supporto</b>	<b>14</b>
4.1 Ambiente collaborativo e infrastruttura del progetto . . . . .	14
4.1.1 Obiettivo . . . . .	14
4.1.2 Comunicazione sincrona e asincrona . . . . .	14
4.1.3 Gestione del repository e versionamento . . . . .	14
4.1.4 Gestione delle attività . . . . .	14
4.1.5 Sistema di pubblicazione e consultazione della documentazione . . . . .	15
4.1.6 Automazione tramite GitHub Actions . . . . .	15
4.2 Documentazione . . . . .	16
4.2.1 Obiettivo . . . . .	16
4.2.2 Formati di riferimento . . . . .	16
4.2.3 Struttura dei Documenti . . . . .	17
4.2.4 Convenzioni di Scrittura e Nomenclatura . . . . .	17
4.2.5 Ciclo di Vita dei Documenti . . . . .	18
4.2.6 Verbali . . . . .	18
<b>5 Processi organizzativi</b>	<b>20</b>
5.1 Finalità . . . . .	20
5.2 Ambito di applicazione . . . . .	20
5.3 Ruoli e responsabilità . . . . .	20
5.3.1 Responsabile di Progetto . . . . .	20
5.3.2 Amministratore . . . . .	21
5.3.3 Analista . . . . .	21
5.3.4 Progettista . . . . .	22
5.3.5 Programmatore . . . . .	22
5.3.6 Verificatore . . . . .	22

5.4	Comunicazione e incontri . . . . .	23
5.4.1	Canali di comunicazione interna . . . . .	23
5.4.2	Canali di comunicazione esterna . . . . .	23
5.4.3	Organizzazione degli incontri . . . . .	23
5.4.4	Responsabilità durante gli incontri . . . . .	24
5.5	Pianificazione e tracciamento delle attività . . . . .	25
5.5.1	Approccio metodologico . . . . .	25
5.5.2	Processo di gestione dei task . . . . .	25
5.6	Sviluppo delle competenze . . . . .	25
5.6.1	Obiettivi formativi . . . . .	26
5.6.2	Modalità di apprendimento . . . . .	26
5.7	Ottimizzazione dei processi . . . . .	26
5.7.1	Ciclo di Deming (PDCA) . . . . .	26
5.7.2	Metriche di Processo e di Prodotto . . . . .	27
5.7.3	Meccanismi di implementazione . . . . .	27
5.8	Gestione dei rischi . . . . .	28
5.8.1	Classificazione dei rischi . . . . .	28
5.9	Diario di Bordo . . . . .	28
5.9.1	Finalità . . . . .	28
5.9.2	Modalità di svolgimento . . . . .	29
5.9.3	Contenuti della presentazione . . . . .	29
5.10	Progettazione dell’interfaccia utente . . . . .	29
5.10.1	Strumenti adottati . . . . .	29
5.10.2	Processo di design . . . . .	29

## 1 Informazioni Generali

### 1.1 Componenti del Gruppo

Cognome	Nome	Matricola
Biasuzzi	Davide	2111000
Bilato	Leonardo	2071084
Zanella	Francesco	2116442
Romascu	Mihaela-Mariana	2079726
Ogniben	Michele	2042325
Perozzo	Samuele	2110989
Ponso	Giovanni	2000558

Tabella 1: Componenti del gruppo NightPRO.

## 2 Introduzione

### 2.1 Scopo del documento

Questo documento definisce le Norme di Progetto<sub>G</sub> del gruppo NightPRO per il corso di Ingegneria del Software (A.A. 2025/2026). Stabilisce in forma univoca: metodo di lavoro, regole redazionali e criteri di gestione dei documenti, così da garantire coerenza, tracciabilità e chiarezza operativa. La presente versione costituisce la base metodologica iniziale e sarà aggiornata con l'avanzare delle attività.

### 2.2 Glossario

Per garantire chiarezza terminologica e uniformità nella comprensione dei concetti utilizzati, viene redatto un Glossario<sub>G</sub> contenente i termini tecnici e potenzialmente ambigui. Alla **prima occorrenza** nel documento, ogni termine seguito dal simbolo <sub>G</sub> viene definito formalmente nel glossario. Le occorrenze successive non riportano tale marcatura.

### 3 Processi primari

#### 3.1 Fornitura

##### 3.1.1 Descrizione

Questa sezione definisce le norme, gli strumenti e le metodologie che ogni membro del gruppo *NightPRO* si impegna a rispettare. L'obiettivo è garantire una collaborazione efficace e trasparente con l'azienda proponenteG, *Ergon Informatica*.

##### 3.1.2 Gestione della Comunicazione

Il gruppo *NightPRO* si impegna a garantire una comunicazione costante e trasparente con *Ergon Informatica* per tutta la durata del progetto, interfacciandosi primariamente con il referente Gianluca Carlesso. Gli incontri periodici con il referente sono finalizzati a:

- Discutere i requisitiG (necessari, desiderabili, facoltativi) da includere nel prodotto.
- Ricevere feedbackG sullo stato di avanzamento dei lavori.
- Ottenere chiarimenti e risolvere eventuali ambiguità.
- Definire vincoli e requisiti dei processi intermedi.

##### 3.1.3 Strumenti

Di seguito sono elencati i principali strumenti software adottati dal gruppo:

- **Git**: software per il *controllo di versione* (VCS) distribuito, utilizzato per tracciare le modifiche al codice sorgente durante lo sviluppo.
- **GitHub**: piattaforma di *hosting* basata su Git, impiegata per la gestione dei repository, la revisione del codice e la collaborazione.
- **Google Meet**: piattaforma per videoconferenze online. È lo strumento scelto per l'organizzazione di incontri, presentazioni e revisioni periodiche con il proponente.
- **Telegram**: applicazione di messaggistica istantanea, utilizzata per le comunicazioni operative rapide e gli aggiornamenti tempestivi con i referenti aziendali.
- **L<sup>A</sup>T<sub>E</sub>X**: linguaggio di *markup* e sistema di preparazione documentale, scelto dal gruppo per la produzione e la stesura di tutta la documentazione ufficiale del progetto.
- **Overleaf**: piattaforma online per l'*editing* collaborativo di documenti L<sup>A</sup>T<sub>E</sub>X, utilizzata dal gruppo per la redazione collaborativa della documentazione. Una volta completata e verificata una versione, il documento viene caricato direttamente sul branch `main` del repository GitHub.

#### 3.2 Sviluppo

##### 3.2.1 Scopo

L'obiettivo primario della fase di sviluppo è tradurre i requisiti di progetto, definiti con il proponente, in un prodotto software funzionante, verificato e conforme alle specifiche. Questa fase rappresenta il nucleo costruttivo dell'intero progetto.

### 3.2.2 Descrizione

Il processo di sviluppo consiste nell'esecuzione di un insieme strutturato di attività tecniche volte alla realizzazione del software. Il gruppo *NightPRO* adotta un approccio metodico per assicurare che il prodotto finale non solo soddisfi le *aspettative* del proponente, ma rispetti anche elevati standard di qualità in termini di robustezza e manutenibilità.

Il flusso di sviluppo si articola nelle seguenti macro-attività:

- **Progettazione Dettagliata (Design):** Definizione dell'architettura<sub>G</sub> software, delle interfacce dei moduli e dei pattern da implementare, partendo dai requisiti analizzati.
- **Implementazione (Coding):** Scrittura del codice sorgente in linea con la progettazione e gli standard di codifica definiti.
- **Verifica e Test:** Esecuzione di test unitari e di integrazione per assicurare che ogni componente funzioni correttamente e che il sistema nel suo complesso sia privo di difetti.

### 3.2.3 Risultato Atteso

Il gruppo *NightPRO* si impegna a utilizzare il processo di sviluppo per fornire un prodotto software che sia pienamente validato e rispondente ai requisiti concordati con il proponente *Ergon Informatica*. Il risultato finale sarà un software testato, funzionante e pronto per le fasi successive di rilascio o collaudo.

### 3.2.4 Analisi dei Requisiti

**Scopo** L'attività di Analisi dei Requisiti ha lo scopo di raccogliere, documentare e organizzare in modo sistematico tutti i requisiti<sub>G</sub> del prodotto software. Questa fase costituisce il fondamento su cui si basano le successive attività di progettazione, sviluppo e verifica, garantendo che il sistema finale risponda pienamente alle esigenze del proponente e degli stakeholder<sub>G</sub>.

**Descrizione** L'analisi dei requisiti viene condotta attraverso un processo iterativo che prevede:

- **Studio del dominio applicativo:** approfondimento del contesto aziendale e delle problematiche che il sistema deve risolvere.
- **Raccolta dei requisiti:** acquisizione delle esigenze tramite incontri con il proponente, analisi del capitolato<sub>G</sub> e studio della documentazione fornita.
- **Modellazione dei casi d'uso:** definizione degli scenari operativi mediante diagrammi UML<sub>G</sub> dei casi d'uso.
- **Specifiche dei requisiti:** formalizzazione dei requisiti funzionali e non funzionali in forma chiara, completa e verificabile.
- **Tracciabilità:** correlazione bidirezionale tra requisiti e casi d'uso per garantire copertura completa.

**Classificazione dei Requisiti** Ogni requisito viene identificato univocamente mediante un codice strutturato secondo la seguente convenzione:

R[Tipo] [Numero] . [Sottorequisito]

Dove:

- **R:** prefisso che identifica un requisito.
- **Tipo:** indica la natura del requisito:
  - F – Requisito Funzionale: descrive una funzionalità che il sistema deve offrire.
  - Q – Requisito di Qualità: specifica attributi qualitativi del prodotto.
  - V – Requisito di Vincolo: definisce vincoli tecnologici o progettuali.
  - P – Requisito Prestazionale: stabilisce parametri di performance.
  - S – Requisito di Sicurezza: riguarda aspetti di protezione e sicurezza.
- **Numero:** identificativo numerico progressivo all'interno della categoria.
- **Sottorequisito:** numerazione opzionale per requisiti derivati o di dettaglio.

**Esempi:**

- RF1.1 – Requisito Funzionale 1, sottorequisito 1
- RV02 – Requisito di Vincolo numero 2
- RP03 – Requisito Prestazionale numero 3

**Priorità dei Requisiti** Ogni requisito viene classificato in base alla sua priorità:

- **Obbligatorio:** requisito essenziale per il funzionamento del sistema; la sua assenza compromette l'accettazione del prodotto.
- **Desiderabile:** requisito che apporta valore aggiunto al prodotto; la sua implementazione è raccomandata ma non vincolante.
- **Opzionale:** requisito che può essere implementato se le risorse lo consentono, senza impatto sulla validazione del prodotto.

**Casi d'Uso** I casi d'uso (Use Case, UC) descrivono le interazioni tra gli attori e il sistema, rappresentando scenari operativi concreti. Ogni caso d'uso viene identificato mediante la seguente convenzione:

UC [Numero] . [Sottocaso]

Dove:

- **UC:** prefisso che identifica un caso d'uso.
- **Numero:** identificativo numerico progressivo del caso d'uso principale.
- **Sottocaso:** numerazione opzionale per scenari derivati o alternativi.

**Esempi:**

- UC1 – Caso d'uso principale numero 1
- UC1.1 – Primo sottocaso del caso d'uso 1
- UC7.2 – Secondo sottocaso del caso d'uso 7

**Struttura dei Casi d’Uso** Ogni caso d’uso viene documentato con i seguenti elementi:

- **Identificativo e titolo:** codice univoco e nome descrittivo.
- **Attore principale:** soggetto che avvia o partecipa all’interazione.
- **Precondizioni:** stato del sistema prima dell’esecuzione del caso d’uso.
- **Postcondizioni:** stato del sistema al termine dell’esecuzione.
- **Scenario principale:** sequenza ordinata delle azioni che costituiscono il flusso normale.
- **Estensioni:** scenari alternativi o di errore che possono verificarsi.
- **User Story:** descrizione sintetica dell’obiettivo dal punto di vista dell’utente.

**Documentazione di Riferimento** I risultati dell’attività di analisi vengono formalizzati nel documento *Analisi dei Requisiti<sub>G</sub>*, che costituisce il riferimento ufficiale per le fasi successive del progetto. Questo documento include:

- la descrizione del dominio applicativo;
- l’elenco completo dei casi d’uso con relativi diagrammi UML;
- la specifica dettagliata di tutti i requisiti;
- le matrici di tracciabilità tra casi d’uso e requisiti.

### 3.2.5 Progettazione

**Scopo** L’attività di progettazione mira a definire l’architettura tecnica del prodotto in modo che sia robusta, scalabile e in grado di soddisfare pienamente le esigenze di tutti gli *stakeholder*, come identificate in fase di analisi dei requisiti. Attraverso la documentazione di soluzioni tecniche dettagliate e la definizione delle interfacce tra i componenti, questa fase assicura una chiara ripartizione delle responsabilità di sviluppo e semplifica le future operazioni di manutenzione. Il risultato è una base documentale esaustiva che include l’intera struttura del sistema, le specifiche funzionali e non funzionali, e le scelte tecnologiche fondamentali.

**Descrizione** La progettazione è strutturata su più livelli per coprire in modo esaustivo sia gli aspetti funzionali che quelli strutturali del prodotto:

- **Progettazione logica:** Questa fase si concentra sulla selezione dello stack tecnologico (tecnologie, *framework*, librerie). L’obiettivo è motivare l’adeguatezza di tali scelte e confermarne la fattibilità tecnica, tipicamente attraverso la realizzazione di un *Proof of Concept<sub>G</sub>* (PoC).
- **Progettazione di dettaglio:** Basandosi sulle fondamenta stabilite dalla progettazione logica, questa fase definisce l’architettura software specifica. Vengono sviluppate e documentate le rappresentazioni complete di tutte le componenti del sistema e delle loro interazioni.

### 3.2.6 Codifica (MVP)

**Scopo** L’attività di codifica ha lo scopo di tradurre le specifiche di progettazione in codice sorgente eseguibile, producendo un software funzionante, manutenibile e conforme agli standard di qualità definiti dal gruppo. Le norme qui stabilite garantiscono uniformità, leggibilità e tracciabilità del codice prodotto.

**Descrizione** La codifica rappresenta la fase implementativa del processo di sviluppo e si articola nelle seguenti attività:

- **Implementazione delle funzionalità:** traduzione delle specifiche tecniche in codice sorgente.
- **Documentazione del codice:** inserimento di commenti esplicativi e documentazione inline.
- **Versionamento:** gestione delle modifiche attraverso il sistema di controllo versione Git<sub>G</sub>.
- **Test unitari:** verifica del corretto funzionamento delle singole unità di codice.

**Convenzioni di Stile** Per garantire uniformità e leggibilità, il gruppo adotta le seguenti convenzioni di stile:

#### Indentazione

- L'indentazione deve essere di **4 spazi**.
- È vietato l'utilizzo di caratteri di tabulazione (**tab**).
- L'indentazione deve essere coerente in tutto il file sorgente.

#### Lunghezza delle Righe

- La lunghezza massima consigliata per una riga di codice è di **120 caratteri**.
- Le righe che superano tale limite devono essere suddivise in modo logico e leggibile.

#### Spaziatura e Formattazione

- Utilizzare una riga vuota per separare blocchi logici di codice.
- Inserire spazi attorno agli operatori binari (es. **a = b + c**).
- Non inserire spazi prima delle parentesi nelle chiamate a funzione.
- Utilizzare parentesi graffe anche per blocchi di una sola istruzione.

**Convenzioni di Nomenclatura** La nomenclatura degli identificatori segue convenzioni specifiche in base al contesto:

#### Classi e Interfacce

- Utilizzare la notazione **PascalCase<sub>G</sub>**: ogni parola inizia con lettera maiuscola.
- I nomi devono essere sostantivi che descrivono l'entità rappresentata.
- Esempio: **OrderProcessor**, **AudioTranscriber**.

#### Metodi e Funzioni

- Utilizzare la notazione **camelCase<sub>G</sub>**: prima parola minuscola, successive con iniziale maiuscola.
- I nomi devono essere verbi che descrivono l'azione eseguita.
- Esempio: **processOrder()**, **validateInput()**.

### Variabili e Parametri

- Utilizzare la notazione **camelCase**.
- I nomi devono essere descrittivi e significativi.
- Evitare abbreviazioni non standard o nomi di singole lettere (eccetto indici di ciclo).
- Esempio: `orderList`, `confidenceScore`.

### Costanti

- Utilizzare la notazione **UPPER\_SNAKE\_CASE**: tutte maiuscole con underscore come separatore.
- Esempio: `MAX_RETRY_COUNT`, `DEFAULT_TIMEOUT`.

### File Sorgente

- I nomi dei file devono riflettere il contenuto principale (classe, modulo).
- Utilizzare convenzioni coerenti con il linguaggio adottato.

**Documentazione del Codice** Ogni componente del codice deve essere adeguatamente documentato:

- **Intestazione del file**: ogni file sorgente deve contenere un'intestazione con descrizione del contenuto, autore e data di creazione.
- **Documentazione delle funzioni**: ogni funzione pubblica deve essere documentata con descrizione, parametri e valore di ritorno.
- **Commenti inline**: utilizzare commenti per spiegare logiche complesse o non immediatamente intuibili.
- **TODO e FIXME**: utilizzare marcatori standardizzati per segnalare lavori in sospeso o problemi noti.

**Gestione del Versionamento** Il codice sorgente viene gestito attraverso GitG e GitHubG, seguendo le seguenti convenzioni:

### Branch

- **main**: branch principale contenente il codice stabile e rilasciabile.
- **feature/[nome]**: branch per lo sviluppo di nuove funzionalità.
- **fix/[nome]**: branch per la correzione di bug.
- **hotfix/[nome]**: branch per correzioni urgenti su codice in produzione.

### Commit

- I messaggi di commitG devono essere chiari e descrittivi.
- Utilizzare il tempo presente imperativo (es. “Aggiungi validazione input”).
- Ogni commit deve rappresentare un'unità logica di modifica.
- Evitare commit con modifiche non correlate tra loro.

### Pull Request

- Ogni pull request<sub>G</sub> deve essere associata a un task su GitHub Projects<sub>G</sub>.
- La pull request deve contenere una descrizione delle modifiche apportate.
- Prima del merge, è richiesta la revisione da parte di almeno un verificatore.
- I conflitti devono essere risolti prima del merge sul branch principale.

**Standard di Qualità del Codice** Il codice prodotto deve rispettare i seguenti criteri di qualità:

- **Leggibilità:** il codice deve essere facilmente comprensibile da altri sviluppatori.
- **Manutenibilità:** le modifiche future devono poter essere applicate con minimo impatto.
- **Modularità:** il codice deve essere organizzato in componenti indipendenti e riutilizzabili.
- **Testabilità:** ogni componente deve poter essere verificato tramite test automatizzati.
- **Efficienza:** il codice deve essere ottimizzato per le prestazioni, evitando operazioni ridondanti.

## 4 Processi di Supporto

### 4.1 Ambiente collaborativo e infrastruttura del progetto

#### 4.1.1 Obiettivo

L'obiettivo di questa sezione è definire l'insieme di strumenti e configurazioni adottate dal gruppo per supportare la collaborazione, la comunicazione, l'organizzazione e l'integrazione tecnica del progetto.

#### 4.1.2 Comunicazione sincrona e asincrona

**Telegram** Il gruppo utilizza Telegram<sub>G</sub> come canale di comunicazione principale. La scelta è motivata dalla disponibilità della funzionalità topic<sub>G</sub>, che permette di suddividere le conversazioni per argomento (es. documentazione, riunioni, diario di bordo), garantendo ordine e tracciabilità.

**Google Meet** Per le riunioni sincrone viene adottato Google Meet<sub>G</sub>, in quanto:

- consente la partecipazione rapida e trasversale ai membri del gruppo;
- offre funzionalità di condivisione schermo utili durante attività collaborative;
- supporta riunioni sia programmate che estemporanee.

#### 4.1.3 Gestione del repository e versionamento

Il gruppo ha istituito un'organizzazione GitHub<sub>G</sub> denominata NightPRO, all'interno della quale sono presenti tre repository<sub>G</sub> distinti:

- **Documentazione**: dedicato alla gestione della documentazione di progetto, configurazione del sito pubblico e automazione CI<sub>G</sub>;
- **SmartOrder-PoC**: dedicato al PoC<sub>G</sub> (proof of concept) del progetto SmartOrder;
- **SmartOrder-MVP**: dedicato al MVP<sub>G</sub> (minimum viable product) del progetto SmartOrder.

**Struttura del repository documentazione** La struttura adottata è la seguente:

```
.github/workflows/    # Workflow CI (build PDF, deploy sito)
src/                  # File .tex sorgenti dei documenti
docs/                 # PDF generati automaticamente
site/                 # Codice sorgente sito GitHub Pages
template/              # Template e documenti base
report.md             # Report compilazione automatica
```

#### 4.1.4 Gestione delle attività

Per la pianificazione, il monitoraggio delle attività e la tracciabilità dello stato di avanzamento, il gruppo utilizza GitHub Projects<sub>G</sub>. Questa piattaforma permette di:

- assegnare responsabilità chiare ai membri del gruppo;
- monitorare l'avanzamento delle attività;
- mantenere una visione condivisa delle priorità;

- aggiornare in tempo reale lo stato dei task<sub>G</sub>.

In questa fase iniziale il flusso operativo è in definizione e verrà progressivamente raffinato in base alle esigenze progettuali e all’evoluzione delle attività. L’obiettivo è arrivare a un sistema di lavoro che garantisca trasparenza, coordinamento ed efficienza, riducendo il rischio di sovrapposizioni o attività non monitorate.

#### 4.1.5 Sistema di pubblicazione e consultazione della documentazione

Per garantire un accesso semplice, centralizzato e sempre aggiornato alla documentazione di progetto, è stato sviluppato un sito web statico pubblicato tramite **GitHub Pages**<sub>G</sub>.

**Obiettivo** Rendere la documentazione facilmente consultabile da tutti gli stakeholder<sub>G</sub> (membri del team, committenti e corpo docente), mantenendo coerenza, ordine e tracciabilità delle versioni.

**Funzionalità principali** Il sito offre:

- navigazione dei documenti tramite struttura a cartelle collassabili;
- motore di ricerca interno per nome, data o versione del documento;
- possibilità di apertura o download diretto dei PDF;
- pagina informativa con i membri del gruppo e relativi riferimenti;
- selezione del tema grafico chiaro, scuro o sistema.

**Aggiornamento automatico** La struttura del sito e l’elenco dei documenti vengono aggiornati automaticamente ad ogni esecuzione della pipeline<sub>G</sub> di pubblicazione. Durante la build<sub>G</sub>, un processo analizza la cartella `docs/` e genera la mappa della documentazione pubblicata, garantendo che il sito rifletta sempre lo stato più recente e valido dei file. Grazie a questa configurazione, la consultazione della documentazione è:

- automatizzata,
- coerente con il repository principale,
- priva di interventi manuali,
- sempre aggiornata all’ultima versione approvata.

#### 4.1.6 Automazione tramite GitHub Actions

Al fine di garantire coerenza, tracciabilità e aggiornamento continuo della documentazione, il gruppo ha configurato due workflow<sub>G</sub> automatizzati tramite **GitHub Actions**<sub>G</sub>: uno dedicato alla compilazione dei documenti `LATeX` e uno alla pubblicazione del sito web. L’obiettivo è ridurre al minimo gli interventi manuali, eliminare errori operativi e mantenere un processo documentale affidabile, verificabile e riproducibile.

**Compilazione automatica della documentazione** Il workflow di build è attivato al verificarsi di modifiche ai file sorgenti L<sup>A</sup>T<sub>E</sub>X presenti nella directory `src/`. Il sistema esegue una serie di controlli e operazioni automatiche per garantire la coerenza tra sorgenti e PDF generati:

- rileva le modifiche rispetto all’ultima build valida e identifica i file da ricompilare;
- elimina PDF obsoleti o non più associati a sorgenti esistenti, escludendo i PDF firmati che vengono gestiti manualmente;
- compila i progetti L<sup>A</sup>T<sub>E</sub>X tramite `latexmk` in ambiente Docker<sub>G</sub>;
- genera un file `report.md` contenente l’esito della compilazione per ciascun documento (con rispettivo link diretto ai pdf generati dalla build);
- effettua automaticamente un `commitG` dei soli file effettivamente aggiornati.

Questa procedura assicura che:

- nella cartella `docs/` siano presenti solamente PDF aggiornati e validi;
- nessun documento obsoleto rimanga nel repository;
- eventuali errori di compilazione impediscono la pubblicazione di versioni incoerenti.

**Pubblicazione automatica della documentazione online** Al termine di una compilazione L<sup>A</sup>T<sub>E</sub>X completata con esito positivo, un secondo workflow provvede automaticamente alla pubblicazione della documentazione tramite GitHub Pages. Il processo si occupa di:

- sincronizzare la struttura del sito con la cartella `docs/`;
- includere automaticamente nuove versioni o nuovi documenti;
- rendere il contenuto immediatamente accessibile online.

La pubblicazione avviene senza intervento manuale e garantisce che solo materiale correttamente compilato e validato venga reso disponibile. Questo meccanismo assicura un punto di accesso unico e costantemente aggiornato alla documentazione ufficiale del progetto.

## 4.2 Documentazione

### 4.2.1 Obiettivo

La documentazione rappresenta un elemento fondamentale del progetto: definisce i processi, registra le decisioni, formalizza gli avanzamenti e costituisce un riferimento verificabile e condiviso. Questa sezione definisce regole, convenzioni e ciclo di vita<sub>G</sub> adottati dal gruppo per la produzione dei documenti ufficiali. Poiché il progetto si trova nella fase di candidatura, tali norme costituiscono una baseline che sarà estesa con l’avanzare delle attività progettuali.

### 4.2.2 Formati di riferimento

L’intera documentazione ufficiale viene redatta in formato L<sup>A</sup>T<sub>E</sub>X<sub>G</sub> (`.tex`) e distribuita in formato PDF<sub>G</sub> (`.pdf`).

- Il formato L<sup>A</sup>T<sub>E</sub>X garantisce qualità tipografica, modularità, controllabilità delle modifiche e standardizzazione.

- Il formato **PDF** costituisce la versione ufficiale, consultabile e non modificabile.

Le due forme convivono con ruoli distinti:

<b>File .tex</b>	Codice sorgente, modificabile, tracciabile
<b>File .pdf</b>	Documento ufficiale distribuito e verificabile

#### 4.2.3 Struttura dei Documenti

Ogni documento prodotto segue una struttura coerente per garantire leggibilità e uniformità:

- Frontespizio<sub>G</sub> con dati e metadati ufficiali
- Tabella delle versioni e delle modifiche
- Indice dei contenuti
- Corpo del documento, articolato in sezioni e sottosezioni
- (Se necessario) Appendici, glossari, tabelle e riferimenti

Template<sub>G</sub> condivisi sono forniti nel repository del progetto per garantire omogeneità.

#### 4.2.4 Convenzioni di Scrittura e Nomenclatura

Per assicurare tracciabilità e organizzazione dei file, si adottano le seguenti convenzioni.

**Nomi dei file** La convenzione utilizzata è snake\_case<sub>G</sub>. Quando rilevante, si includono versione e/o data:

- La versione è indicata come vX.Y
- La data è riportata nel formato ISO AAAA-MM-GG

Esempi:

- norme\_di\_progetto\_v0.1.tex
- verbale\_interno\_2025-10-29.tex

**Documenti firmati** In caso di documenti soggetti a firma, la copia firmata mantiene lo stesso nome del documento ufficiale aggiungendo il suffisso **\_firmato** (o **\_signed**):

- norme\_di\_progetto\_v0.1\_firmato.pdf

Questo consente distinzione chiara tra versione certificata e versioni operative.

#### 4.2.5 Ciclo di Vita dei Documenti

Tutti i documenti prodotti dal gruppo seguono un processo di revisione volto a garantire qualità, coerenza e tracciabilità. Il ciclo di vita adottato è iterativo e prevede controlli continui ad ogni modifica significativa. Le fasi previste sono:

- **Redazione<sub>G</sub>** Il documento viene creato o aggiornato da uno o più redattori. La stesura avviene in modo collaborativo utilizzando **Overleaf<sub>G</sub>**, una piattaforma online per l'editing collaborativo di documenti **LATEX**. Una volta completata e verificata una nuova versione del documento, questa viene caricata direttamente sul branch **main** del repository GitHub, senza l'utilizzo di branch intermedi per la documentazione.
- **Verifica<sub>G</sub>** Ogni commit contenente contenuto documentale richiede una verifica da parte di almeno un membro diverso dal redattore. La verifica assicura:
  - coerenza con gli standard adottati;
  - correttezza dei contenuti;
  - qualità formale e lessicale;
  - aderenza alle norme di progetto.

Un documento è considerato verificato quando tutte le modifiche proposte sono state esaminate e validate.

- **Approvazione<sub>G</sub> (solo per i documenti ufficiali)** Per i documenti formali del progetto (es. Lettera di Presentazione, Valutazione Capitolati, Preventivo dei Costi) è prevista una fase di approvazione finale, successiva alla verifica. L'approvazione certifica che il documento è completo, corretto e pronto per essere rilasciato nella sua versione ufficiale.
- **Pubblicazione<sub>G</sub>** Dopo la verifica (per verbali) o dopo l'approvazione (per documenti ufficiali), il documento viene inserito nell'archivio PDF (con il sistema automatico descritto in precedenza), indicizzato e pubblicato sul portale del gruppo.

#### 4.2.6 Verbali

I verbali<sub>G</sub> sono i documenti che riportano le discussioni e le decisioni prese durante gli incontri ufficiali del gruppo. Hanno lo scopo di tracciare l'evoluzione del progetto e formalizzare gli impegni presi. Ogni verbale prodotto dal gruppo NightPRO deve essere strutturato nelle seguenti sezioni principali:

- **Sezione 1: Informazioni Generali** Contiene i metadati della riunione. È suddivisa in:
  - Componenti del Gruppo: La tabella standard con l'elenco dei membri.
  - Dettagli Riunione: Elenco puntato con Data, Ora, Luogo, Partecipanti (con eventuali assenti), Redatto da, Verificato da e Versione del verbale.
- **Sezione 2: Ordine del Giorno<sub>G</sub> (Agenda)** Un elenco puntato che elenca tutti gli argomenti pianificati per la discussione.
- **Sezione 3: Diario della Riunione<sub>G</sub>** Il resoconto dettagliato della discussione. Questa sezione è suddivisa in sottosezioni che rispecchiano i punti dell'Ordine del Giorno, riportando le analisi e i fatti emersi.
- **Sezione 4: Decisioni Prese** Un elenco numerato che riassume in modo chiaro e sintetico tutte le decisioni ufficiali deliberate dal gruppo durante l'incontro.

- **Sezione 5: Attività da Svolgere (To-Do)** Una tabella dei task decisi durante la riunione per il prossimo periodo di sprint, con assegnato il ruoli di riferimento.
- **Sezione 6: Definizione dei ruoli (Preventivo)** Una tabella con l'assegnazione dei ruoli ai membri del team, con una indicazione preventiva delle ore lavorative previste.

## 5 Processi organizzativi

### 5.1 Finalità

Questa sezione stabilisce le norme e le procedure che regolano l'organizzazione interna del gruppo. Definisce i meccanismi di coordinamento, comunicazione e gestione delle risorse umane necessari per condurre efficacemente il progetto lungo tutto il suo sviluppo. Le regole qui definite riguardano l'assegnazione dei ruoli, la pianificazione e il monitoraggio delle attività, la gestione degli incontri, i canali di comunicazione, nonché i processi di apprendimento e di ottimizzazione dei metodi di lavoro.

### 5.2 Ambito di applicazione

I dettagli operativi e le tempistiche specifiche relative ai processi organizzativi sono riportati nel documento *Piano di Progetto<sub>G</sub>*. Le norme qui presentate coprono i seguenti ambiti:

- Ruoli e responsabilità dei membri del team;
- Comunicazione interna ed esterna;
- Pianificazione e tracciamento delle attività;
- Sviluppo delle competenze;
- Ottimizzazione dei processi.

### 5.3 Ruoli e responsabilità

La distribuzione dei ruoli tra i membri del gruppo è di competenza del *Responsabile di Progetto*, che deve assicurare una rotazione equa delle responsabilità. Nel corso del progetto, ogni componente del team dovrà ricoprire almeno una volta ciascuno dei ruoli previsti. La copertura simultanea di tutti i ruoli non è sempre necessaria, in quanto dipende dalla fase di sviluppo in cui si trova il progetto.

**Politica di rotazione dei ruoli** Per garantire che tutti i membri del team acquisiscano esperienza in ogni ruolo previsto, il gruppo ha definito una politica di rotazione strutturata. Durante le prime sette settimane di progetto (Fase Iniziale), si adotta una rotazione a cadenza settimanale, con l'obiettivo di assicurare che ogni membro abbia ricoperto tutti i ruoli disponibili. Una volta completato il ciclo di rotazione iniziale (Fase a Regime, post-settimana 7), la rotazione avverrà ogni 2 settimane. Il team si riserva la facoltà di adottare cadenze differenti esclusivamente per gestire occasioni eccezionali o specifiche necessità progettuali urgenti.

Lo switch formale dei ruoli avviene il **Sabato** (alla mezzanotte tra Venerdì e Sabato), per permettere ai membri di completare le attività amministrative derivanti dalla riunione del venerdì mantenendo il ruolo corrente.

Di seguito vengono descritti i sei ruoli previsti e le relative responsabilità.

#### 5.3.1 Responsabile di Progetto

Il Responsabile di Progetto funge da interfaccia principale tra il gruppo e le parti esterne, in particolare con il *committente<sub>G</sub>* e l'*azienda proponente<sub>G</sub>*. Supervisiona l'avanzamento del progetto ed è responsabile del coordinamento generale delle attività del team e dell'approvazione delle decisioni strategiche.

**Compiti principali** Nello specifico, il Responsabile di Progetto, con cadenza periodica (ad ogni sprint), deve:

- Redigere il verbale della riunione interna: dedicata alla review del periodo concluso e alla pianificazione dello sprint successivo. Il Responsabile redige il verbale della riunione in cui assume il ruolo (inizio mandato), e non quello della riunione che andrà a gestire.
- Gestire GitHub Projects: aggiornare i task, inserire le attività emerse dalle riunioni e assegnarle ai membri del team.
- Curare le comunicazioni esterne: mantenere i contatti con proponente e committente, garantendo un canale di comunicazione trasparente e tempestivo.
- Redigere ed esporre il diario di bordo: documentare attività, avanzamento e stato generale del progetto.
- Gestire la riunione interna di sprint: coordinare lo svolgimento dell'incontro e assicurare la trattazione di tutti i punti previsti.
- Compilare il Piano di Progetto: aggiornare la sezione relativa al periodo corrente, indicando attività svolte e le risorse impiegate.
- Approvare la documentazione di milestone: verificare e approvare la documentazione ufficiale prodotta dal team al raggiungimento delle milestone.
- Definire pianificazione e risorse: contribuire alla definizione degli obiettivi, alla pianificazione delle scadenze e all'allocazione delle risorse disponibili.

### 5.3.2 Amministratore

L'Amministratore ha la responsabilità di controllare e amministrare l'ambiente di lavoro del gruppo, garantendo che strumenti e infrastrutture siano sempre funzionanti e accessibili.

**Compiti principali** L'Amministratore si occupa della salvaguardia e della gestione della documentazione di progetto, curando il sistema di archiviazione e versionamento sia per la documentazione che per il codice sorgente. Amministra l'infrastruttura tecnologica e gli strumenti utilizzati dal team, mantenendo l'ambiente di sviluppo efficiente. Fornisce supporto tecnico ai membri del gruppo, risolvendo errori e gestendo le segnalazioni di malfunzionamenti. Si dedica all'automazione dei processi ricorrenti, identifica opportunità di miglioramento e implementa soluzioni per ottimizzare l'ambiente di lavoro. Esegue il controllo delle versioni e delle configurazioni del prodotto software, gestendo il sistema di versionamento e configurazione. Redige e attua i piani e le procedure per la gestione della qualità, assicurando l'efficacia e l'efficienza dei processi. Fornisce inoltre supporto per la gestione delle risorse e delle comunicazioni del progetto.

### 5.3.3 Analista

L'Analista si occupa dell'analisi dei requisiti<sub>G</sub> e della definizione delle specifiche del progetto, traducendo le esigenze del cliente in requisiti formali e verificabili.

**Compiti principali** L'Analista raccoglie e analizza i requisiti del cliente, interpretando i bisogni del proponente e trasformandoli in aspettative chiare che il gruppo deve soddisfare per realizzare un prodotto di qualità professionale. Definisce le *specifiche funzionali<sub>G</sub>* e *tecniche<sub>G</sub>* del prodotto, sviluppa una modellazione concettuale del sistema e organizza i requisiti in categorie logiche. Collabora strettamente con il progettista per individuare soluzioni che soddisfino i requisiti

individuati. Redige il documento *Analisi dei Requisiti<sub>G</sub>*, descrivendo in dettaglio i servizi che il sistema deve fornire e garantendo che tutti i requisiti siano espressi in modo chiaro, completo e non ambiguo. Studia approfonditamente il problema e il *contesto applicativo<sub>G</sub>*, valutando la complessità del dominio e identificando sia i requisiti esplicativi che quelli impliciti. Assicura che i requisiti siano definiti con sufficiente precisione per evitare ambiguità nelle fasi successive di progettazione e sviluppo.

### 5.3.4 Progettista

Il Progettista ha il compito di trasformare i requisiti emersi dalla fase di analisi in un'architettura<sub>G</sub> software coerente e realizzabile.

**Compiti principali** Il Progettista progetta l'architettura del sistema in modo che soddisfi tutti i requisiti identificati, privilegiando soluzioni con elevata manutenibilità e ridotto *accoppiamento<sub>G</sub>* tra i componenti. Prende decisioni riguardanti gli aspetti tecnici e tecnologici del progetto, selezionando le tecnologie più appropriate per garantire efficacia ed efficienza. Valuta e seleziona eventuali *pattern architettonici<sub>G</sub>* da adottare e sviluppa lo *schema UML<sub>G</sub>* delle classi che modella la struttura del sistema. Garantisce che la soluzione proposta sia economicamente sostenibile e mantenibile, rispettando i vincoli di budget definiti nel preventivo. Collabora con gli analisti per comprendere appieno i requisiti e con i programmatore per guidare l'implementazione delle soluzioni tecniche. Coordina le attività di progettazione per assicurare che il prodotto finale soddisfi tutti i requisiti. Si dedica inoltre alla ricerca e all'approfondimento delle conoscenze tecniche, esplorando strumenti e tecnologie innovative che possano migliorare l'architettura del sistema.

### 5.3.5 Programmatore

Il Programmatore realizza l'implementazione concreta delle soluzioni tecniche definite dal progettista, trasformando le specifiche di progettazione in codice funzionante.

**Compiti principali** Il Programmatore scrive codice che aderisce rigorosamente alle specifiche di progettazione, applicando le "*best practices<sub>G</sub>*" consolidate nel settore per garantire qualità e leggibilità. Risolve i problemi tecnici che emergono durante lo sviluppo, trovando soluzioni efficaci e compatibili con l'architettura definita. Esegue test sul proprio codice per verificare che funzioni correttamente e soddisfi i requisiti specificati. Scrive codice ben documentato, mantenibile e correttamente versionato, facilitando così le attività di verifica e manutenzione. Redige la documentazione necessaria per la comprensione e l'utilizzo del codice prodotto. Collabora attivamente con il progettista e gli altri membri del team per implementare correttamente l'architettura definita nella fase di progettazione.

### 5.3.6 Verificatore

Il Verificatore controlla il lavoro prodotto dagli altri membri del gruppo, assicurando che rispetti gli standard di qualità e le norme di progetto stabilite.

**Compiti principali** Il Verificatore esamina i documenti redatti su Overleaf prima che vengano caricati sul repository, verificandone la conformità alle Norme di Progetto. Per quanto riguarda il codice software, esamina i file caricati in un *branch<sub>G</sub>* protetto della repository e, durante la revisione di una *pull request<sub>G</sub>*, controlla i file modificati o aggiunti per individuare errori ortografici, sintattici, logici e problemi di build. Esamina i prodotti in fase di revisione utilizzando le tecniche e gli strumenti definiti nelle Norme di Progetto, verificando che siano conformi ai

requisiti funzionali e di qualità. Segnala eventuali errori o non conformità riscontrati durante la verifica, fornendo *feedback<sub>G</sub>* dettagliati e costruttivi a chi ha prodotto il lavoro. Assicura che la qualità di quanto prodotto sia conforme agli standard imposti, verificando la conformità in ogni fase del ciclo di vita del prodotto. Redige la parte retrospettiva del *Piano di Qualifica<sub>G</sub>*, documentando le verifiche e le prove effettuate. Mantiene una sorveglianza continua durante l'intera durata del progetto, garantendo che tutte le attività rispettino il livello di qualità atteso.

## 5.4 Comunicazione e incontri

La gestione efficace della comunicazione e degli incontri è fondamentale per garantire il coordinamento tra i membri del team e con le parti esterne. Questa sezione definisce i canali e le modalità di comunicazione adottate dal gruppo, distinguendo tra comunicazioni interne ed esterne.

### 5.4.1 Canali di comunicazione interna

Le comunicazioni tra i membri del gruppo avvengono principalmente attraverso due strumenti, scelti in base alle esigenze specifiche.

**Telegram** Telegram è il canale principale per le comunicazioni asincrone interne. Viene utilizzato per scambi rapidi di messaggi testuali e vocali, nonché per la pianificazione e l'organizzazione degli incontri interni. La scelta di questo strumento è stata motivata dalla disponibilità della funzionalità topic, che consente di organizzare le conversazioni per argomento, migliorando l'ordine e la tracciabilità delle discussioni.

**Google Meet** Google Meet viene utilizzato per le riunioni sincrone interne quando è necessario un confronto diretto tra i membri del gruppo. Questo strumento permette la partecipazione simultanea di tutti i membri e offre funzionalità di condivisione schermo particolarmente utili durante attività collaborative.

### 5.4.2 Canali di comunicazione esterna

Le comunicazioni con il proponente e i committenti sono gestite dal Responsabile di Progetto e avvengono attraverso canali dedicati.

**Google Meet** Google Meet è utilizzato per le riunioni esterne con il proponente e i committenti. Queste riunioni possono essere richieste da entrambe le parti in base alle necessità del progetto. Al termine di ogni riunione esterna viene redatto un verbale per documentare gli argomenti trattati e le decisioni prese.

**Email** L'email viene utilizzata per comunicazioni formali e dettagliate con le parti esterne. L'indirizzo email del gruppo ([swe.nightpro@gmail.com](mailto:swe.nightpro@gmail.com)) è condiviso tra tutti i membri del team per garantire trasparenza e accessibilità alle comunicazioni ufficiali.

**Telegram** Per le comunicazioni operative e le richieste che necessitano di una risposta rapida da e verso l'azienda, si utilizza l'applicazione di messaggistica istantanea Telegram. Questo canale è dedicato alla gestione agile<sub>G</sub> delle questioni quotidiane e agli aggiornamenti tempestivi che non richiedono la struttura formale di un'email.

### 5.4.3 Organizzazione degli incontri

Gli incontri del gruppo si distinguono in interni ed esterni, ciascuno con modalità e finalità specifiche.

**Riunioni interne** Le riunioni interne coinvolgono esclusivamente i membri del gruppo e si svolgono con cadenza regolare, tipicamente una volta a settimana. La riunione principale del gruppo si tiene ogni venerdì mattina ed è gestita dal Responsabile di Progetto. Qualsiasi membro del gruppo può richiedere un incontro aggiuntivo al Responsabile di Progetto, che provvederà a organizzarlo in base alla disponibilità di tutti i partecipanti. Le riunioni si svolgono prevalentemente in modalità virtuale tramite Google Meet.

Per garantire efficienza e produttività, ogni riunione interna segue una struttura ben definita. Prima dell'incontro viene preparata una scaletta con i principali punti da discutere. Durante la riunione si discute del lavoro svolto da ogni membro dall'ultimo incontro e si affrontano i punti previsti nella scaletta, con spazio per il confronto su eventuali dubbi o problematiche emerse. Infine, si pianificano le attività da svolgere per ogni membro fino al prossimo incontro.

Al termine di ogni riunione interna, il Responsabile di Progetto redige il verbale, contenente una descrizione dei punti principali discussi durante l'incontro. Ai fini della coerenza organizzativa, il Responsabile redige il verbale della riunione in cui assume l'incarico (inizio mandato), e non di quella successiva che andrà a gestire. Una volta completato, il verbale viene verificato da un altro membro del gruppo e approvato dal Responsabile prima della pubblicazione.

**Riunioni esterne** Le riunioni esterne coinvolgono i membri del gruppo insieme ai referenti aziendali o ai committenti. La frequenza e le modalità di svolgimento di questi incontri verranno stabilite in accordo con il proponente. Entrambe le parti possono richiedere incontri aggiuntivi quando necessario. Le riunioni esterne si svolgono principalmente in modalità virtuale tramite Google Meet. Al termine di ogni riunione esterna viene redatto un verbale che documenta i momenti salienti e le decisioni prese; quando necessario, il verbale viene sottoposto ai referenti per approvazione.

#### 5.4.4 Responsabilità durante gli incontri

Durante gli incontri, sia il Responsabile di Progetto che i partecipanti hanno responsabilità specifiche da rispettare.

**Responsabilità del Responsabile di Progetto durante gli incontri** Il Responsabile di Progetto ha il compito di pianificare l'ordine del giorno delle riunioni, comunicare tempestivamente eventuali variazioni orarie e verificare la presenza dei membri. Durante la riunione, guida le discussioni in modo ordinato ed efficace, assicurando che tutti i punti previsti vengano affrontati.

In particolare, per la riunione del venerdì mattina, il Responsabile di Progetto:

- gestisce e coordina lo svolgimento dell'incontro;
- redige il verbale della riunione (il verbale viene redatto nella riunione in cui assume l'incarico, non in quella successiva);
- inserisce su GitHub Projects le attività emerse durante la riunione, assegnandole ai membri del gruppo;
- aggiorna il diario di bordo con le attività svolte e lo stato di avanzamento del progetto;
- conclude il suo ruolo al termine del periodo assegnato, documentando lo stato del progetto e le attività in corso per garantire una transizione fluida al prossimo Responsabile.

**Responsabilità dei partecipanti** I partecipanti alle riunioni si impegnano a partecipare puntualmente, comunicando tempestivamente eventuali ritardi o assenze. Durante gli incontri devono partecipare attivamente alle discussioni e mantenere un comportamento professionale e rispettoso.

## 5.5 Pianificazione e tracciamento delle attività

Una gestione efficace dei compiti e dei task è essenziale per mantenere l'organizzazione del progetto, garantendo tracciabilità, qualità e rispetto delle tempistiche. Questa sezione illustra l'approccio metodologico adottato dal team e descrive il processo che regola lo svolgimento di ciascun task.

### 5.5.1 Approccio metodologico

Il team adotta il metodo di sviluppo *Agile<sub>G</sub>* per organizzare e pianificare le attività progettuali. Questo approccio favorisce una gestione flessibile e iterativa, incoraggiando la collaborazione continua tra i membri del team e una comunicazione trasparente con il committente.

I task vengono definiti, pianificati e assegnati utilizzando **GitHub Projects<sub>G</sub>**, uno strumento che agevola la condivisione delle responsabilità e il lavoro collaborativo. Il sistema permette di visualizzare lo stato delle attività e di tracciare l'avanzamento, individuando rapidamente eventuali blocchi o ritardi. Il team valuta periodicamente i risultati ottenuti e pianifica i miglioramenti per i cicli successivi, garantendo un adattamento flessibile ai cambiamenti nei requisiti e promuovendo il coinvolgimento attivo di tutti i partecipanti.

### 5.5.2 Processo di gestione dei task

Il *ciclo di vita<sub>G</sub>* di ciascun task segue un processo strutturato, gestito attraverso GitHub Projects per assicurare coerenza e tracciabilità.

Il processo inizia con la **creazione** del task da parte del Responsabile, che definisce su GitHub Projects tutti i campi essenziali: titolo, descrizione, priorità, assegnatari e stima temporale, associandolo eventualmente a una milestone<sub>G</sub>.

Nella fase di **assegnazione**, i task vengono distribuiti ai membri del team in base alle loro competenze e al carico di lavoro corrente. Se un task rimane non assegnato, i membri possono prendersene carico autonomamente.

Durante l'**esecuzione**, l'assegnatario aggiorna lo stato del task (ad esempio da "To Do" a "In Progress"). Per i task di documentazione, il lavoro viene svolto su Overleaf in modo collaborativo. Per i task di sviluppo software, l'assegnatario lavora su un branch dedicato, garantendo una gestione separata e tracciabile delle modifiche.

Una volta completato, il task entra in fase di **revisione**. Il Verificatore esamina il lavoro prodotto: per la documentazione, verifica i contenuti su Overleaf prima del caricamento su GitHub; per il codice software, segnala eventuali modifiche necessarie tramite commenti o review. Valida il task se conforme agli standard stabiliti.

Infine, nella fase di **accettazione**, il Responsabile approva il task completato. Per la documentazione verificata, viene effettuato il push diretto sul branch **main** del repository. Per il codice software, viene eseguito il merge del branch quando pertinente. Lo stato del task viene aggiornato a "Completato".

Questo processo strutturato consente al team di gestire le attività con precisione, tracciando ogni modifica e mantenendo una visione chiara e condivisa dell'avanzamento.

## 5.6 Sviluppo delle competenze

Il processo di formazione e aggiornamento dei membri del team è fondamentale per garantire che tutti possiedano le competenze necessarie per svolgere efficacemente le proprie attività.

### 5.6.1 Obiettivi formativi

Il processo di formazione mira a sviluppare le competenze necessarie per la produzione della documentazione e per la realizzazione del prodotto software richiesto. Gli obiettivi formativi includono:

- l'acquisizione di una conoscenza solida del linguaggio L<sup>A</sup>T<sub>E</sub>X;
- lo sviluppo di dimestichezza con i linguaggi di programmazione, le librerie e gli strumenti necessari per il prodotto software assegnato dal proponente;
- la familiarizzazione con l'ambiente di lavoro relativo al capitolato C8.

### 5.6.2 Modalità di apprendimento

Il gruppo adotta un approccio flessibile alla formazione, permettendo a ciascun membro di scegliere il metodo di apprendimento più adatto alle proprie esigenze. Questa libertà consente di ampliare la visione del gruppo sulle tecnologie utilizzate e migliorare le scelte implementative attraverso l'esperienza diretta. La rotazione periodica dei ruoli, come definita nella sezione "Ruoli e responsabilità", garantisce che ogni membro del team acquisisca esperienza pratica in tutti i ruoli previsti.

## 5.7 Ottimizzazione dei processi

Il gruppo adotta un approccio di miglioramento continuo per ottimizzare costantemente i processi di lavoro e la qualità dei prodotti.

### 5.7.1 Ciclo di Deming (PDCA)

Per garantire un miglioramento continuo e sistematico, il gruppo NightPRO adotta il **Ciclo di Deming<sub>G</sub>**, noto anche come PDCA<sub>G</sub> (*Plan-Do-Check-Act*). Questo modello iterativo guida tutte le attività di ottimizzazione dei processi attraverso quattro fasi interconnesse:

- **Plan (Pianificare)**: definire gli obiettivi da raggiungere e i processi necessari per ottenere risultati conformi alle aspettative. In questa fase si identificano le attività da svolgere, si stabiliscono le metriche di riferimento e si pianificano le risorse necessarie.
- **Do (Fare)**: attuare il piano stabilito, eseguire i processi definiti e realizzare le attività previste. Durante questa fase si raccolgono dati e informazioni utili per la successiva valutazione.
- **Check (Verificare)**: monitorare e misurare i processi e i risultati ottenuti confrontandoli con gli obiettivi pianificati, le politiche stabilite e i requisiti definiti. Si analizzano gli scostamenti e si identificano le cause di eventuali difformità.
- **Act (Agire)**: adottare azioni correttive per eliminare le cause degli scostamenti rilevati e implementare miglioramenti nei processi. Le modifiche validate vengono consolidate e diventano la nuova baseline per il ciclo successivo.

Questo ciclo viene applicato in modo ricorrente durante tutto il progetto, garantendo un adattamento continuo alle esigenze emergenti e un progressivo incremento dell'efficacia dei processi.

### 5.7.2 Metriche di Processo e di Prodotto

Per supportare la fase di *Check* del ciclo PDCA e garantire decisioni basate su dati oggettivi, il gruppo utilizza un sistema strutturato di metriche quantitative. Le metriche adottate sono definite e documentate nel *Piano di Qualifica<sub>G</sub>* e si suddividono in due categorie principali:

**Metriche di Processo** Le metriche di processo misurano l'efficacia e l'efficienza delle attività svolte dal team. Il gruppo adotta la metodologia **Earned Value Management<sub>G</sub>** (EVM) per monitorare l'avanzamento del progetto in termini di costi e tempi. Le principali aree di misurazione includono:

- **Gestione del budget:** scostamenti tra costi preventivati e consuntivati, indici di efficienza economica.
- **Gestione dei tempi:** varianze orarie, rispetto delle scadenze pianificate, indici di performance temporale.
- **Avanzamento del progetto:** valore del lavoro completato, proiezioni di completamento, stime a finire.

**Metriche di Prodotto** Le metriche di prodotto valutano la qualità degli artefatti prodotti, sia documentali che software. Comprendono:

- **Metriche di documentazione:** indice Gulpease<sub>G</sub> per la leggibilità, correttezza ortografica e conformità agli standard redazionali.
- **Metriche di codice:** copertura dei test, complessità ciclomatica, densità di commenti, rispetto delle convenzioni di stile.

**Soglie di Accettazione** Per ciascuna metrica sono definite soglie di accettazione che distinguono tra:

- **Valore preferibile:** obiettivo ottimale da perseguire.
- **Valore accettabile:** soglia minima al di sotto della quale è richiesto un intervento correttivo.

I valori specifici delle soglie e le formule di calcolo sono riportati nel documento *Piano di Qualifica*. Il monitoraggio delle metriche avviene con cadenza regolare (tipicamente al termine di ogni sprint) e i risultati vengono discussi durante le riunioni periodiche per identificare eventuali azioni correttive.

### 5.7.3 Meccanismi di implementazione

L'implementazione del miglioramento continuo avviene attraverso meccanismi strutturati che coinvolgono tutto il team e si integrano con il ciclo PDCA.

Le problematiche riscontrate durante lo sviluppo e le soluzioni adottate vengono sistematicamente documentate e analizzate. Questo permette una gestione consapevole e mirata delle sfide incontrate, evitando la ripetizione di errori già commessi e fornendo un repertorio di soluzioni efficaci per situazioni simili.

Durante le riunioni periodiche, il team:

- analizza i valori delle metriche raccolte durante lo sprint concluso;
- confronta i risultati ottenuti con le soglie di accettazione definite;

- identifica le aree di successo e le opportunità di miglioramento;
- formula azioni correttive concrete da implementare nel ciclo successivo.

Le azioni correttive stabilite vengono tracciate attraverso GitHub Projects<sub>G</sub> e documentate nei verbali delle riunioni. Al termine del ciclo successivo, l'efficacia delle azioni intraprese viene valutata attraverso il confronto delle metriche, chiudendo così il ciclo PDCA e avviando una nuova iterazione di miglioramento.

## 5.8 Gestione dei rischi

La gestione dei rischi consente di anticipare e mitigare potenziali problematiche che potrebbero compromettere il successo del progetto. Il gruppo adotta un approccio strutturato, documentato nel *Piano di Progetto<sub>G</sub>*, articolato in cinque fasi:

- **Identificazione:** individuazione sistematica delle fonti di rischio attraverso l'analisi delle attività, degli strumenti e delle dinamiche organizzative.
- **Analisi:** valutazione della probabilità di occorrenza e del grado di pericolosità di ciascun rischio.
- **Valutazione:** determinazione delle priorità e definizione dell'ordine di attuazione delle misure di mitigazione.
- **Gestione:** implementazione di misure preventive e azioni di mitigazione mirate.
- **Monitoraggio:** controllo periodico dell'efficacia delle soluzioni adottate e identificazione di nuovi rischi emergenti.

### 5.8.1 Classificazione dei rischi

I rischi sono identificati mediante la convenzione R[Tipo] [Indice], dove:

- **RT:** rischi tecnologici (strumenti, tecnologie, infrastrutture);
- **RO:** rischi organizzativi (gestione, comunicazione, coordinamento);
- **RP:** rischi personali (disponibilità, competenze, impegni individuali).

L'elenco dettagliato dei rischi identificati, con relative strategie di mitigazione, è riportato nel *Piano di Progetto*.

## 5.9 Diario di Bordo

Il Diario di Bordo è un'attività settimanale con funzione formativa, finalizzata a esporre e discutere lo stato di avanzamento del progetto.

### 5.9.1 Finalità

L'attività ha lo scopo di:

- comunicare le difficoltà incontrate durante lo sprint;
- esporre dubbi e incertezze su come procedere;
- favorire la discussione collettiva e il confronto con il docente.

La cadenza settimanale garantisce che la quantità di argomenti da trattare rimanga gestibile e permetta un intervento tempestivo sulle problematiche emerse.

### 5.9.2 Modalità di svolgimento

Il Diario di Bordo si svolge secondo le seguenti modalità:

- il gruppo prepara 2-3 diapositive di contenuto per un'esposizione di massimo 4 minuti;
- l'esposizione è tenuta dal Responsabile di Progetto in carica nel periodo corrente;
- la presentazione avviene condividendo lo schermo nella stanza Zoom designata;
- al termine delle esposizioni di tutti i gruppi, segue una discussione collettiva sui punti più rilevanti.

### 5.9.3 Contenuti della presentazione

Le diapositive devono includere:

- sintesi delle attività svolte nel periodo;
- difficoltà tecniche o organizzative incontrate;
- dubbi aperti e richieste di chiarimento;
- stato di avanzamento rispetto alla pianificazione.

## 5.10 Progettazione dell'interfaccia utente

Per la progettazione dell'interfaccia utente il gruppo utilizza strumenti di prototipazione che consentono di definire l'aspetto e il comportamento del prodotto prima dell'implementazione.

### 5.10.1 Strumenti adottati

Il gruppo utilizza **Penpot<sub>G</sub>**, una piattaforma open source per la creazione di mockup e prototipi interattivi. La scelta è motivata dalla natura collaborativa dello strumento e dalla sua disponibilità gratuita.

### 5.10.2 Processo di design

Il processo di progettazione dell'interfaccia si articola nelle seguenti fasi:

- **Creazione dei mockup:** definizione delle schermate principali e dei flussi di navigazione.
- **Revisione interna:** discussione dei mockup durante le riunioni di sprint per raccogliere feedback.
- **Validazione con il proponente:** presentazione dei prototipi al proponente per conferma o richieste di modifica.
- **Passaggio all'implementazione:** i mockup approvati diventano riferimento per lo sviluppo del frontend.