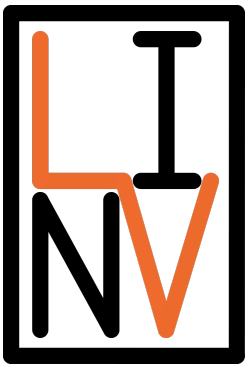

LINV Team

Way of Working

Progetto di Ingegneria del Software
A.A. 2022/2023

Informazioni	
Versione	2.0
Uso	Interno
Data	29/06/2023
Destinatari	LINV Team
Responsabile	Matteo Cusin
Amministratore	Riccardo Rossi
Verificatori	Alessandro Baldissera
	Mauro Carnuccio
	Alberto Casado Moreno
	Matteo Cusin
	Nicola Ravagnan
	Riccardo Rossi
	Alessandro Santin
Redattori	Alessandro Baldissera
	Mauro Carnuccio
	Alberto Casado Moreno
	Matteo Cusin
	Nicola Ravagnan
	Riccardo Rossi
	Alessandro Santin

Indice

Elenco delle figure

Elenco delle tabelle

Registro delle modifiche

Ver.	Data	Autore	Ruolo	Verificatore	Descrizione
2.0	29/06/2023	Matteo Cusin	Responsabile		Approvazione documento
1.9	14/05/2023	Alberto Casado Moreno	Amministratore	Matteo Cusin	Ristrutturazione
1.8	05/05/2023	Alberto Casado Moreno	Amministratore	Alessandro Baldissera	Aggiornate regole di codifica
1.7	26/04/2023	Alberto Casado Moreno	Amministratore	Matteo Cusin	Scrittura norme processi di sviluppo: progettazione
1.6	24/04/2023	Matteo Cusin	Amministratore	Riccardo Rossi	Aggiornamento norme verbali
1.5	14/04/2023	Alessandro Santin	Amministratore	Nicola Ravagnan	Aggiunta sezione Strumenti
1.4	14/04/2023	Alessandro Santin	Amministratore	Nicola Ravagnan	Aggiornamento norme di pianificazione, preventivo e consuntivo periodo
1.3	13/04/2023	Alessandro Santin	Amministratore	Nicola Ravagnan	Aggiornamento norme di approvazione verbali esterni
1.2	13/04/2023	Alessandro Santin	Amministratore	Nicola Ravagnan	Aggiornamento norme numero di versione
1.1	11/04/2023	Alberto Casado Moreno	Amministratore	Matteo Cusin	Aggiornamento del changelog
1.0.0	11/03/2023	Alessandro Santin	Responsabile		Approvazione del documento
0.9.1	09/03/2023	Alessandro Baldissera	Amministratore	N/A	Correzione generale del documento
0.8.1	07/01/2023	Mauro Carnuccio	Amministratore	N/A	Aggiunto processo di sviluppo Analisi dei Requisiti

0.7.2	07/01/2023	Mauro Carnuccio	Amministratore	N/A	Aggiunto processo di fornitura
0.7.1	06/01/2023	Mauro Carnuccio	Amministratore	N/A	Aggiunta codifica ai processi di sviluppo
0.6.1	05/01/2023	Matteo Cusin	Analista	N/A	Aggiunta sezione relativa allo script cleanup.sh
0.5.1	26/12/2022	Matteo Cusin	Amministratore	N/A	Aggiornata struttura file, aggiunto schema di flusso delle attività
0.4.2	26/12/2022	Matteo Cusin	Amministratore	N/A	Aggiunti ruoli di progetto, normato incontro interno settimanale
0.4.1	21/12/2022	Matteo Cusin	Amministratore	N/A	Aggiornata la sezione di organizzazione delle attività
0.3.2	19/12/2022	Matteo Cusin	Amministratore	N/A	Aggiornati riferimenti al glossario
0.3.1	16/12/2022	Matteo Cusin	Amministratore	N/A	Aggiornata la sezione delle regole tipografiche
0.2.1	11/12/2022	Nicola Ravagnan	Amministratore	N/A	Sostituito Github Projects con Jira, aggiunta sezione tracciamento ore e altre regole di scrittura
0.1.1	09/11/2022	Alessandro Baldissera	Amministratore	N/A	Aggiornate le norme sul registro delle modifiche e sul numero di versione
0.0.3	05/11/2022	Alessandro Baldissera	Amministratore	N/A	Aggiunta della sezione di verifica, automazione, accettazione e processi organizzativi

0.0.2	31/10/2022	Alessandro Baldissera	Amministratore	N/A	Aggiunta della sezione di stesura della documentazione
0.0.1	28/10/2022	Alessandro Baldissera	Amministratore	N/A	Prima versione preliminare

1 Introduzione

1.1 Scopo del documento

Lo scopo del documento è quello di definire regole e "best practices" che ogni membro del gruppo si impegna a rispettare per raggiungere in modo efficiente ed efficace la realizzazione del prodotto finale.
Vengono inoltre specificate delle convenzioni sull'uso dei vari strumenti scelti per lo sviluppo del prodotto.

1.2 Glossario

Questo documento, come tutti gli altri stilati durante la realizzazione del progetto, è corredato da un ***Glossario***, che si può trovare allegato alla documentazione, nel quale si definiscono tutti i termini specifici al progetto o di significato ambiguo.
Quando un termine è definito nel ***Glossario*** si trova una G a pedice del termine stesso.

1.2.1 Raccolta termini del glossario

I termini del ***Glossario*** verranno raccolti tramite un documento Google Docs$_G$ condiviso tra i membri del gruppo dove vengono trascritti termini di rilevante interesse nel dominio del progetto in una checklist.
Questi termini in un momento successivo saranno approfonditi nella stesura del ***Glossario*** da parte dell'*Amministratore*.

1.3 Riferimenti

1.3.1 Riferimenti normativi

- ***Way of Working***;
- Regolamento del progetto didattico:
  https://www.math.unipd.it/~tullio/IS-1/2022/Dispense/PD02.pdf.

1.3.2 Riferimenti informativi

- Capitolato C5:
  https://www.math.unipd.it/~tullio/IS-1/2022/Progetto/C5.pdf;
- ***Verbali interni***;
- ***Verbali esterni***;
- I processi di ciclo di vita del software:
  https://www.math.unipd.it/~tullio/IS-1/2022/Dispense/T02.pdf;
- Gestione di progetto:
  https://www.math.unipd.it/~tullio/IS-1/2022/Dispense/T04.pdf;
- Amministrazione di progetto:
  https://www.math.unipd.it/~tullio/IS-1/2022/Dispense/FC2.pdf.

2 Processi primari

Il seguente capitolo descrive i processi adottati dal gruppo relativamente alle operazioni di fornitura e sviluppo del prodotto.

2.1 Fornitura

Il processo di fornitura ha lo scopo di determinare i compiti, le attività e le risorse necessarie allo svolgimento del progetto nel suo complesso soddisfacendo le richieste del *Proponente*.

Aspettative Le aspettative del processo di fornitura sono:

- Determinare i bisogni che il prodotto dovrà soddisfare;
- Chiarire i dubbi e stabilire i vincoli con il *Proponente*;
- Stabilire i tempi di lavoro;
- Stimare i costi;
- Ottenere feedback da parte del *Proponente* e dei committenti riguardo al lavoro svolto.

Piano di Qualifica
Questo documento descrive le norme$_G$ necessarie per garantire la qualità dei prodotti e dei processi messi in atto dal gruppo.
È diviso nelle seguenti sezioni:

- Qualità di processo;
- Qualità di prodotto;
- Specifica dei test;
- Resoconto delle attività di verifica.

Piano di Progetto
Questo documento viene mantenuto durante tutta la durata del progetto.
Contiene le seguenti sezioni:

- Stima dei costi di realizzazione;
- Rischi attesi e loro mitigazione.

Per ogni periodo di avanzamento del progetto si trovano le seguenti sezioni:

- Pianificazione del lavoro;
- Preventivo delle ore e dei costi;
- Consuntivo delle ore e dei costi.

Tali sezioni vengono realizzate in base alle seguenti norme guida:

- Pianificazione del lavoro:
  - Nel realizzare la pianificazione di un periodo, il *Responsabile* stabilisce una serie di obiettivi da raggiungere all'interno dello stesso, sulla base della loro importanza rispetto allo stato del progetto e realizzabilità nel tempo disponibile.
    Da questi obiettivi macroscopici sono derivate attività concrete, visualizzate in un diagramma di Gantt, disposte in relazione tra loro all'interno della durata temporale del periodo.
- Preventivo delle ore e dei costi:
  - In base alle disponibilità individuali e in relazione a quanto individuato nella fase di pianificazione, a ogni membro è associato un carico di ore lavorative, tracciato e diviso per ruolo, tramite il quale si ricava una stima dei costi per il periodo.
    Tale stima viene visualizzata tramite grafici quantitativi delle ore e dei costi rispetto al budget totale fornendo proiezioni rispetto a quella che si prevede essere la situazione a fine periodo.
- Consuntivo delle ore e dei costi:
  - Al termine del periodo, il *Responsabile* confronta il lavoro svolto con quanto pianificato, sia da un punto di vista contabile sia tenendo conto degli obiettivi prefissati.
    Viene effettuato un confronto tra il preventivo ore e costi realizzato ad inizio periodo e quanto riportato al suo termine, utilizzato poi per aggiornare lo stato attuale del budget e monte ore utile al preventivo successivo;
  - Viene effettuata collettivamente una riflessione sull'esito del periodo, sia per quanto riguarda gli aspetti quantitativi (ore e budget preventivato rispetto all'effettivo consumo) sia qualitativi (stato di avanzamento degli obiettivi perseguiti e delle relative attività rispetto a quanto previsto).
    Tale analisi è utile al fine di accertare che a un consumo di risorse corrisponda un adeguato avanzamento negli obiettivi a lungo termine del progetto.

2.1.1 Strumenti

Gli strumenti da utilizzare per produrre i documenti precedentemente elencati sono:

- **Google Sheets**: utilizzato per annotare le ore, calcolare i costi dei vari periodi e per produrre i relativi grafici;
- **Online Gantt**: utilizzato per creare i diagrammi di Gantt.

2.2 Sviluppo

Il processo di sviluppo ha lo scopo di descrivere le attività di analisi, progettazione, codifica, test, installazione e accettazione del prodotto software da sviluppare.

Aspettative Le aspettative del processo di sviluppo sono:

- Determinare i requisiti del prodotto e relativi vincoli;
- Progettare il prodotto seguendo i vincoli di design identificati;
- Realizzare il prodotto finale, superando tutti i test identificati e rispettando le aspettative del *Committente*.

2.2.1 *Analisi dei Requisiti*

L'*Analisi dei Requisiti* ha lo scopo di identificare i requisiti obbligatori e desiderabili dei prodotti richiesti dal *Proponente* attraverso uno studio approfondito del capitolato, tramite le seguenti procedure:

- *Studio iniziale*: viene effettuata un analisi del capitolato;
- *Discussione con il Proponente*: nel caso sorgessero perplessità si organizza un incontro con il *Proponente* per chiarire i dubbi;
- *Individuazione dei casi d'uso*: vengono messi per iscritto le funzionalità del prodotto tramite i casi d'uso.

Aspettative Il risultato di questa attività è la creazione di un documento che contenga tutti i requisiti richiesti dal *Proponente* anche dal punto di vista dell'utente.
Deve essere presente una sezione di tracciamento tra requisiti e casi d'uso.

Casi d'uso Ciascun caso d'uso definisce uno scenario in cui ci sono interazioni con il sistema da parte di uno o più attori.
Il codice dei casi d'uso è definito così:

UC[Applicazione]-[Numero caso d'uso].[Sottocaso] [Titolo]

In cui:

- **UC**: acronimo di "Use Case";
- **Applicazione**: identifica a quale applicazione si riferisce il caso d'uso:
  - **V**: *SmartLogViewer*;
  - **S**: *SmartLogStatistics*.

Un caso d'uso è composto da:

- Descrizione;
- Scenario;
- Attore principale;
- Attore secondario;
- Estensioni;

- Inclusioni;
- Precondizioni;
- Postcondizioni;
- Generalizzazioni.

Requisiti I requisiti vengono ricavati nei seguenti modi:

- Analisi approfondita del capitolato;
- Discussioni fra i componenti del gruppo;
- Confronto con il *Proponente*.

Il codice dei requisiti è definito così:

R[Tipologia][Applicazione]-[Importanza].[Caso d'uso relativo].[Sottocaso]

In cui:

- **R**: acronimo di "Requisito";
- **Tipologia**: tipologia del requisito:
  - **F**: Funzionale;
  - **Q**: Qualità;
  - **V**: Vincolo.
- **Applicazione**: identifica a quale applicazione si riferisce il requisito:
  - **V**: *SmartLogViewer*;
  - **S**: *SmartLogStatistics*.
- **Importanza**: identifica l'importanza del requisito:
  - **1**: Obbligatorio;
  - **2**: Desiderabile;
  - **3**: Opzionale.

Strumenti Gli strumenti da utilizzare per produrre i documenti precedentemente elencati sono:

- **StarUML**: utilizzato per creare i diagrammi dei casi d'uso utilizzando il linguaggio *UML*.

2.2.2 Progettazione

Questa attività ha lo scopo di definire l'architettura, la struttura, i compiti e le funzionalità del sistema software in modo da garantire la soddisfazione dei requisiti del cliente, la facile manutenzione e l'espandibilità futura del sistema, seguendo le seguenti procedure:

- Funzionalità: vengono stilate le funzionalità basate sulle richieste del *Proponente*;
- Architettura: vengono create l'architettura e lo schema delle classi del prodotto;
- Verifica: i *verificatori* controllano l'adeguatezza della progettazione prima che essa venga implementata.

Aspettative Le aspettative delle attività di progettazione sono:

- Determinare le componenti dell'architettura di sistema;
- Comprendere a fondo le interazioni che avvengono tra le componenti sopra citate;
- Garantire che le componenti siano coerenti con gli obiettivi funzionali, prestazionali e di sicurezza tracciando i requisiti di sistema alle relative componenti.

Tramite una dimostrazione prototipale sviluppata durante le attività di *Analisi dei Requisiti* (detta *Proof of Concept*), tali aspettative possono essere meglio comprese.

Strumenti Gli strumenti da utilizzare per produrre i documenti precedentemente elencati sono:

- **StarUML**: utilizzato per creare i diagrammi delle classi utilizzando il linguaggio *UML*.

2.2.3 Codifica

Scopo Questa attività ha lo scopo concretizzare il risultato del processo di progettazione.

Aspettative Il risultato di questa attività sarà un prodotto software che rispetti tutti i requisiti richiesti dal *Proponente*.
Devono essere identificati dei linguaggi e delle tecnologie da utilizzare, oltre che degli strumenti per lo sviluppo.
Il codice prodotto dovrà rispettare delle regole, elencate in seguito, che permetteranno maggiore coerenza e leggibilità nelle varie parti del codice nonostante la scrittura possa essere eseguita da persone diverse.

Procedure Dopo aver svolto l' `Analisi dei Requisiti` e la `Progettazione`, vengono eseguite le seguenti procedure per completare la `Codifica`.

- *Implementazione*: coloro i quali abbiano ruolo di *Programmatore* dovranno scrivere il codice utilizzando le norme descritte in questo documento;
- *Test*: il codice scritto viene testato per assicurarsi che si comporti come previsto;
- *Verifica del codice*: il codice viene verificato dai *verificatori* che si assicureranno segua le norme di progetto e soddisfi i requisiti contenuti nel documento ***Analisi dei Requisiti***.

Regole generali di codifica **Regole per C#**:

- Variabili:
  - Non costanti:
    - Nomi: `camelCase`.
  - Costanti:
    - Nomi: `PascalCase`.
- Metodi:
  - Nomi: `PascalCase`;
  - Allo scopo di rendere il codice più lineare e leggibile, in qualsiasi metodo non possono essere superati i 3 livelli di annidamento.
- Classi:
  - Nomi: `PascalCase`;
  - Quelle che non devono poter essere ereditate devono essere marchiate con la parola chiave `sealed`;
  - È vietato l'utilizzo della keyword `partial`.
- Interfacce:
  - Nomi: `PascalCase`;
  - Secondo la documentazione ufficiale di C# i nomi delle interfacce dovrebbero avere come prefisso `I` a indicare il fatto che sono interfacce.
    Si è deciso di non adottare questa convenzione, di conseguenza i nomi delle interfacce saranno trattati esattamente come i nomi delle classi.
- Proprietà:
  - Nomi: `PascalCase`.

Regole per TypeScript:

- Variabili:
  - Nomi: `camelCase`.
- Metodi:
  - Nomi: `camelCase`;
  - Allo scopo di rendere il codice più lineare e leggibile, in qualsiasi metodo non possono essere superati i 3 livelli di annidamento.
- Classi:
  - Nomi: `PascalCase`;
- Interfacce:
  - Nomi: `PascalCase`;
- Proprietà:
  - Nomi: `camelCase`.

Linguaggi Per la scrittura della parte di **back-end** degli applicativi, verranno impiegate le seguenti tecnologie:

- **ASP.NET**: un framework per C# per la creazione di web app;
- **MSTest**: il framework ufficiale per effettuare i test in C#;
- **Moq**: una libreria C# atta alla creazione di mock delle classi per facilitare lo Unit Testing$_G$;
- **Npgsql**: una libreria per l'accesso a database PostgreSQL;
- **PostgreSQL**: un database relazionale che utilizza il linguaggio SQL per la manipolazione dei dati.

Per quanto riguarda la scrittura della parte di **front-end** degli applicativi, i linguaggi scelti sono HTML, CSS e TypeScript e verranno supportati dalle seguenti tecnologie:

- **Angular**: un framework front-end per applicazioni web;
- **Bootstrap v5.3**: una libreria per lo stile delle pagine;
- **D3.js**: una libreria per la creazione di grafici vettoriali.

Strumenti Come descritto nella sezione Utilizzo dei repository$_G$, per lo sviluppo delle applicazioni il gruppo utilizzerà un repository$_G$ dedicato al solo codice.
Per lo sviluppo delle applicazioni potranno essere utilizzati i seguenti ambienti di sviluppo (la scelta è dettata dalla preferenza individuale di ciascun componente del gruppo, data la loro compatibilità, e dalla loro integrazione con le tecnologie scelte):

- **Rider**: un IDE cross-platform prodotto da *JetBrains* per lo sviluppo di applicazioni C# e ASP.NET;
- **Visual Studio 2022**: un IDE per Windows e Mac prodotto da *Microsoft* per lo sviluppo di applicazioni C# e ASP.NET.

Regole per il testing:

- Ogni requisito deve avere ben definiti tutti i test necessari per determinare se il requisito implementato rispetta le aspettative;
- Tutti i test devono essere correttamente implementati;
- Nel momento in cui un branch$_G$ è pronto per la revisione e l'integrazione, il codice da revisionare deve passare tutti i test relativi alla parte di codice interessata dal ticket$_G$.

3 Processi di supporto

Nella seguente sezione sono descritti tutti i processi di supporto legati allo sviluppo del progetto, dalla documentazione alla verifica di tutte le componenti prodotte.

3.1 Documentazione

Tutte le attività di sviluppo dovranno essere documentate al fine di tenere traccia e consultare in modo semplice e veloce ciò che è stato fatto.
Questa sezione annota tutte le norme$_G$ per una corretta stesura e verifica di tutta la documentazione che verrà prodotta durante lo sviluppo del progetto.

3.1.1 Pianificazione

Il contenuto viene ideato e organizzato logicamente.
Fase di fondamentale importanza per documenti ricchi di contenuti.

Nomi dei file Tutti i documenti devono essere contenuti in una cartella omonima e, se il nome del documento è composto da più parole, si utilizzano le abbreviazioni che si possono trovare nelle tabelle riguardanti i documenti e le revisioni.
Il nome del file deve coincidere col nome del documento per esteso in **PascalCase**.
Se il documento è particolarmente lungo può essere separato in più file contenenti le varie sezioni del documento, il cui nome deve seguire il nome della sezione contenuta.

Identificazione dei destinatari L'identificazione dei destinatari è il primo passo fondamentale nella produzione di un documento.
Occorre capire chi sono i destinatari finali del documento, in modo da poter scrivere il testo utilizzando lessico e registro linguistico adeguati.

Definizione del tipo di intestazione L'intestazione di un documento deve contenere informazioni fondamentali come il nome del documento, i destinatari, la data dell'ultimo aggiornamento, la versione e chi vi ha contribuito.

Pianificazione dei contenuti Durante la pianificazione dei contenuti è importante definire quali informazioni devono essere incluse nel documento e come organizzarle in modo che il testo risulti chiaro e coerente.

Creazione della struttura Con questa procedura si definisce la struttura del documento e l'ordine dei contenuti.
Questo potrebbe variare da documento a documento in base alle procedure precedenti.

3.1.2 Impostazione del lavoro

Viene creato lo scheletro del documento a partire da un template adatto.
Il *Responsabile* crea dei ticket$_G$ su Jira che rappresentano le varie attività da svolgere per stilare le parti del documento; i ticket$_G$ possono essere aggiunti nel backlog$_G$ se non devono essere svolti nel periodo corrente, altrimenti verranno creati nella lista delle attività da fare nel periodo corrente.

Utilizzo dei template Tutti i documenti scritti con LaTeX devono inizialmente essere impostati tramite dei comandi. In particolare, verranno utilizzate le seguenti impostazioni:

- **Documenti generici**: definite nel template `./Template/Documento.tex`;
- **Verbali**: definite nel template `./Template/Verbale.tex`;
- **Lettere**: definite nel template `./Template/Lettera.tex`;
- **Preventivi**: definite nel template `./Template/Preventivo.tex`;
- **Pianificazioni**: definite nel template `./Template/Pianificazione.tex`;
- **Consuntivi**: definite nel template `./Template/Consuntivo.tex`.

Per scrivere un nuovo documento (o sezioni ripetute in esso) basterà copiare il template, eventualmente rinominando il file, e procedere con la scrittura.

Utilizzo intestazione Tutti i documenti devono iniziare con un'opportuna intestazione, posizionata in prima pagina, standardizzata nei comandi contenuti nel file `./Common/Intestazione.` da importare nel file principale del documento utilizzando il comando `\input`.
Le diverse intestazioni sono documentate direttamente nel file `./Common/Intestazione.tex`.
Le intestazioni disponibili sono:

- **Documenti generici**: un'intestazione completa ed esaustiva per i principali documenti (ad es. ***Way of Working***, ***Piano di Progetto***, etc.);
- **Verbali**: un'intestazione adattata alla stesura di un verbale;
- **Lettere**: un'intestazione semplificata per la stesura di lettere (ad es. ***Lettera di Candidatura***, lettere di presentazione alle revisioni).

Utilizzo indice Tutti i documenti generici (a eccezione di verbali e lettere) devono essere provvisti di indice.
Dove necessario deve essere presente un indice delle tabelle (vedi Tabelle) e delle immagini (vedi Immagini).

Utilizzo registro delle modifiche Tutti i documenti generici (a eccezione di verbali e lettere) devono essere provvisti di un **registro delle modifiche** che contiene un riassunto delle modifiche apportate al documento in formato tabellare.
La tabella non deve essere registrata nell'indice delle tabelle ma deve essere inserita nella sezione **registro delle modifiche** subito dopo l'indice del documento.
La tabella deve registrare le seguenti informazioni:

1. **Versione del file**;
2. **Data di rilascio**;
3. **Autore**;
4. **Ruolo**;

5. **Verificatore**;

6. **Descrizione**: un riassunto delle modifiche apportate.

Per creare la tabella nel file `./Common/Intestazione.tex` è stato definito un ambiente LaTeX chiamato `changelog`$_G$ che genera la tabella e un nuovo comando `\changelogline` che crea una nuova riga della tabella.
Per esempio, un registro delle modifiche potrebbe avere un sorgente così strutturato:

```
1  \begin{changelog}
2      \changelogline{1.1}{\data{11}{04}{2023}}{Alberto
         ↪ Casado Moreno}{Amministratore}{Matteo
         ↪ Cusin}{Aggiornamento del changelog}
3  \end{changelog}
```

Utilizzo numeri di versione Il versionamento è una pratica utile sia per i documenti che per i prodotti software.
Nella fattispecie, lo schema per indicare la versione di un prodotto è la seguente: `X.Y`, dove:

- **X** incrementa quando il *Responsabile* accetta il prodotto;
- **Y** incrementa quando un *Verificatore* revisiona con esito positivo il prodotto.

Quando il valore di **X** incrementa fa tornare a zero il valore di **Y**.

3.1.3 Avanzamento di versione

Avviene tramite una richiesta di modifica di versione da parte del *Verificatore* o *Responsabile* e deve essere eseguita prima dell'accettazione della pull request$_G$.
Chi ha effettuato l'ultima modifica registrata nel documento dovrà incrementare di una unità il valore di **X** e azzerare il valore di **Y** se la richiesta arriva dal *Responsabile*, o incrementare di una unità il valore di **Y** se la richiesta arriva dal *Verificatore*.

3.1.4 Scrittura

I membri identificati, in base al ruolo, alla stesura del documento si auto organizzano secondo le regole identificate nella sezione Processi Organizzativi, assegnandosi i ticket$_G$ relativi ai contenuti che ognuno deve stilare.

Strumenti utilizzati Per scrivere i documenti è utilizzato il linguaggio LaTeX tramite i seguenti strumenti:

- **pdfTex** versione `3.141592653-2.6-1.40.24` dalla distribuzione **TexLive** scaricabile dal seguente link: `https://www.latex-project.org/get/#tex-distributions`;
- **Microsoft Visual Studio Code** come editor di testo (d'ora in poi VSC);
- **LaTeX Workshop** versione `v8.29.0` come plug-in di VSC come supporto alla scrittura del documento.

Utilizzo comandi di comodo Nel file `./Common/Goodies.tex` è presente una raccolta di comandi personalizzati per la formattazione o per inserire particolari costrutti ripetuti nei documenti e mantenere separata la parte logica di supporto e formattazione dai contenuti.
Ogni membro del gruppo è libero di inserire un nuovo comando in base alle necessità riscontrate nel file indicato prima.
Le regole per l'inserimento di un nuovo comando sono le seguenti:

- Tutti i nuovi comandi devono poter compilare basandosi solo sui pacchetti inclusi nel template, in caso contrario bisognerà attivarsi per correggere tale problema in modo da evitare possibili fallimenti durante la compilazione del codice;
- Ogni comando inserito deve essere preceduto da una descrizione dettagliata di cosa fa e come si utilizza (sotto forma di commento), fornendo una lista esaustiva degli eventuali parametri che necessita;
- Si aggiunge il comando alla lista riassuntiva, all'inizio del file, con una breve descrizione del comportamento (sempre sotto forma di commento).

Regole tipografiche

- I **titoli** delle sezioni iniziano con la lettera **maiuscola**;
- Gli **elementi degli elenchi** iniziano tutti con la lettera **maiuscola** e terminano con un ';' tranne per l'ultimo elemento che termina con un '.';
- Gli **acronimi** si scrivono utilizzando solo lettere **maiuscole**, tranne nei casi nel quale il nome preveda esplicitamente delle lettere minuscole (ad es. VoIP$_G$);
- I **nomi dei ruoli**, di **applicazioni** e **aziende** si scrivono con l'iniziale **maiuscola** e in **italico** (ad es. *Responsabile*);
- I **nomi dei documenti** e delle revisioni si scrivono per esteso, in italico, in grassetto e con le iniziali maiuscole, tranne per le preposizioni (ad es. ***Piano di Progetto***);
- I **nomi dei documenti** e delle **revisioni** che si ripetono spesso possono essere abbreviati seguendo la tabella presentata nella sezione Nomi dei documenti abbreviati, mantenendo sempre la regola del carattere italico e grassetto;
- I **nomi propri** dei componenti del gruppo vanno scritti per esteso, con il nome che precede il cognome e in italico;
- I **nomi propri** dei componenti del gruppo nelle intestazioni e nel changelog$_G$ vengono scritti per esteso ma non in italico;
- I **nomi dei file** vanno scritti in carattere mono spaziato con nome del file ed estensione (ad es. `./Common/Intestazione.tex`);
- Le **date** vengono scritte nel formato GG/MM/AAAA;
- I **numeri razionali** si scrivono utilizzando la virgola come separatore tra parte intera e parte decimale.

Utilizzo stili di testo Il testo nei documenti può essere enfatizzato utilizzando i seguenti stili:

- **Grassetto**: per parole ritenute importanti;
- **Mono spaziato**: per link esterni ai documenti e snippet di codice;
- **Sottolineato**: per riferimenti contenuti nel documento.

Utilizzo nomi dei documenti e revisioni abbreviati Nei documenti in cui vi sono molte ripetizioni di nomi di documenti si possono utilizzare le abbreviazioni che si trovano nelle seguenti tabelle:

Nome documento	Nome abbreviato
Glossario	*G*
Manuale Utente	*MU*
Manuale Sviluppatore	*MS*
Piano di Progetto	*PdP*
Piano di Qualifica	*PdQ*
Specifica Tecnica	*ST*
Way of Working	*WoW*

Tabella 2: Tabella riassuntiva delle abbreviazioni dei nomi dei documenti

Nome revisione	Nome abbreviato
Requirements and Technology Baseline	*RTB*
Product Baseline	*PB*
Customer Acceptance	*CA*

Tabella 3: Tabella riassuntiva delle abbreviazioni dei nomi delle revisioni

Utilizzo tabelle Tutte le tabelle che si inseriscono nel documento devono essere provviste di descrizione tramite il comando `\caption` e, quando nel documento si trova almeno una tabella, il documento deve essere provvisto di indice delle tabelle.
Devono inoltre essere rispettati i seguenti stili di base:

- Le righe e colonne di intestazione devono avere un colore diverso dalle righe e colonne che contengono informazioni;
- Devono essere centrate sul foglio;
- Non devono essere di tipo 'flottante', in modo che la tabella venga inserita nel punto nel quale è stata inserita nel codice sorgente;
- La descrizione (o 'caption') della tabella deve trovarsi sotto la stessa.

Utilizzo immagini Tutte le immagini che si inseriscono nel documento devono essere provviste di descrizione tramite il comando `\caption` e, quando nel documento è presente almeno un'immagine, il documento deve essere provvisto di indice delle immagini.
Devono inoltre essere rispettati i seguenti stili di base:

- Le figure devono essere centrate nel foglio;
- Non devono essere di tipo 'flottante', in modo che l'immagine venga inserita nel punto nel quale è stata inserita nel codice sorgente;
- La descrizione (o 'caption') dell'immagine deve trovarsi sotto alla stessa.

Utilizzo riferimenti cliccabili all'interno del documento È possibile inserire dei riferimenti cliccabili all'interno del documento per una migliore navigabilità.
Per fare ciò si utilizzano le `\label`$_G$ come segnaposto e `\hyperref` per creare un link.
I link devono seguire le seguenti regole:

- Devono essere sottolineati;
- Non devono avere un colore diverso dal resto del testo;
- Non devono usare font mono spaziato in quanto riservato ai link a risorse esterne.

3.2 Verifica

Tutto ciò che viene prodotto dal team, dalla documentazione al software, deve essere verificato e validato da un membro del gruppo differente da colui che ha creato il prodotto.
La verifica permette di non introdurre modifiche al prodotto che possano portare ad una regressione o uno stagnamento dello stesso.
Il gruppo si doterà sia di **verifiche manuali**, svolte dai *Verificatori*, sia di verifiche automatiche quali **Test di unità**$_G$, **Test di integrazione**$_G$, **Test di sistema**$_G$, **Test di non regressione**$_G$ e ogni altra verifica automatica considerata utile.

3.2.1 Verifica dei documenti

Quando un prodotto può essere verificato I prodotti possono essere verificati quando la persona identificata ad apportarne modifiche ha terminato le issues$_G$ relative all'attività e ritiene di aver effettuato un buon lavoro.
In tale momento l'assegnatario può aprire una **pull request**$_G$ dal branch$_G$ della feature$_G$ al branch$_G$ **devel**, il ticket$_G$ passerà automaticamente allo stato **Da revisionare**.
A quel punto un *Verificatore* si incarica di verificare il ticket$_G$ spostandolo nella colonna **In revisione**.

La verifica dei documenti deve assicurarsi che:

- Il documento prodotto segua la struttura decisa durante la sua progettazione;
- Tutti i contenuti siano consoni;
- Vengano rispettate le regole definite per la stesura dei documenti (vedi sezione Documentazione);

- Non ci siano errori grammaticali;
- La compilazione dei file sorgenti del documento abbia successo;
- Il documento finale non presenti artefatti grafici che ne precludano la consultazione.

Per ogni ticket_G di documentazione, la checklist di verifica è la seguente:

- Controllo ortografico del documento;
- Controllo della sintassi del documento;
- Controllo che i contenuti rispettino le aspettative;
- Controllo che i termini presenti nel ***Glossario*** siano marcati correttamente;
- Controllo che tutti i contenuti rispettino le regole tipografiche identificate.

La checklist può comunque essere ampliata in base alle specifiche necessità, indicando gli ulteriori punti di verifica nella descrizione del ticket_G stesso.

Verifica del documento *Analisi dei requisiti*

Il documento ***Analisi dei requisiti*** richiede oltre, alla checklist vista nella sezione precedente, alcuni ulteriori passi:

- Tutti i casi d'uso identificati devono essere rimandabili a un requisito identificato nel capitolato;
- Tutti i casi d'uso identificati devono essere forniti di almeno:
  - Titolo;
  - Descrizione;
  - Scenario;
  - Attore principale;
  - Precondizioni;
  - Postcondizioni.
- Ogni requisito identificato deve comparire nella tabella di tracciamento.

Verifica del documento *Piano di Qualifica*

Il documento ***Piano di Qualifica*** richiede, oltre alla checklist vista nella sezione precedente, alcuni ulteriori passi:

- Tutti i test identificati devono essere rimandabili a un requisito identificato nel capitolato;
- Tutti i requisiti devono essere adeguatamente testati.

Verifica del documento *Piano di Progetto*
Il documento ***Piano di Progetto*** richiede, oltre alla checklist vista nella sezione precedente, alcuni ulteriori passi:

- Le ore e il budget rimanente devono essere corretti (un errore potrebbe propagarsi a cascata nei periodi successivi).

3.2.2 Verifica del codice sorgente

Quando un'attività viene creata, si apre una pull request$_G$ verso il branch$_G$ **devel** e il *Verificatore* può iniziare la fase di verifica, composta dalle seguenti fasi:

1. Controllo che tutte le verifiche automatiche diano esito positivo; per fare ciò, si apre la pull request$_G$ su GitHub$_G$ e, nella sezione **Check**$_G$, i test non devono dare esito negativo;
2. Controllo manuale del sorgente nella sezione **Files changed**$_G$ avviando una revisione, assicurandosi che i cambiamenti rispettino le regole definite e lasciando dei commenti, evidenziando le righe di codice di riferimento se qualcosa non è conforme o genera dei dubbi; nel caso un file non presenti problemi può essere indicato come **Viewed**$_G$ nella revisione;
3. Infine, se le due fasi precedenti vengono completate senza riscontrare problemi da risolvere, la pull request$_G$ può essere accettata e integrata.

3.3 Automazione

L'automazione è un'attività molto importante nella realizzazione efficiente ed efficace di un progetto.
Permette di rendere automatico o semiautomatico un compito ripetitivo e/o tedioso, indifferentemente dalla difficoltà di tale compito.
Consente inoltre di eliminare il fattore umano, limitando la generazione di errori.

3.3.1 Automazione project board di Jira

Il software Jira consente di automatizzare i cambiamenti di stato delle attività di progetto basandosi su azioni compiute sui ticket$_G$ o sugli epic$_G$ relativi.
L'automazione è resa disponibile in Jira$_G$ tramite una funzionalità (configurabile) che svincola l'utente dall'azione di spostare manualmente un ticket$_G$ all'interno della board per farlo cambiare di stato: si abbassa così il rischio di avere incongruenza tra lo stato effettivo del progetto e ciò che è rappresentato tramite Jira$_G$.
La creazione e automazione della board del software Jira$_G$ è avvenuta seguendo i seguenti passaggi:

- **Creazione colonne**: vengono ideati i possibili stati di un'attività, ovvero `Da completare`, `In corso`, `Da revisionare`, `In revisione`, `In ritardo` e `Completato`;
- **Creazione workflow**: vengono indicati i possibili cambiamenti di stato e le condizioni che li fanno avvenire automaticamente;
- **Creazione azioni**: vengono create le automazioni per far si che un'attività possa mutare di stato in risposta a eventi esterni, principalmente avvenuti tramite GIT$_G$.

Automazione board
Le situazioni in cui l'automazione offerta da Jira$_G$ è stata modellata sono le seguenti:

- Il passaggio di un ticket$_G$ dallo stato **Da completare** a **In corso** aggiorna i campi **Assegnatario** e **Realizzatore**;
- L'apertura di una pull request$_G$ fa passare un ticket$_G$ da **In corso** a **Da revisionare**;
- La presa in carico di un *Verificatore* fa avanzare il ticket$_G$ nello stato **In revisione** e aggiornare il campo **Verificatore**;
- La chiusura della pull request$_G$ fa completare il ticket$_G$;
- I ticket$_G$ in ritardo finiranno nella colonna **In ritardo** in qualunque stato essi si trovino.

3.3.2 Automazione con script

Lo 'scripting' è un'attività di scrittura di codice utilizzata per automatizzare un'attività. Normalmente si crea uno script$_G$ per creare uno strumento estremamente flessibile e adattato allo use case preso in considerazione, rispetto a uno strumento già esistente.
Il gruppo non mette dei vincoli sul linguaggio da utilizzare per creare lo script$_G$, tuttavia utilizzare linguaggi e strumenti di base permette a tutti di poter utilizzare lo script$_G$ senza nessuna difficoltà.
In ogni caso, chi crea uno script$_G$ deve tener presente che:

- Va documentato, annotando il codice e aggiornando l'apposita sezione Script custom;
- Il codice deve essere inserito nel repository$_G$ e quindi versionato;
- Lo script$_G$ deve essere specifico per una precisa attività, quindi deve avere dimensioni ridotte.

Documentazione di uno script Per documentare un nuovo script$_G$ si deve:

- Indicare su quale repository$_G$ si trova;
- Indicare il nome dello script$_G$;
- Descriverne il funzionamento e l'utilizzo, specificando ove necessario la lista dei parametri.

3.3.3 Automazione con GitHub Actions

Le GitHub Actions$_G$ sono il sistema di automazione proprietario di GitHub$_G$.
Esse permettono di eseguire script$_G$ di shell o software (forniti da uno store interno) per eseguire attività di ogni genere in modo automatizzato a eventi specifici che accadono nel repository$_G$.
Ogni membro del gruppo può creare una nuova Action$_G$ a patto che venga documentata in questo documento nella sezione dedicata GitHub Actions$_G$.

Documentazione di una GitHub Action$_G$ Per documentare una nuova GitHub Action$_G$ si deve:

- Indicare su quale repository$_G$ si trova;
- Indicare il nome della Action$_G$;
- Descriverne il funzionamento ed eventualmente come modificarla.

3.3.4 Creazione notifiche automatiche

Per tenere sempre aggiornati tutti i membri del gruppo sugli eventi che accadono nei repository$_G$ di progetto, è attivo il sistema di notifiche automatico di GitHub$_G$ che, attraverso un webhook$_G$, invierà un messaggio in un canale dedicato su Discord$_G$.
Queste notifiche riguarderanno

- Nuovi commit$_G$;
- Apertura di pull request$_G$;
- Revisioni del codice;
- Risultati sul sistema di GitHub Actions$_G$.

Per quanto riguarda l'aggiornamento dei ticket$_G$ di Jira, vengono utilizzate le notifiche tramite E-Mail: in questo caso verranno notificati solo i diretti interessati, cioè tutti i membri che hanno interagito attivamente con il ticket$_G$.

3.4 Accettazione

3.4.1 Scopo

Lo scopo dell'attività di accettazione è rendere consapevole il *Responsabile* del prodotto che si sta rilasciando al *Committente* e/o al *Proponente*.

3.4.2 Quando si attiva l'Accettazione

L'accettazione si attiva nel momento in cui i *Verificatori* ritengono che le attività svolte e integrate nel branch$_G$ **devel** dei repository$_G$ si trovino in uno stato sufficientemente maturo e coerente per essere presentate al *Committente*.
Un altro momento in cui si deve attivare la fase di accettazione è allo scadere prossimo di un ticket$_G$ che prevede la pubblicazione del materiale prodotto.

3.4.3 Come attivare l'Accettazione

Per attivare l'Accettazione, i *Verificatori* devono aprire una pull request$_G$ su GitHub$_G$ dal branch$_G$ **devel** verso il branch$_G$ **main**.

3.4.4 Regole di accettazione dei prodotti

Il *Responsabile* deve assicurarsi che:

- I prodotti, visti come black-box$_G$, siano coerenti con gli obbiettivi previsti;
- I prodotti rispettino i vincoli e i requisiti obbligatori imposti dal *Committente*.

3.5 Rilascio

Lo scopo della fase di rilascio è rendere disponibile a chiunque (in particolare al *Committente* e al *Proponente*) una baseline approvata del prodotto: la pubblicazione della baseline avviene in un sito GitHub Pages$_G$ che si basa sul repository$_G$ contenente la documentazione pubblica.

3.5.1 Attivazione

Quando si attiva il rilascio L'attivazione del rilascio ha come precondizione l'esecuzione del processo di accettazione sopra descritto: tale processo deve dare esito positivo, ovvero il *Responsabile* deve dare conferma della bontà del prodotto, chiudendo la pull request$_G$ verso il branch$_G$ **main**.

Come attivare il rilascio Per attivare il rilascio è necessaria l'accettazione del prodotto da parte del *Responsabile*, con conseguente **chiusura** della pull request$_G$ verso il branch$_G$ **main**.

3.5.2 Regole di rilascio della documentazione

Tale processo è stato automatizzato tramite lo script$_G$ custom **docpublisher**.
Per una corretta esecuzione dello script$_G$, nelle **lettere** deve essere utilizzato il comando `\letter` il quale indica che la tipologia del documento è "lettera" (serve per la suddivisione dei contenuti all'interno del sito GitHub Pages$_G$).
Il comando appena citato necessita di un parametro che indica a quale **revisione** è destinata la lettera:

- **Candidatura**;
- **Requirements and Technology Baseline**;
- **Product Baseline**;
- **Customer Acceptance**.

A pubblicazione avvenuta, tutti i documenti saranno disponibili e organizzati a questo link: `https://linvteam.github.io/SmartLog/`.

3.6 Shell

3.6.1 Script custom

Pulizia dai file temporanei di LaTeX

- Repository$_G$: `linvteam/documentazione`
- Nome: `cleanup.ps1`

Descrizione e utilizzo Lo script$_G$ si occupa di rimuovere tutti i file temporanei creati da LaTeX.
Spesso è necessario rimuovere quei file per pulire il repository$_G$ o per far generare il PDF da zero.
Per rendere il processo automatico ed evitare di commettere errori causati dalla distrazione, si è pensato di creare questo script$_G$ che implementa delle funzioni di sicurezza (ad esempio, ignorare la cartella `Common` o `Immagini`).

Per utilizzarlo, necessita dei seguenti parametri:

1. Directory target della pulizia o `--all` per far pulire tutte le cartelle del repository$_G$;
2. `--no-pdf` per ignorare l'eliminazione del file PDF generato da LaTeX;
3. `--recursive` o `-r` per attivare la modalità ricorsiva, rispetta comunque le directory ignorate;
4. `--help` o `-h` per vedere un messaggio di aiuto e una lista delle directory ignorate.

Per aggiungere altre cartelle da ignorare, basta aprire lo script$_G$ e aggiungerle in coda alla lista `$ignoreDirectories`.

Pulizia dai file temporanei di LaTeX per sistemi Linux-based e MacOS

- Repository$_G$: `linvteam/documentazione`
- Nome: `cleanup.sh`

Descrizione e utilizzo Lo script$_G$ è il risultato del processo di porting$_G$ dello script$_G$ cleanup.ps1 per sistemi Linux-based e MacOS; ha come scopo la rimozione dei file di compilazione di documenti scritti in linguaggio LaTeX.
Allo stesso modo dello script$_G$ cleanup.ps1, `cleanup.sh` ignora le cartelle `Common` ed `Immagini` in quanto in esse sono salvati file funzionali alla compilazione di altri documenti.
Per utilizzarlo, necessita dei seguenti parametri:

1. Directory target della pulizia o `--all` per far pulire tutte le cartelle del repository$_G$;
2. `--no-pdf` per ignorare l'eliminazione del file PDF generato da LaTeX;
3. `--recursive` o `-r` per attivare la modalità ricorsiva, rispetta comunque le directory ignorate;
4. `--help` o `-h` per vedere un messaggio di aiuto e una lista delle directory ignorate.

Per aggiungere altre cartelle da ignorare occorre modificare le righe 116 e 183 dello script$_G$.

Pubblicazione automatica dei documenti

- Repository$_G$: `linvteam/documentazione`
- Nome: `docpublisher`
- Eseguibile: `index.js`

Descrizione ed utilizzo Lo script$_G$ si occupa di pubblicare in automatico sul repository$_G$ `linvteam/SmartLog` i documenti prodotti previa approvazione del *Responsabile*.
È stato scelto di eseguire questa operazione in automatico per evitare errori umani, quali dimenticanze o errori di battitura nel processo di pubblicazione.
Lo script$_G$ inizialmente cerca nel repository$_G$ `linvteam/documentazione` i file PDF dei documenti compilati e, sapendo che si trovano nella stessa cartella del codice sorgente con lo stesso nome, andrà anche a leggere il contenuto del file sorgente per estrarne alcune informazioni necessarie alla pubblicazione.
Una volta lette queste informazioni, lo script$_G$ si occuperà di compilare il file `README.md` del repository$_G$ `linvteam/SmartLog`.
Lo script$_G$ in particolare non si occupa né della compilazione né della pubblicazione dei documenti nel repository$_G$, questo compito è delegato alle Github Actions$_G$.
Il sorgente dello script$_G$ è diviso in 4 file:

- `documentFinder.js`: insieme di funzioni che si occupano di individuare i documenti ed estrarne le informazioni necessarie;
- `index.js`: il file principale dello script$_G$ che crea il nuovo file `README.md`;
- `markdownCreator.js`: insieme di funzioni che si occupano di generare del markdown generico;
- `README-template.js`: il template di partenza per il file `README.md`.

Lo script$_G$ eseguirà la ricerca direttamente dalla **directory di lavoro**, di conseguenza il comando da eseguire sarà: `node docpublisher/index.js` dalla root del repository$_G$.
Il risultato sarà il file `OUTPUT.md` da copiare al posto del file `README.md` nel repository$_G$ `linvteam/SmartLog`.
Lo script$_G$ può anche essere eseguito in locale per verificare l'output prima di pubblicare i documenti nel repository$_G$ pubblico.

4 Processi organizzativi

Il seguente capitolo descrive i ruoli di progetto e tutte le modalità di comunicazione interne ed esterne, oltre a presentare le modalità di ricezione di comunicazioni automatiche generate dagli strumenti utilizzati.

4.1 Gestione risorse umane

4.1.1 Rotazione ruoli

Per tutta la durata del progetto, i membri del gruppo assumeranno diversi **ruoli**, ovvero assumeranno le responsabilità e le mansioni di figure professionali in ambito informatico.
La rotazione dei ruoli avviene durante la cerimonia di **pianificazione** del periodo.
La rotazione ruoli avviene seguendo i seguenti punti:

- Valutazione delle disponibilità orarie dei vari membri;
- Valutazione dei ruoli già ricoperti dai vari membri;
- Valutazione della quantità di tempo intercorsa dall'ultima volta in cui un membro ha ricoperto un certo ruolo;
- Valutazione della distribuzione delle ore per ruolo, per far si che le ore di un ruolo siano suddivise equamente tra i membri.

Ogni membro del gruppo potrà assumere all'occorrenza più di un ruolo durante ogni periodo, evitando accuratamente situazioni in cui si potrebbero avere conflitti di interesse (un membro non può verificare delle modifiche al prodotto se ne è stato l'autore).

Ruoli di progetto Le figure professionali disponibili per il progetto sono:

- *Responsabile*: è la figura di riferimento per il progetto e si occupa di:
  - Gestire le relazioni con l'esterno (con gli *stakeholders* in generale);
  - Gestire le risorse (umane, economiche e temporali);
  - Pianificare le attività da svolgere;
  - Preventivare i costi da sostenere;
  - Affrontare scelte chiave relative alle caratteristiche del prodotto;
  - Valutare i rischi delle scelte da effettuare;
  - Eseguire un'opera di retrospettiva in modo continuo nel tempo per migliorare l'economicità (efficienza ed efficacia) delle azioni compiute dal gruppo di lavoro;
  - Redigere il documento ***Piano di Progetto***;
  - Aggiornare i dati dei grafici del ***Piano di Qualifica***.

  Data l'importanza di queste attività, ove necessario (es. durante la retrospettiva) ogni membro del gruppo verrà coinvolto in una discussione collettiva.
- *Amministratore*: è la figura di riferimento per quanto riguarda le norme$_G$ di progetto, ovvero le regole di lavoro che il team deve mettere in atto; si occupa di:

  - Selezionare l'uso delle tecnologie che il gruppo dovrà utilizzare nel progetto;
  - Stabilire i protocolli di comunicazione del gruppo (interni ed esterni);
  - Normare le tecnologie e i processi in uso nel documento ***Way of Working***;
  - Stilare il documento ***PdQ***.

- *Analista*: è la figura di riferimento per quanto riguarda l'attività di **analisi dei requisiti**; si occupa di:
  - Comprendere a fondo i bisogni espressi dal *Proponente* (bisogni "lato utente");
  - Analizzare e scomporre i bisogni "lato utente" fino a ottenere, per fasi successive, i bisogni "lato soluzione": essi sono le specifiche per il prodotto, ovvero rispondono alla domanda "cosa deve fare il prodotto per soddisfare i bisogni dell'utente?";
  - Redigere il documento ***Analisi dei Requisiti***.

- *Progettista*: è la figura di riferimento per quanto riguarda le scelte progettuali derivate dai requisiti "lato soluzione"; si occupa di:
  - Costruire l'architettura di prodotto tramite scelte architetturali su più livelli di specificità (progettazione logica e progettazione di dettaglio) che soddisfino i requisiti "lato soluzione";
  - Assicurarsi che l'architettura sia di qualità, ovvero che sia dotata di caratteristiche che le consentono di avere certe proprietà desiderabili;
  - Tracciare la corrispondenza tra i requisiti del prodotto (elencati nel documento ***Analisi dei Requisiti***) e le scelte progettuali che li soddisfano.

- *Programmatore*: è la figura di riferimento per la codifica del software di prodotto; si occupa di:
  - Seguire le direttive del *Progettista* codificando le varie parti dell'architettura elaborata;
  - Scrivere i test che il codice deve superare per essere considerato "corretto";
  - Tracciare la corrispondenza tra i requisiti del prodotto (elencati nel documento ***Analisi dei Requisiti***) e le unità di codice che li soddisfano.

- *Verificatore*: è la figura di riferimento per il controllo della correttezza del prodotto; si occupa di:
  - Verificare che le modifiche apportate ai prodotti rispettino il livello di qualità stabilito nel documento ***Piano di Qualifica***;
  - Segnalare possibili correzioni all'*Assegnatario* di un compito.

Strumenti Gli strumenti da utilizzare per tenere traccia dei ruoli assegnati sono:

- **Google Sheets**: utilizzato per annotare i ruoli assegnati, le ore svolte e le ore da svolgere.

4.2 Comunicazione

Il gruppo si impegna a mantenere delle comunicazioni attive tra i componenti in modalità sia sincrone che asincrone, a seconda delle necessità.
Anche in questo ambito l'automazione può essere applicata per veicolare informazioni: le misure adottate dal gruppo sono riportate nella sezione Notifiche automatiche.

4.2.1 Comunicazioni sincrone

Comunicazioni interne
Le comunicazioni sincrone interne avverranno nelle seguenti modalità:

- **Presenza**: in incontri organizzati;
- **Discord**$_G$: consente di effettuare chiamate, videochiamate e di condividere il contenuto dello schermo dei partecipanti, consentendo inoltre lo scambio di informazioni testuali e multimediali tramite chat.

Svolgimento di incontri formali interni Durante lo sviluppo del progetto saranno necessari degli incontri sincroni e formali nei quali discutere di diverse tematiche sia interne al gruppo che riguardanti lo sviluppo del progetto stesso.
Per questo motivo gli incontri formali seguiranno le seguenti regole:

1. Il *Responsabile* trova la data migliore per il prossimo incontro chiedendo ai membri la disponibilità;
2. Una volta trovato il giorno e l'ora dell'incontro, il *Responsabile*, nella cartella condivisa su Google Drive$_G$ dedicata ai verbali, crea un Google Doc$_G$ a partire dal template e si compilano i vari campi dell'intestazione;
3. I membri del gruppo sono liberi di inserire i punti di discussione all'interno del documento;
4. Il *Responsabile* organizza i punti di discussione in base all'urgenza e al tempo a disposizione;
5. Il *Responsabile* dirige la discussione seguendo i punti inseriti precedentemente;
6. Lo *Scriba* eletto dal *Responsabile* trascrive per ogni punto di discussione una breve conclusione;
7. Lo *Scriba* ufficializza il verbale formattandolo in LaTeX e caricandolo nel repository$_G$ della documentazione.

Gli incontri devono essere pianificati, possibilmente di settimana in settimana, in modo tale che i vari componenti del gruppo si possano organizzare per essere presenti attivamente all'incontro.

Riunione di aggiornamento e sincronizzazione Indicativamente, ogni venerdì mattina alle ore 09:00 avrà luogo, sulla piattaforma Discord$_G$, la riunione di **aggiornamento e sincronizzazione**: in tale occasione, i membri del gruppo discutono di situazioni accadute successivamente alla riunione precedente e di eventuali incombenze.
L'orario può comunque variare in base alle necessità dei membri del gruppo.
Qualora si presentassero situazioni in cui la riunione settimanale non potesse essere svolta (vincoli temporali, personali o semplicemente per mancanza di utilità in un dato istante) le comunicazioni sui canali asincroni dovranno essere tempestive.

Comunicazioni esterne

Le comunicazioni sincrone esterne avvengono secondo le seguenti modalità:

- **Microsoft Teams**$_G$: incontri in videochiamata con il *Proponente*;
- **Zoom**$_G$: incontri in videochiamata con il *Committente*.

Svolgimento di incontri formali esterni Durante lo sviluppo del progetto saranno necessari diversi incontri con il *Proponente* per discutere e fare il punto della situazione sullo sviluppo del prodotto.
Gli incontri seguiranno le seguenti regole:

1. Il *Proponente* o il *Responsabile* richiedono di organizzare un incontro;
2. Una volta organizzato l'incontro, il *Responsabile*, nella cartella condivisa su Google Drive$_G$ dedicata ai verbali, crea un Google Doc$_G$ a partire dal template e compila i vari campi dell'intestazione;
3. I membri del gruppo sono liberi di inserire i punti di discussione all'interno del documento;
4. Il *Responsabile* organizza i punti di discussione in base all'urgenza e al tempo a disposizione;
5. Il *Responsabile* espone i punti di discussione al *Proponente*, lasciando la parola ai membri del gruppo interessati dove necessario;
6. Lo *Scriba* eletto dal *Responsabile* trascrive per ogni punto di discussione una breve conclusione;
7. Lo *Scriba* ufficializza il verbale formattandolo in LaTeX
8. Il verbale viene inviato all'interessato esterno chiedendo a questi di confermare che quanto in esso riportato corrisponda al vero;
9. Se necessario vengono effettuate eventuali modifiche al verbale fino ad ottenerne l'approvazione;
10. Il verbale con conferma di approvazione riportata al suo interno viene caricato nel repository$_G$ della documentazione.

Strumenti Gli strumenti utilizzati per le comunicazioni sincrone sono i seguenti:

- **Discord**$_G$: utilizzato per le comunicazioni sincrone interne;
- **Microsoft Teams**$_G$: utilizzato per le comunicazioni sincrone esterne;
- **Zoom**$_G$: utilizzato per le comunicazioni sincrone esterne.

4.2.2 Comunicazioni asincrone

Comunicazioni interne

Le comunicazioni asincrone avvengono tramite la piattaforma **Telegram**$_G$, utilizzando un apposito **Gruppo** per comunicazioni che non richiedono risposta immediata e possono essere recepite in qualsiasi momento da tutti i membri del gruppo.
Vi è inoltre la possibilità di comunicazioni dirette e private tra due componenti del gruppo tramite **Chat privata**.

Comunicazioni esterne
Il gruppo è intenzionato a mantenere comunicazioni attive e costanti con il *Proponente* e il *Committente*.
Le comunicazioni avverranno in giorni lavorativi e potranno variare da semplici domande sul capitolato, principalmente tramite il **Gruppo chat Microsoft Teams**$_G$, a comunicazioni più formali, come l'organizzazione di un incontro sincrono, principalmente tramite **E-Mail**.

Strumenti Gli strumenti utilizzati per le comunicazioni asincrone sono i seguenti:

- **Gruppo Telegram**$_G$: utilizzato per le comunicazioni asincrone interne;
- **Chat privata Telegram**$_G$: utilizzato per le comunicazioni asincrone interne;
- **E-Mail**: utilizzato per le comunicazioni asincrone esterne, come l'organizzazione di incontri o revisioni$_G$;
- **Diario di bordo**; utilizzato per comunicare lo stato di avanzamento delle attività al *Committente*;
- **Microsoft Teams**$_G$: utilizzato per le comunicazioni asincrone esterne brevi.

5 Strumenti

La seguente sezione racchiude e descrive, in ordine alfabetico, gli strumenti utilizzati per la realizzazione del progetto, da un punto di vista pratico e organizzativo, e i modi nei quali questi vengono utilizzati.
Viene inoltre indicata per ogni strumento una lista dei processi nei quali è coinvolto.

5.1 Angular

https://angular.io/

Framework front-end open source per sviluppare applicazioni web, basato su TypeScript e sviluppato da Google.
Utilizzato dal gruppo per realizzare il front-end degli applicativi richiesti.
Processi di riferimento:

- Sviluppo.

5.2 ASP.NET

https://dotnet.microsoft.com/en-us/apps/aspnet

Framework open source per la realizzazione di applicazioni web dinamiche sviluppato da Microsoft.
Utilizzato dal gruppo per realizzare il back-end degli applicativi richiesti.
Processi di riferimento:

- Sviluppo.

5.3 Bootstrap v5.3

https://getbootstrap.com/

Framework CSS open source per applicazioni web.
Utilizzato dal gruppo per realizzare il front-end degli applicativi richiesti.
Processi di riferimento:

- Sviluppo.

5.4 D3.js

https://d3js.org/

Libreria JavaScript per la realizzazione di grafici dinamici e interattivi all'interno di un browser.
Utilizzato dal gruppo per la creazione dei grafici negli applicativi richiesti.
Processi di riferimento:

- Sviluppo.

5.5 Discord

https://discord.com/

Piattaforma di VoIP e instant messagging che permette agli utenti di comunicare tramite messaggi testuali, chiamate vocali, e videochiamate.
Utilizzata per comunicazioni interne al gruppo e riunioni.
Processi di riferimento:

- Comunicazioni interne;
- Notifiche automatiche;
- Svolgimento di incontri formali interni.

5.6 Git

https://git-scm.com/

Software di controllo versione distribuito open source.
Utilizzato per tracciare e gestire il versionamento della documentazione e del codice sviluppati nel corso del progetto.

Uso dei branch Nei vari repository$_G$ si utilizzerà una versione di Gitflow semplificata su 3 livelli di branch$_G$:

- **main**: dove verranno convogliate tutte le modifiche per le varie release;
- **devel**: derivato da **main**, dove verranno convogliate tutte le attività sviluppate e confermate dai verificatori;
- Per ogni ticket$_G$ di **Jira** si aprirà un branch$_G$ dedicato derivante da **devel** o dal branch$_G$ dell'epic$_G$ associato (se presente), il nome del branch$_G$ dovrà iniziare con il codice identificativo del ticket$_G$ a cui si vuole contribuire.

Convenzioni sui commit Qualsiasi codice che viene caricato sul repository$_G$ condiviso deve almeno compilare e passare una buona parte dei test scritti.
Nel messaggio di commit$_G$ dovrà essere contenuto l'identificatore del ticket$_G$ a cui si riferisce.
Il messaggio di commit$_G$ dovrà essere breve ma conciso, senza perdersi in dettagli, eventualmente da descrivere nella issue$_G$ relativa all'attività che si sta svolgendo o in altra sede consona.

Commit verificati Ogni commit$_G$ entrante nel repository$_G$ dovrà essere **firmato** tramite chiave SSH: ogni membro del gruppo, dopo aver creato una chiave SSH per ogni device in uso per il progetto, assocerà le chiavi al proprio account GitHub$_G$.
La firma dei commit$_G$ è automatica se preceduta da un'adeguata configurazione del device; GitHub$_G$ garantirà l'autenticazione invertendo la procedura di firma.
Processi di riferimento:

- Sviluppo;

- Documentazione;
- Automazione;
- Utilizzo dei repository.

5.7 GitHub

https://github.com/

Servizio online di hosting e controllo di versione mediante l'uso di Git$_G$.
Utilizzato dal gruppo per raccogliere il codice e la documentazione sviluppati durante il progetto.

Utilizzo dei repository Allo scopo di separare i contenuti (e facilitare la scrittura del ***Manuale Utente***) si utilizzeranno 3 repository$_G$:

- **Documentazione privata**: che conterrà tutti i sorgenti LaTeX dei documenti, oltre alla documentazione interna al team;
- **Documentazione pubblica**: che conterrà tutti i documenti da condividere con il *Committente*;
- **Software**: che conterrà tutto il software prodotto nelle varie fasi di sviluppo.

Processi di riferimento:

- Verifica;
- Automazione;
- Rilascio;
- Comunicazioni interne;
- Notifiche automatiche;
- Utilizzo dei repository.

5.8 GitHub Actions

https://github.com/features/actions

Sistema di automazione proprietario di GitHub.
Permettono di eseguire script di shell o software forniti da uno store interno per eseguire attività in modo automatizzato ad ogni evento che accade nel repository.
Utilizzate come strumento di automazione, per dettagli consultare la sezione apposita.
Processi di riferimento:

- Automazione;
- Notifiche automatiche.

Build e Deploy dei documenti Latex

- Repository$_G$: `linvteam/documentazione`
- Nome: `Build and Deploy Latex document`

Descrizione e modifica Questa GitHub Action$_G$ è composta da due fasi distinte:

- Compilazione automatica dei documenti e caricamento di un archivio `.zip` nell'artifact repository$_G$;
- Caricamento automatico dei documenti prodotti nel repository$_G$ pubblico `linvteam/SmartLog` su approvazione del *Responsabile*.

Per modificare questa Action$_G$ si deve editare il file `.github/workflows/build.yml` andando ad aggiungere il nome del file sorgente del documento (inteso come il 'main' del documento) e il nome del file PDF prodotto nei relativi punti indicati dai commenti che si trovano sul file stesso.
La seconda fase al contrario non è necessario che venga modificata per ogni nuovo documento creato.
La fase di pubblicazione dei documenti si compone di diverse sotto azioni che vengono eseguite solo se la fase precedente dà esito positivo:

- Scaricamento dei file del repository$_G$ `linvteam/documentazione`;
- Scaricamento ed estrazione del file `Documenti.zip` dall'artifact repository$_G$;
- Scaricamento dei file del repository$_G$ `linvteam/SmartLog` in una sotto directory;
- Esecuzione dello script$_G$ `docpublisher`;
- Copia dei nuovi file PDF dei documenti e dell'output dello script$_G$ precedente sul repository$_G$ `linvteam/SmartLog`;
- Commit$_G$ e push delle modifiche sul repository$_G$ `linvteam/SmartLog`.

Questa fase viene eseguita solo quando si esegue un'operazione di push sul branch$_G$ `main` di `linvteam/documentazione`.
Questa GitHub Action$_G$ per funzionare necessita di un Secret$_G$ di nome `UPLOAD_TOKEN` per eseguire il commit$_G$ e push delle modifiche nel repository$_G$ `linvteam/SmartLog`.

5.9 Gmail

`https://www.google.com/gmail/about/`

Servizio di email gratuito fornito da Google, utilizzato per comunicazioni formali con entità esterne al gruppo.
Processi di riferimento:

- Comunicazioni esterne;
- Notifiche automatiche.

5.10 Google Docs

https://www.google.com/docs/about/

Word processor online per la creazione e modifica collaborativa sincrona e asincrona di documenti testuali.
Utilizzato come strumento di supporto agli incontri formali e temporaneo tracciamento delle informazioni prima che queste vengano formalizzate.
Processi di riferimento:

- Comunicazioni interne;
- Comunicazioni esterne;
- Svolgimento di incontri formali interni;
- Svolgimento di incontri formali esterni.

5.11 Google Drive

https://www.google.com/drive/

Servizio di memorizzazione e sincronizzazione file online gestito da Google.
Utilizzato per la raccolta informale di documenti utili e di supporto allo sviluppo, e per contenere le comunicazioni asincrone effettuate con il *Committente* tramite Diari di Bordo.
Processi di riferimento:

- Comunicazioni interne;
- Comunicazioni esterne;
- Svolgimento di incontri formali interni;
- Svolgimento di incontri formali esterni.

5.12 Google Sheets

https://www.google.com/sheets/about/

Servizio di fogli di calcolo online fornito da Google.
Utilizzato per il tracciamento delle ore lavorative e del budget associato e la realizzazione dei relativi grafici.
Processi di riferimento:

- Fornitura;
- Comunicazioni interne;
- Organizzazione delle attività.

5.13 Google Slides

https://www.google.com/slides/about/

Programma online per la realizzazione di presentazioni multimediali fornito da Google.
Utilizzato per la realizzazione dei Diari di Bordo.
Processi di riferimento:

- Comunicazioni esterne.

5.14 JetBrains Rider

https://www.jetbrains.com/rider/

IDE cross-platform .NET prodotto da *JetBrains*.
Utilizzato per la realizzazione degli applicativi richiesti dal progetto.
Processi di riferimento:

- Sviluppo.

5.15 Jira

https://www.atlassian.com/software/jira

Issue tracking system sviluppato da *Atlassian*.
Utilizzato per la gestione interna del progetto, l'associazione di ticket alle attività e il loro tracciamento.
Processi di riferimento:

- Automazione;
- Notifiche automatiche;
- Organizzazione delle attività;
- Utilizzo dei repository.

5.16 LaTeX Workshop

https://github.com/James-Yu/LaTeX-Workshop

Estensione per Visual Studio Code per facilitare la creazione e modifica di documenti tramite LaTeX.
Utilizzata per lavorare alla documentazione realizzata tramite LaTeX.
Processi di riferimento:

- Documentazione.

5.17 Microsoft Teams

https://www.microsoft.com/en-us/microsoft-teams/group-chat-software/

Piattaforma di comunicazione sviluppata da *Microsoft* che permette la comunicazione tramite chat, teleconferenza e condivisione di contenuti.
Utilizzata per le comunicazioni tra il gruppo e il *Proponente*.
Processi di riferimento:

- Comunicazioni esterne;
- Svolgimento di incontri formali esterni.

5.18 Moq

`https://github.com/moq/moq4`

Libreria implementata in C# per il framework .NET per la realizzazione di mock delle classi.
Utilizzata nella creazione di test di unità per facilitare gli stessi.
Processi di riferimento:

- Sviluppo.

5.19 MSTest

`https://github.com/microsoft/testfx`

Framework ufficiale per la realizzazione di test in C#.
Utilizzato per la creazione di test nello sviluppo degli applicativi richiesti.
Processi di riferimento:

- Sviluppo.

5.20 Npgsql

`https://www.npgsql.org/`

Libreria open source per l'accesso di programmi .NET a database PostgreSQL.
Utilizzata per permettere all'applicativo *SmartLogStatistics* l'accesso al database utilizzato.
Processi di riferimento:

- Sviluppo.

5.21 Online Gantt

`https://www.onlinegantt.com`

Servizio online gratuito per la realizzazione di diagrammi di Gantt.
Utilizzato per realizzare i diagrammi di Gantt presenti nella sezione di pianificazione di periodo nel ***Piano di Progetto***.
Processi di riferimento:

- Fornitura.

5.22 PdfTex

https://www.tug.org/applications/pdftex/

Estensione del programma di composizione TeX.
Utilizzata per scrivere la documentazione tramite LaTeX.
Processi di riferimento:

- Documentazione.

5.23 PostgreSQL

https://www.postgresql.org

DBMS ad oggetti open source.
Utilizzato per la creazione del database associato all'applicativo *SmartLogStatistics*.
Processi di riferimento:

- Sviluppo.

5.24 StarUML

https://staruml.io/

Software per la realizzazione di diagrammi aderenti al linguaggio UML.
Utilizzato per creare i diagrammi dei casi d'uso e delle classi dell'architettura di sistema.
Processi di riferimento:

- Documentazione;
- Sviluppo.

5.25 Telegram

https://web.telegram.org/

Strumento di instant messagging cross-platform centralizzato che permette agli utenti di comunicare in chat private o di gruppo.
Utilizzato per comunicazioni interne al gruppo.
Processi di riferimento:

- Comunicazioni interne.

5.26 Visual Studio 2022

https://visualstudio.microsoft.com/

IDE sviluppato da *Microsoft*.
Utilizzato per la realizzazione degli applicativi richiesti dal progetto.
Processi di riferimento:

- Sviluppo.

5.27 Visual Studio Code

https://code.visualstudio.com/

Editor di codice sorgente sviluppato da *Microsoft*.
Utilizzato nella realizzazione tramite LaTeX della documentazione.
Processi di riferimento:

- Documentazione.

5.28 Zoom

https://zoom.us/

Software di videotelefonia per la realizzazione di videconferenze e videochiamate.
Utilizzato per incontri formali con il *Committente*.
Processi di riferimento:

- Comunicazioni esterne;
- Svolgimento di incontri formali esterni.