Norme di Progetto

Gruppo TeamAFK - Progetto "Predire in Grafana"

gruppoafk15@gmail.com

Informazioni sul documento

Versione	1.0.0
Approvatore	Simone Federico Bergamin
Redattori	Olivier Utshudi Davide Zilio Simone Meneghin
Verificatori	Alessandro Canesso Fouad Farid
Uso	Interno
Distribuzione	Prof. Vardanega Tullio Prof. Cardin Riccardo Gruppo AFK

Descrizione

Questo documento racchiude le regole, gli strumenti e le convenzioni adottate dal *teamAFK* durante la realizzazione del progetto *Predire in Grafana*.

Registro delle modifiche

Versione	Data	Descrizione	Nominativo	Ruolo
1.0.0	2020-03-27	Approvazione del documento	Simone Federico Bergamin	*Responsabile di Progetto*
0.4.0	2020-03-27	Quarta ed ultima verifica generale	Fouad Farid	*Verificatore*
0.3.3	2020-03-26	Aggiornamento §2.2 - §2.4	Olivier Utshudi	*Amministratore*
0.3.2	2020-03-26	Aggiornamento §3.10	Davide Zilio	*Amministratore*
0.3.1	2020-03-26	Aggiornamento §4.1 - §4.3	Simone Meneghin	*Amministratore*
0.3.0	2020-03-26	Terza verifica	Fouad Farid	*Verificatore*
0.2.3	2020-03-25	Terminata stesura §3	Davide Zilio	*Amministratore*
0.2.2	2020-03-25	Terminata stesura §4	Simone Meneghin	*Amministratore*
0.2.1	2020-03-25	Terminata stesura §2	Olivier Utshudi	*Amministratore*
0.2.0	2020-03-25	Seconda verifica	Alessandro Canesso	*Verificatore*
0.1.3	2020-03-24	Stesura §2.2 - §2.3.2.2	Olivier Utshudi	*Amministratore*
0.1.2	2020-03-24	Stesura §4.2.2 - §4.3	Simone Meneghin	*Amministratore*
0.1.1	2020-03-24	Stesura §3.6.3 - §3.8.4	Davide Zilio	*Amministratore*
0.1.0	2020-03-24	Prima verifica	Alessandro Canesso	*Verificatore*
0.0.3	2020-03-23	Stesura §1 e §2 - §2.1.5	Olivier Utshudi	*Amministratore*
0.0.2	2020-03-23	Stesura §4 - §4.2.1.2	Simone Meneghin	*Amministratore*
0.0.1	2020-03-23	Stesura §3 - §3.6.2	Davide Zilio	*Amministratore*

Indice

Elenco delle figure

Elenco delle tabelle

1 Introduzione

1.1 Scopo del documento

Il seguente documento ha il compito di stabilire le regole che il team di sviluppo intende seguire in ogni attività di progetto, così da omologare il materiale prodotto. I fornitori intendono adottare un approccio incrementale$_G$, affinché lo sviluppo del prodotto si basi su decisioni prese di comune accordo. Perciò ogni componente del team deve far riferimento a questo documento per garantire la coesione e uniformità delle scelte prese.

1.2 Scopo del prodotto

Il capitolato C4$_G$ illustra il prodotto da fornire. Tale prodotto consiste in tool di addestramento ed un plugin$_G$ di Grafana$_G$ che prenderà il nome *Predire in Grafana*. Entrambi gli applicativi verranno scritti in linguaggio JavaScript$_G$. Il primo avrà il compito di produrre un file di estensione JSON$_G$ basato su dati di addestramento$_G$. Il secondo invece si occuperà di leggere il file creato, effettuare previsioni basandosi su di esso e rendere disponibili al sistema i risultati ottenuti, in modo da poterli visualizzare in grafici e dashboard.

1.3 Glossario

Per evitare ambiguità nei documenti formali, viene fornito il documento *Glossario_v1.0.0*, contenente tutti i termini considerati di difficile comprensione. Perciò nella documentazione fornita ogni vocabolo contenuto in *Glossario_v1.0.0* è contrassegnato dalla lettera *G* a pedice.

1.4 Riferimenti

1.4.0.1 Riferimenti normativi

- **Standard ISO/IEC 12207:1995**: https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf;
- **Capitolato d'appalto C4 - Predire in Grafana**: https://www.math.unipd.it/~tullio/IS-1/2019/Progetto/C4.pdf.

1.4.0.2 Riferimenti informativi

- **Software Engineering - Ian Sommerville - 10th Edition**
- **Slide L05 del corso Ingegneria del Software - Ciclo di vita del software**: https://www.math.unipd.it/~tullio/IS-1/2019/Dispense/L05.pdf;
- **Slide L06 del corso Ingegneria del Software - Gestione di Progetto**: https://www.math.unipd.it/~tullio/IS-1/2019/Dispense/L06.pdf;
- **Slide L12 del corso Ingegneria del Software - Qualità di Prodotto**: https://www.math.unipd.it/~tullio/IS-1/2019/Dispense/L12.pdf;

- **Slide L13 del corso Ingegneria del Software - Qualità di Processo**: https://www.math.unipd.it/~tullio/IS-1/2019/Dispense/L13.pdf;

2 Processi primari

2.1 Fornitura

2.1.1 Scopo

Lo scopo del processo di fornitura consiste nelle attività e compiti dell'acquirente, come risposta ai bisogni del cliente in ambito di sistema, prodotto e/o servizio software. Nello specifico, le caratteristiche richieste dal proponente sono analizzate e stilate nello *Studio di Fattibilità*, il quale è alla base del processo di fornitura. Dopodiché è possibile stabilire le risorse e le procedure necessarie alla redazione di un *Piano di Progetto* da seguire fino alla consegna del materiale prodotto. Nel dettaglio le attività svolte da questo processo sono:

- avvio;
- studio di fattibilità;
- contrattazione;
- progettazione;
- esecuzione e controllo;
- revisione e valutazione;
- consegna e completamento.

2.1.2 Aspettative

Il gruppo deve avere un frequente contatto con il cliente al fine di rispettare i vincoli obbligatori da lui fissati, onde evitare di scostarsi troppo dal prodotto finale richiesto. Per questo è necessario comunicare con il proponente, per poter aver chiari i bisogni che tale prodotto intende soddisfare con le sue funzionalità. Quindi, ogniqualvolta il gruppo venga a contatto con il proponente, deve stabilire:

- aspetti chiave che soddisfano i bisogni richiesti;
- requisiti e vincoli dei processi;
- verifica continua;
- chiarimento di eventuali dubbi;
- accordo sulla qualifica del prodotto.

2.1.3 Descrizione

Questa sezione ha il compito mostrare le norme che *TeamAFK* adotterà in tutte le attività di progettazione, sviluppo e consegna del prodotto *Predire in Grafana*, con lo scopo di diventare fornitori del capitolato proposto dalla proponente *Zucchetti SPA* e dai committenti Prof. Tullio Vardanega e Prof. Riccardo Cardin.

2.1.4 Attività

2.1.4.1 Studio di Fattibilità

Lo *Studio di Fattibilità*, redatto per ogni capitolato$_G$ dagli analisti, illustra:

- **Descrizione generale**: sintesi delle informazioni generali del capitolato e delle circostanze da cui sorgono tali richieste;
- **Obiettivi**: vengono mostrate le caratteristiche principali del prodotto ed i relativi obiettivi da raggiungere;
- **Tecnologie utlizzate**: corrisponde all'elenco degli strumenti messi a disposizione dall'azienda proponente al team per sviluppare il progetto;
- **Valutazione finale**: si tratta di un commento elaborato dal gruppo una volta riunitosi e analizzato le richieste dell'azienda, motivando la scelta di focalizzarsi o meno su una determinata offerta.

2.1.4.2 Piano di Progetto

Il *Piano di Progetto* è un documento redatto dal progect manager$_G$ e dagli amministratori; si aggiunge all'insieme dei documenti che il team dovrà seguire per tutta la durata del progetto. In particolare questo documento conterrà:

- **Analisi dei rischi**: il fornitore analizza tutti gli eventuali rischi di tipo tecnico, economico e temporale che possono presentarsi durante il progetto, fornendo anche soluzioni che possano risolverli o almeno limitare i loro effetti negativi;
- **Modello di sviluppo**: viene definita la struttura su cui basarsi per la pianificazione, esecuzine e consegna del prodotto software;
- **Pianificazione**: viene pianificato l'insieme delle attività da seguire durante le fasi di progetto, collocandole nel tempo e stabilendo le loro scadenze;
- **Preventivo e consuntivo**: con il preventivo viene mostrata una stima di quello che sarà il carico di lavoro che il team sosterrà durante il progetto, in termini di tempo e i relativi costi. Con il consuntivo di periodo, invece verranno riportati le variazioni dei costi rispetto a quanto preventivato.

2.1.4.3 Piano di Qualifica

I verificatori si occuperanno di redigere il *Piano di Qualifica*, ovvero l'insieme di attività con il compito di fissare obiettivi di qualità del prodotto nel suo complesso. Verranno quindi considerati anche i processi e le risorse necessarie a raggiungere tali obiettivi. Nel dettaglio il *Piano di Qualifica* si focalizza su:

- **Qualità di prodotto**: vengono fissate le politiche per il raggiungimento della qualità, gli obiettivi da raggiungere e gli strumenti necessari al controllo;

- **Qualità di processo**: stabilire sulla base di opportune misurazioni il grado di efficacia ed efficenza di un processo a partire dalla sua definizione;
- **Standard di qualità**: vengono selezionati gli standard che verranno seguiti, per garantire la stabilità del prodotto;
- **Valutazione di miglioramento**: i problemi e le relative soluzioni vengono evidenziate in questa sezione del *Piano di Qualifica*;
- **Resoconto dell'attività di verifica**: vengono riportate i risultati delle metriche come resoconto dell'attività di qualifica.

2.1.5 Strumenti

Di seguito sono illustrati gli strumenti utilizzati per lo svolgimento dell'attivita di fornitura.

- **GanttProject**: software per la produzione di diagrammi di Gantt$_G$ (analizzato nel dettaglio in § 3.8.3);
- **Google Calendar**: sistema di calendari gestito da Google.

2.2 Sviluppo

2.2.1 Scopo

E' il processo che si occupa di stabilire le attività da svolgere per costruire e consegnare il prodotto finale.

2.2.2 Aspettative

Le aspettative sono le seguenti:

- stabilire obiettivi di sviluppo;
- stabilire vincoli tecnici;
- stabilire vincoli di design;
- realizzare il prodotto software che superi i test, e soddisfi i vincoli di progetto.

2.2.3 Descrizione

Il processo di sviluppo si divide in:

- *Analisi dei Requisiti*;
- Progettazione;
- Codifica.

2.2.4 Attività

2.2.4.1 Analisi dei Requisiti

L'elenco dei requisiti necessari allo svolgimento del processo di sviluppo viene raccolto e redatto dagli analisti in un apposito documento di *Analisi dei Requisiti*. Quest'ultimo ha lo scopo di:

- descrivere l'obiettivo del prodotto;
- fornire ai progettisti riferimenti precisi ed affidabili;
- stabilire le prospettive e le funzionalità del prodotto in base ai vincoli fissati dal proponente;
- fornire ai verificatori riferimenti affidabili per la loro attività di controllo;
- valutare rischi, costi e benefici in relazione al carico di lavoro.

Aspettative

L'obiettivo di questa attività è quello di redigere un documento formale contenente tutti i requisiti richiesti dal progetto.

Descrizione

I requisiti saranno raccolti nel seguente modo:

- lettura del capitolato d'appalto;
- confronto con il proponente;
- discussione tra i componenti del gruppo;
- studio dei casi d'uso.

Casi d'uso

I casi d'uso sono scenari che descrivono una possibile sequenza di iterazioni dell'utente, visto come attore$_G$ attivo e/o passivo, e il sistema. La struttura di un caso d'uso è la seguente:

- codice identificativo;
- titolo;
- diagramma UML$_G$ (se presente);
- attore primario;
- precondizioni;
- postcondizioni;
- scenario principale;

- inclusioni (se presenti);
- estensioni (se presenti).

Codice identificativo dei casi d'uso

Il codice di ogni caso d'uso seguirà questo formalismo:

UC[codice_padre].[codice_figlio]

Requisiti

I requisiti seguiranno la seguente struttura:

- **codice identificativo**: è un codice univoco e conforme alla codifica:

  R[Importanza][Tipologia][Codice]

  Le voci riportate nella precedente codifica significano:

  - **Importanza**: la quale può assumere come valori:
    - 1: requisito obbligatorio, irrinuunciabile;
    - 2: requisito desiderabile, perciò non obbligatorio ma riconoscibile;
    - 3: requisito opzionale, ovvero trattabile in un secondo momento o relativamente utile.
  - **Tipologia**: la quale può assumere come valori:
    - F: requisito funzionale;
    - P: requisito prestazionale;
    - Q: requisito di qualità;
    - V: requisito di vincolo.
  - **Codice identificativo**: il quale è un identificatore univoco del requisito, e viene espresso in forma gerarchica padre/figlio.
- **Classificazione**: specifica il peso del requisito facilitando la sua lettura anche se causa ridondanza;
- **Descrizione**: sintesi completa di un requisito;
- **Fonti**: il requisito può avere le seguenti provenienze:
  - capitolato;
  - interno: requisito che gli analisti ritengono di aggiungere in base alle esigenze del team;

- caso d'uso: il requisito proviene da uno o più casi d'uso, dei quali è necessario riportare il codice univoco di caso d'uso;
- verbale: dopo un chiarimento da parte del proponente è possibile che sorga un requisito non preventivato. E le informazioni su di esso sono riportate e tracciati nei rispettivi verbali.

UML

I diagrammi UML servono per descrivere un caso d'uso. Gli analisti dovranno utilizzare la versione v.2.0.

2.3 Progettazione

2.3.1 Scopo

Il processo di *Progettazione* ha il compito di stabilire le migliori operazioni da effettuare per fornire una soluzione soddisfacente per tutti gli stakeholders$_G$. Per fare ciò è necessario stabilire l'architettura$_G$ logica del prodotto. Sulla base del documento *Analisi_dei_requisiti_1.0.0* è possibile individuare le singole parti che compongono il prodotto, il loro dominio e una complessità trattabile. Una volta avute ben chiare queste peculiarità, sarà possibile determinare l'architettura del prodotto.

2.3.2 Descrizione

Il processo di **Progettazione** si divide in:

- **Technology baseline**: mostra ad alto livello le specifiche di progettazione del prodotto e le sue componenti, elencando i diagrammi UML utilizzati per descrivere l'architettura del prodotto;
- **Product baseline**: parte incrementale del prodotto finale da cui continuare a lavorare, integrando le specifiche riportate nella Tecnology Baseline e definendo anche i test necessari alla verifica.

2.3.2.1 Technology baseline

Il progettista incaricato, si occuperà di includere:

- **Diagrammi UML**:
  - diagrammi delle classi;
  - diagrammi dei package;
  - diagrammi di attività;
  - diagrammi di sequenza.
- **Tecnologie utilizzate**: vengono mostrate le tecnologie impiegate, mostrando le loro funzionalità, pregi e difetti;

- **Design pattern**$_G$: sono esposti i design pattern adottati per rappresentare l'architettura del prodotto. Tutti i design pattern devono essere accompagnati da un diagramma e una descrizione delle sue caratteristiche;
- **Tracciamento delle componenti**: viene mostrata la relazione tra il componente e il requisito che intende rispettare;
- **Test di integrazione**: sono operazioni che si occupano di verificare l'unione tra le parti, in base alle rispettive interfaccie.

2.3.2.2 Product baseline

Gli aspetti su cui la Product baseline si sofferma sono:

- **Definizione delle classi**: ogni classe deve essere descritta, specificando scopo e funzionalità;
- **Tracciamento delle classi**: ovvero identificare il requisito a cui si lega una classe;
- **Test di unità**: ovvero verificare le funzionalità della sola classe, senza metterla in relazione con altre componenti del sistema.

2.4 Codifica

2.4.1 Scopo

Lo scopo del processo di *Codifica* è quello di implementare il prodotto software richiesto. Il programmatore è colui che ha il compito di attuare i design pattern e non può agire diversamente.

2.4.2 Aspettative

L'obiettivo di questo processo è la costruzione del prodotto richiesto secondo le specifiche richieste dal proponente. Perciò il programmatore deve attenersi alle norme qui stabilite, al fine di produrre un codice solido e uniforme, facilitando l'attivita di manutenzione, verifica, validazione e miglioramento della qualità del prodotto.

2.4.3 Descrizione

Il codice deve attenersi alle norme e rispettare i requisiti di qualità, espressi nel documento *Piano di Qualifica*, garantendo così la qualità del codice.

2.4.3.1 Stile di codifica

Lo stile di codifica stabilisce le norme che il programmatore deve rispettare, focalizzandosi nei seguenti ambiti:

- **Indentazione**: blocchi di codice innestato deve avere per ogni livello di indentazione quattro spazi, ad eccezione dei commenti. È consigliato impostare adeguatamente la configurazione dell'IDE$_G$ usato;

- **Parentesizzazione**: si richiede di aprire le parentesi in linea e non al di sotto dei costrutti a cui si riferiscono;
- **Scrittura dei metodi**: è desiderabile la poca prolessità del codice di un metodo;
- **Univocità dei nomi**: per evitare ambiguità e incomprensioni, tutti i nomi di classi, metodi, funzioni ed interfacce, devono essere univoci ed esplicativi;
- **Classi**: i nomi delle classi devono iniziare con la lettera maiuscola;
- **Costanti**: i nomi delle costanti devono essere scritte con lettere maiuscola;
- **Metodi**: i nomi dei metodi devono iniziare con la lettera maiuscola, seguita da lettere minuscole. Nel caso di nomi composti da più parole, tutte quelle che seguono la prima iniziano con la prima lettera maiuscola a cui seguono lettere minuscole. Viene così rispettato il modello $CamelCase_G$;
- **Lingua**: codice e commenti devono essere espressi in lingua inglese.

2.4.3.2 Ricorsione

La ricorsione deve essere il più possibile evitata, perché è causa di un aumento di complessità della soluzione ad un problema. Per questo è preferibile adottare soluzioni iterative.

2.4.4 Metriche di qualità

Analisi dei Requisiti

Durante il periodo di analisi vengono identificati i requisiti del progetto e definiti i relativi casi d'uso. Gli obiettivi sono:

- formulare la definizione di casi d'uso e requisiti;
- ottenere la loro approvazione;
- tracciare il loro cambiamento nel tempo.

La strategia prevede:

- considerare lo scopo del progetto e le richieste degli stakeholder;
- esprimere ciò in forma di requisiti, classificati in obbligatori, desiderabili e opzionali;
- valutare il corpo dei requisiti e negoziare cambiamenti se necessario;
- ottenere la loro approvazione da parte del proponente. La metrica usata è la percentuale di requisiti obbligatori soddisfatti (PROS) e viene calcolata con la seguente formula:

$$PROS = \frac{requisiti\_obbligatori\_soddisfatti}{requisiti\_obbligatori\_totali}$$

2.4.4.1 Progettazione

La progettazione consiste nell'identificare la struttura complessiva del sistema, per poi suddividerla nei moduli e unità architetturali che la compongono. In questo modo è più semplice per i programmatori che implementeranno il sistema, comprendere e rispettare i requisiti funzionali. A questo scopo si deve:

- tradurre i requisiti in unità di codice (i moduli);
- assegnare ai programmatori singoli compiti;
- fornire un prototipo di sistema da migliorare, ma funzionante;
- garantire il tracciamento dei requisiti per componente.

Strategie adottate:

- suddivisione della macro-architettura;
- implementazione dei moduli;

2.4.4.2 Codifica

Questo paragrafo elenca le metriche a cui il prodotto software viene sottoposto per misurarne la qualità. I valori riportati sono una dichiarazione di intenti; nulla vieta che in corso d'opera possano avvenire delle revisioni.

- **Linee di Codice**: è la metrica che registra la dimensione di tutto il codice sorgente di un metodo;
- **Numero dei metodi**: questa metrica conta il numero di metodi presenti in un oggetto. Questo strumento permette di determinare la necessità di modularizzare un oggetto;
- **Numero dei paramteri**: un numero elevato di parametri di un metodo comporta ad un alto grado di complessità del problema;
- **Commenti per Linee di Codice**: questa metrica calcola il rapporto tra le linee di commento e quelle di codice effettivo. Permette quindi di stabilire il livello di comprensibilità del codice sorgente da parte del programmatore, visto che un codice commentato risulta più comprensibile di quello sprovvisto di commento o con note superficiali.

2.5 Strumenti

2.5.1 ESLint

È un plugin$_G$ utilizzato per effettuare analisi statica del codice e rilevare problematiche nei pattern codificati in linguaggio JavaScript$_G$.

https://eslint.org/

2.5.2 Draw.io

Programma utilizzato per la creazione di diagrammi UML.

2.5.3 IntelliJ IDEA

IDE_G utilizzato per la codifica in JavaScript, garantendo la piena compatibilità con i sistemi operativi Linux, Windows e MacOS.
https://www.jetbrains.com/idea/

3 Processi di Supporto

3.1 Documentazione

Ogni processo$_G$ e attività$_G$ significativi volti allo sviluppo del progetto sono documentati. Lo scopo di questa sezione è definire gli standard che riguardano i documenti prodotti durante l'intero ciclo di vita del software. I documenti sono consultabili nelle relative sezioni della repository$_G$: https://github.com/teamafkSWE/PredireInGrafana-docs.

3.2 Descrizione

Questo capitolo contiene le decisioni e le norme che sono state scelte per la scrittura, verifica e approvazione della documentazione ufficiale. L'insieme di tali norme garantisce consistenza ed omogeneità nella stesura dei testi.

3.3 Ciclo di vita di un documento

Ogni documento segue le seguenti fasi di ciclo di vita:

- **Sviluppo**: creazione del documento, definizione della struttura e prima stesura di tutte le parti che lo compongono;
- **Verifica**: un documento entra in fase di verifica successivamente al suo completamento. È dovere del *Responsabile* assegnare tale compito ad almeno un *Verificatore*. Quest'ultimo deve applicare le procedure di verifica e segnalare eventuali modifiche da apportare al documento;
- **Approvazione**: il *Responsabile* approva il documento, che sarà quindi ritenuto completo e pronto per il rilascio;
- **Rivisitazione e ampliamento**: con l'avanzare del progetto si prevede di espandere ciascun documento, aggiungendo nuove sezioni o migliorando quanto scritto in precedenza. Sarà compito del *Responsabile* istanziare una nuova fase di sviluppo per provvedere alla realizzazione di questi aggiornamenti. Al termine di essa vengono eseguite nuovamente le fasi di verifica ed approvazione del documento.

3.4 Template

È stato creato un template LaTeX per uniformare la struttura grafica e lo stile di formattazione dei documenti. Lo scopo dei template è quello di permettere, a colui che redige il documento, di adottare automaticamente le conformità previste dalle *Norme di Progetto*. Nel caso quest'ultime cambiassero, essi permettono di agevolare la procedura di adeguamento alle nuove norme.

3.5 Struttura dei documenti

Un file "nome_file.tex" (in cui "nome_file" verrà sostituito dal nome del documento) raccoglie, tramite comandi di input, le sezioni di cui è composto il documento. Tra i file in input ci sono:

- "AFKstyle.sty", contenente i pacchetti necessari alla compilazione e i comandi relativi all'impostazione grafica;
- "copertina.tex", che contiene i comandi LaTeX per l'impostazione della prima pagina del documento;
- "registroModifiche.tex", contenente la tabella delle modifiche.

3.5.1 Prima pagina

Il frontespizio è la prima pagina del documento ed è così strutturato:

- **Logo del gruppo**: logo del *TeamAFK* visibile come primo elemento centrato in alto;
- **Titolo**: nome del documento, posizionato centralmente sotto il logo;
- **Gruppo e progetto**: nome del gruppo e del progetto *Predire in Grafana*, visibile centralmente sotto il titolo;
- **Recapito**: indirizzo di posta elettronica del gruppo, posizionato sotto il nome del gruppo e del progetto;
- **Informazioni sul documento**: tabella posizionata al di sotto del recapito, contenente le seguenti informazioni:
  - **Versione**: versione del documento;
  - **Approvatore**: nome e cognome dei membri del gruppo incaricati dell'approvazione del documento;
  - **Redattori**: nome e cognome dei membri del gruppo incaricati della redazione del documento;
  - **Verificatori**: nome e cognome dei membri del gruppo incaricati della verifica del documento;
  - **Uso**: tipologia d'uso del documento, che può essere "interno" o "esterno";
  - **Distribuzione**: destinatari del documento.
- **Descrizione**: descrizione sintetica del documento, posizionata centralmente in fondo alla pagina.

3.5.2 Registro delle modifiche

Ogni documento dispone di un *Registro delle Modifiche*: una tabella posta a seguito della prima pagina, contenente le modifiche apportate al documento. In essa sono indicati:

- versione del documento dopo la modifica;

- data della modifica;
- breve descrizione della modifica;
- nominativo di chi ha modificato;
- ruolo di chi ha modificato.

3.5.3 Indice

L'indice ha lo scopo di riepilogare e dare una visione macroscopica della struttura del documento, mostrando le parti gerarchiche di cui è composto. Ogni documento è corredato dall'indice dei contenuti, posizionato dopo il *Registro delle modifiche*. Se sono presenti immagini o tabelle all'interno del documento, l'indice dei contenuti è seguito prima dalla lista delle immagini, poi dalla lista delle tabelle.

3.5.4 Contenuto principale

La struttura delle pagine di contenuto è così definita:

- **Logo**: presente in alto a sinistra;
- **Nome del documento**: presente in alto a destra;
- **Riga di separazione**: divide l'intestazione dal contenuto;
- **Contenuto della pagina**: posto tra l'intestazione e il piè di pagina;
- **Riga di separazione**: divide il contenuto dal piè di pagina;
- **Nome e versione del documento**: posto in basso a sinistra;
- **Numero della pagina**: presente in basso a destra, con il formato "Pagina X di Y", in cui la X indica il numero della pagina corrente e Y il numero totale delle pagine.

3.5.5 Verbali

I verbali vengono prodotti dal/i soggetto/i incaricato/i della loro stesura in occasione di incontri tra i membri del team, con o senza la presenza di referenti esterni. È prevista la stesura di più verbali, uno per ogni incontro. La struttura è così definita:

- **Luogo**: luogo di svolgimento dell'incontro;
- **Data**: data dell'incontro, nel formato `YYYY-MM-DD`;
- **Ora di inizio**: l'orario di inizio dell'incontro;
- **Ora di fine**: l'orario di fine dell'incontro;
- **Partecipanti**: elenco dei membri del gruppo presenti all'incontro e, se presenti, i nominativi delle persone esterne che vi hanno partecipato;
- **Topic**: argomenti affrontati durante l'incontro.

Ogni verbale dovrà essere denominato secondo il seguente formato:

VX_YYYY-MM-DD

dove con "X" bisognerà indicarne la tipologia:

- **I**: verbale "interno", incontro tra i membri del team di progetto;
- **E**: verbale "esterno", incontro con partecipanti esterni al gruppo (committente o proponente), per chiarimenti riguardanti il progetto.

3.5.6 Uso dei documenti

I documenti possono essere adibiti ad uso interno o esterno:

- **Interno**: documenti utilizzati all'interno del gruppo, tipicamente *Verbali* e *Norme di Progetto*;
- **Esterno**: documenti destinati a persone esterne, ovvero committenti e proponente.

3.5.7 Note a piè di pagina

Eventuali note vanno indicate nella pagina corrente, in basso a sinistra. Ogni nota deve riportare un numero e una breve descrizione.

3.6 Norme tipografiche

3.6.1 Convenzioni sui nomi dei file

I nomi di file (estensione esclusa) e cartelle utilizzano la convenzione "Snake_case$_G$ " e alcune regole aggiuntive elencate di seguito:

1. i nomi dei file composti da più parole usano il carattere *underscore* come carattere separatore, eccetto il file del *Registro delle modifiche* nominato "registroModifiche.tex";
2. i nomi dei file sono scritti interamente in minuscolo;
3. i nomi dei file dei documenti principali, ossia quelli che raggruppano le sezioni, devono contenere anche la versione;
4. le preposizioni **vanno** messe;
5. i nomi delle cartelle seguono la convenzione Snake_case, ma con la prima lettera della prima parola in maiuscolo (e.i. Norme_di_progetto).

Alcuni esempi di file **corretti** sono:

- studio_di_fattibilità_v1.0.0 (contiene la versione);
- processi_di_supporto (file di una sezione delle *Norme di Progetto*, non contiene la versione).

Alcuni esempi di file **non corretti** sono:

- Norme_di_progetto (usa maiuscole);
- norme-di-progetto (non utilizza underscore come separatore);
- norme_progetto (omette la preposizione "di").

3.6.2 Glossario

- ogni termine inserito nel *Glossario* è marcato con una **G** maiuscola a pedice, solamente nella sua prima occorrenza;
- non vengono segnate con la **G** a pedice le parole da *Glossario* presenti nei titoli e nelle didascalie di immagini e tabelle;
- se nel *Glossario* un termine presenta una descrizione che utilizza termini da glossario, è necessario trattare questi termini come tali, segnando la **G** a pedice e aggiungendoli al documento con la relativa descrizione;
- se nel *Glossario* è presente un termine sinonimo (o tradotto il lingua inglese) di un altro già presente, bisognerà collegarlo alla relativa definizione attraveso il comando `\hyperref[par:"nome_paragrafo"]` e la relativa label `\label{par:nome_paragrafo}`, posta sopra la prima occorrenza di definizione.

3.6.3 Stile del testo

- **Grassetto**: viene applicato se necessario alle voci di un elenco puntato, a titoli o a termini di frasi che si vuol far risaltare;
- **Corsivo**: vengono scritti in corsivo e con la prima lettera maiuscola il nome del progetto *Predire in Grafana*, i ruoli, i documenti citati, il nome del gruppo *TeamAFK* ed il nome dell'azienda proponente *Zucchetti SPA*;
- **Maiuscolo**: solo gli acronimi vengono scritti interamente in maiuscolo; nel caso di nomi o titoli composti da più parole verrà indicato con la lettera maiuscola solamente la prima lettera della parola;
- **Nomi dei documenti**:
  - utilizzare il corsivo per citare un documento;
  - ogniqualvolta si cita un documento, bisogna indicare con la lettera maiuscola le iniziali dei nomi di cui è composto, senza indicarne la versione.
    Caso particolare:
    - quest'ultima può essere indicata quando si deve far riferimento ad una **specifica** versione del documento in questione. In questo caso utilizzare la convenzione Snake_case sopra definita: e.i. *Norme_di_Progetto_v1.0.0*.
  - se si utilizza il documento come titolo o in una voce di elenco, si deve seguire la convenzione sopra riportata ma senza utilizzare il corsivo.

3.6.4 Elenchi puntati

Ogni voce di un elenco o sottoelenco comincia per lettera minuscola, e termina per ";", eccetto l'ultima che termina per ".". Se le voci contengono una descrizione, andranno scritte in grassetto e con la prima lettera maiuscola.

3.6.5 Didascalia

Nella didascalia deve comparire il numero della sezione a cui l'immagine o tabella si riferisce, seguita dal numero progressivo delle stesse di quella sezione e una breve descrizione:

X.Y.Z: <descrizione>

- **X.Y**: rappresenta la sezione;
- **Z**: rappresenta il numero progressivo della tabella o immagine nella sezione X.Y;
- **Descrizione**: breve descrizione identificativa.

3.6.6 Formati comuni

In conformità allo standard ISO 8601[1]:

- le date devono essere scritte secondo il formato gregoriano:
  YYYY-MM-DD
  dove YYYY indica l'anno, MM il mese (da 0 a 12) e DD il giorno (da 01 a 31);
- gli orari devono seguire il formato 24 ore:
  HH:MM
  dove HH indica le ore (da 00 a 23) e MM i minuti (da 00 a 59).

3.6.7 Sigle

Il progetto prevede la redazione di un insieme di documenti, suddivisi in documenti interni ed esterni. Sono di seguito elencati con le rispettive sigle e descrizioni.
I documenti interni sono:

- **Studio di fattibilità - SdF**: descrive in modo sintetico i capitolati e spiega le motivazioni della loro scelta o esclusione;
- **Norme di Progetto - NdP**: sono un riferimento normativo per lo svolgimento del progetto.

I documenti esterni sono:

- **Analisi dei Requisiti - AdR**: stabilisce le caratteristiche che il software deve rispettare;
- **Piano di Progetto - PdP**: descrive la strategia di gestione del progetto, evidenziandone la fattibilità e le criticità;

[1] ISO 8601: standard internazionale per la rappresentazione di date e orari.

- **Piano di Qualifica - PdQ**: descrive la qualità del software e dei processi, e come la si intende raggiungere;
- **Glossario - G**: raccoglie i termini di interesse sui quali è necessaria una descrizione più approfondita che ne chiarisca il significato;
- **Manuale Utente - MU**: a disposizione degli utenti;
- **Manuale Sviluppatore - MS**: a disposizione di sviluppatori e manutentori.

I verbali sono un caso particolare di documenti, che possono essere interni o esterni:

- **Verbale - V**: descrivono le interazioni avvenute durante un incontro tra i membri del team (verbale interno) o con il proponente del progetto (verbale esterno).

Le diverse fasi del progetto sono le seguenti:

- **Revisione dei Requisiti - RR**: studio iniziale del capitolato, se ben fatto permette al gruppo di aggiudicarselo;
- **Revisione di Progettazione - RP**: riguarda la definizione dell'architettura del software e di una Proof of Concept$_G$ per mostrarne la fattibilità;
- **Revisione di Qualifica - RQ**: interessa la definizione dettagliata e la codifica del prodotto;
- **Revisione di Accettazione - RA**: se il prodotto soddisfa i requisiti del proponente, viene accettato e rilasciato.

3.7 Elementi grafici

3.7.1 Immagini

Le immagini sono centrate e hanno una breve didascalia descrittiva sottostante. Tutte le immagini devono aver il formato `.png`$_G$.

3.7.2 Tabelle

Le tabelle sono scritte allo stesso modo in tutti i documenti LaTeX: fare riferimento alla Wiki di Overleaf https://www.overleaf.com/learn/latex/tables.
Ogni tabella deve essere accompagnata dalla propria didascalia descrittiva (caption), da posizionare al di sopra della tabella.
Fanno eccezione le tabelle del *Registro delle Modifiche* che non hanno didascalia.

3.7.3 Diagrammi UML

I diagrammi UML$_G$ vengono inseriti nei documenti sotto forma di immagine.

3.8 Strumenti

3.8.1 LaTeX

LaTeX è lo strumento scelto per la stesura dei documenti, un linguaggio basato sul programma di composizione tipografica TeX, che permette di scrivere documenti in modo ordinato, modulare, collaborativo e scalabile.

3.8.2 TeXmaker

TeXmaker è l'editor utilizzato per la stesura del codice LaTeX. Questo strumento, oltre ad integrare un compilatore e un visualizzatore PDF, fornisce suggerimenti di completamento per comandi LaTeX.

https://www.xm1math.net/texmaker/

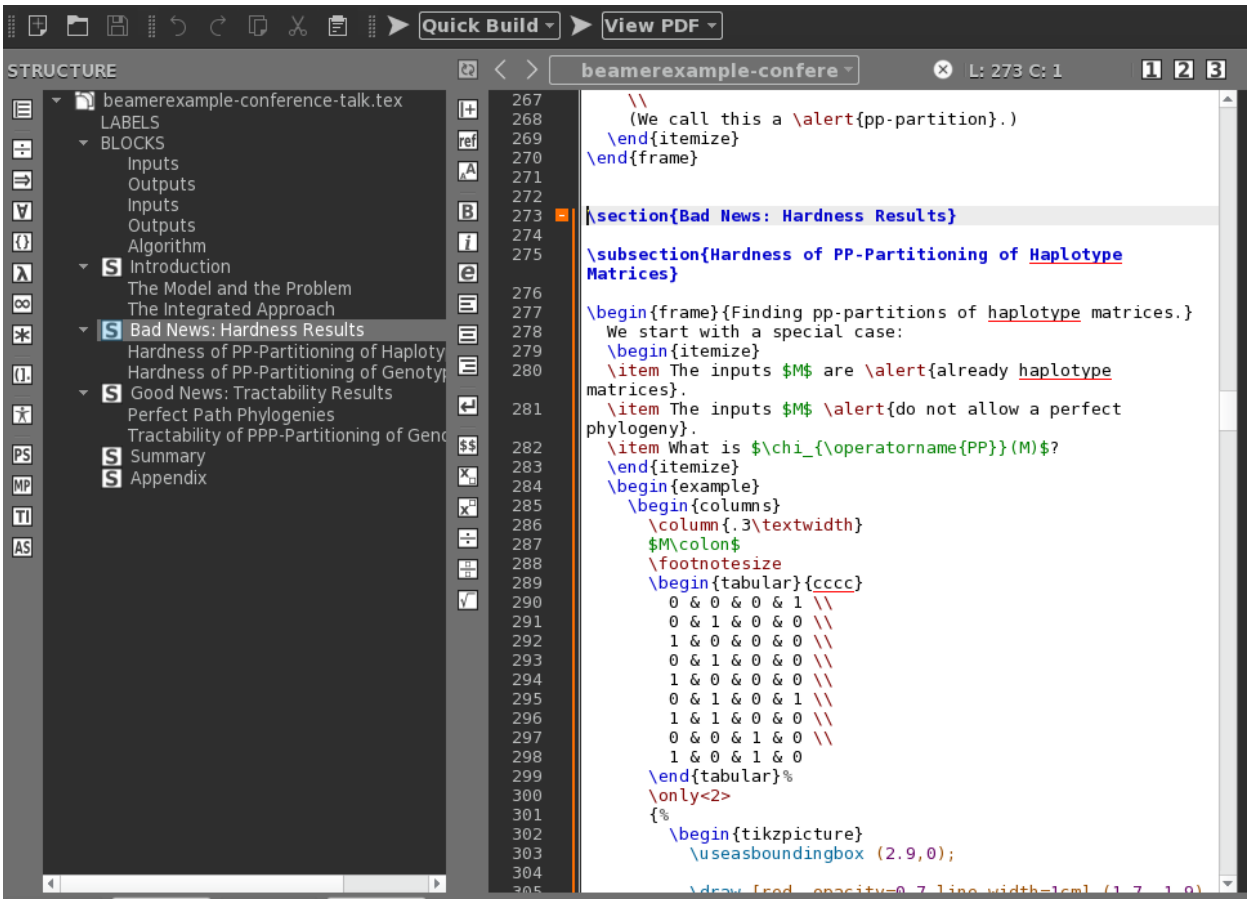

Figura 3.8.1: TeXmaker - per la stesura dei documenti

3.8.3 GanttProject

GanttProject è un programma gratuito dedicato alla produzione dei diagrammi di Gantt$_G$. Permette di creare task e milestone$_G$, organizzare le task in lavoro strutturato a interruzioni, disegnare i vincoli di dipendenza tra di esse e molte altre utilità, generando automaticamente

il relativo diagramma.

https://www.ganttproject.biz/

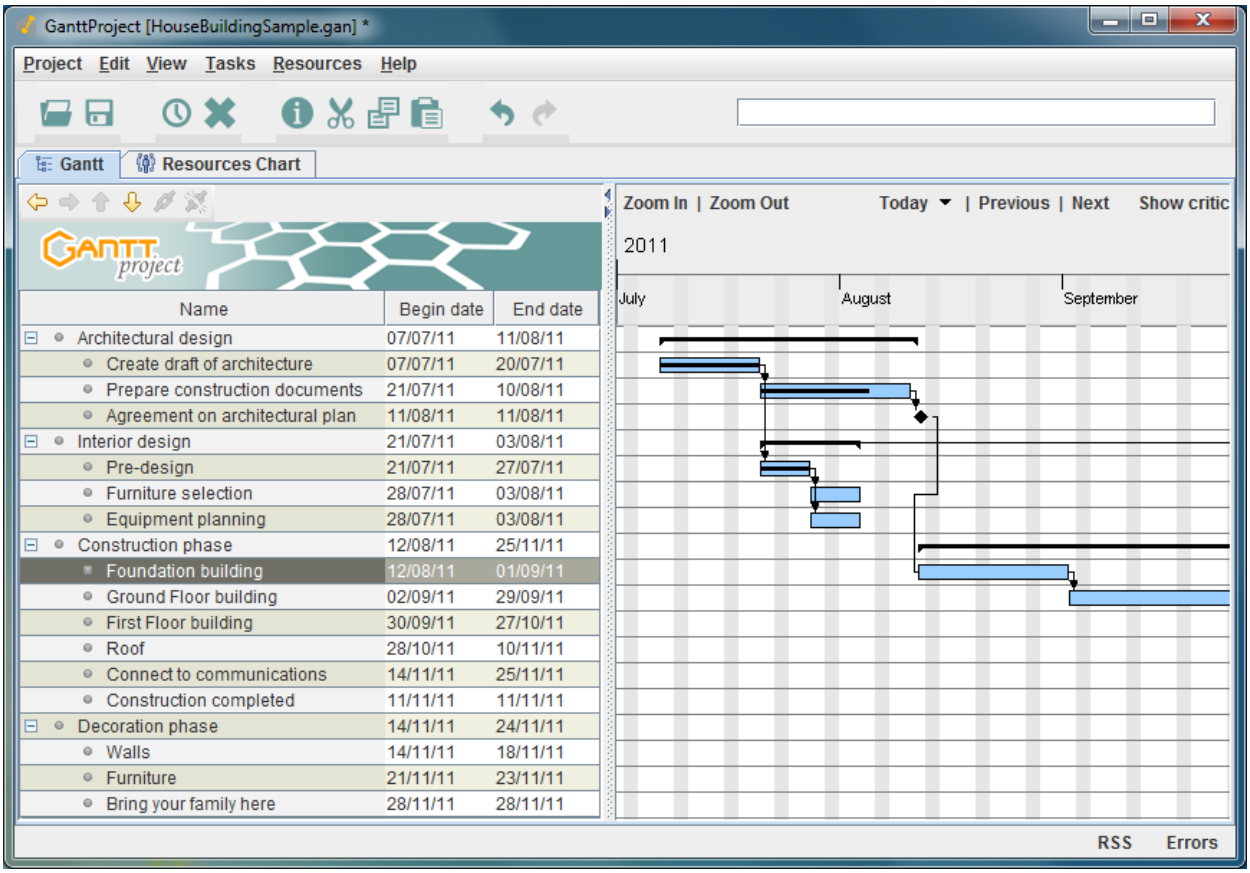

Figura 3.8.2: GanttProject - per la relizzazione di diagrammi di Gantt

3.8.4 Draw.io

Draw.io viene utilizzato per la produzione degli UML.

https://www.draw.io/

3.9 Gestione della configurazione

L'obiettivo della configurazione è di creare ordine tra i documenti e il software. Tutto ciò che è configurato ha uno stato identificativo, è modificato secondo regole ben definite ed è posto sotto versionamento$_G$.

3.9.1 Versionamento

3.9.1.1 Versionamento dei documenti

Ogni versione di qualsiasi documento deve corrispondere ad una riga della tabella delle modifiche. Il numero di versione è composto da tre cifre:

X.Y.Z

- **X**: rappresenta una versione stabile del documento, resa tale dopo l'approvazione del *Responsabile* di progetto:
  - inizia da 0;
  - viene incrementata di un'unità alla volta.
- **Y**: indica l'ultima versione del documento che ha passato la fase di verifica:
  - inizia da 0;
  - viene incrementato dal verificatore ad ogni verifica;
  - quando viene incrementato X, viene riportato a 0.
- **Z**: indica l'ultima modifica apportata al documento dal redattore:
  - inizia da 0;
  - viene incrementato dal redattore del documento ad ogni modifica;
  - quando viene incrementato Y, viene riportato a 0.

3.9.1.2 GitHub

Per le parti del progetto da versionare si è scelto di usare GitHub$_G$, un servizio del sistema di versionamento distribuito Git per contenere la repository remota.
I membri del team possono interagire con il VCS$_G$ sia da linea di comando, sia attarverso software che ne migliorano l'usabilità, come GitKraken e GitHub Desktop. La versione ufficiale del progetto è ospitata in una repository remota su GitHub, all'indirizzo

https://github.com/teamafkSWE

3.9.1.3 Struttura del repository

All'interno della repository principale sopra descritta troviamo due differenti repository:

- **PredireInGrafana-docs**: contiene tutti i documenti ufficiali del progetto, suddivisi in specifiche cartelle:

  https://github.com/teamafkSWE/PredireInGrafana-docs

  - **Cartella principale - RR**: raccoglie i file sorgenti per la compilazione dei documenti, suddivisi tra esterni ed interni, realizzati per la *Revisione dei Requisiti*. In futuro saranno aggiunte cartelle distinte nominate **RP, RQ** e **RA**, contenenti i file delle rispettive consegne.
    - **Tipologia_di_documento**: ogni documento avrà la rispettiva cartella (e.i. Norme_di_progetto), contenente tutti i file (sezioni ed immagini) necessari per la sua compilazione;

- **copertina.tex**: file che permetterà una facile e rapida modifica dell'impostazione testuale del frontespizio.

– **Template**: contiene tutti i file che definiscono il template LaTeX per la creazione di nuovi documenti.

- **PredireInGrafana-SW**: conterrà tutti i file di codifica dei plug-in da sviluppare.
  https://github.com/teamafkSWE/PredireInGrafana-SW

Entrambe le repository, avranno una propria struttura identica a livello:

- **Locale**: ogni membro del gruppo lavora sui file clonati dal repository remoto nel proprio PC;
- **Remoto**: presente su GitHub, contiene il lavoro svolto da ogni componente e che viene condiviso con il team.

3.9.1.4 Tipi di file

I file utilizzati per la documentazione del progetto sono:

- file con estensione .tex di LaTeX;
- file con estensione .pdf (da consegnare);
- file di stile, .sty, e immagini di supporto.

Il file ".gitignore" è presente al livello più esterno della repository ed elenca tutti i file esclusi dal versionamento.

3.9.1.5 Utilizzo di Git

Il repository di Git è composto da vari branch$_G$, per favorire la collaborazione tra i vari membri e il parallelismo delle attività. Si consiglia quindi di seguire questa procedura:

1. scegliere il proprio branch di lavoro;
2. eseguire il pull$_G$ dal repository remoto, che effettua quindi l'aggiornamento del proprio repository locale;
3. svolgere il compito assegnato;
4. eseguire il comando di aggiunta (add) dei file nuovi o modificati da condividere all'area di staging$_G$;
5. eseguire il comando di commit$_G$ dei file aggiunti, corredato da un messaggio che identifica il lavoro svolto;
6. eseguire il push$_G$ del commit sul repository remoto.

3.9.1.6 Gestione delle modifiche

Tutti i membri del team possono modificare i file in ogni branch, ad eccezione del branch master, per il quale occorre richiedere una pull e ottenere l'approvazione di un altro membro.

3.10 Gestione della qualità

Lo scopo è di garantire che il prodotto e i servizi offerti rispettino gli obiettivi di qualità e che i bisogni del proponente siano soddisfatti.

3.10.1 Descrizione

La gestione della qualità viene approfondita nel *Piano di Qualifica*, dove sono descritte le modalità utilizzate per garantire la qualità nello sviluppo del progetto. In particolare:

- sono presentati gli standard utilizzati;
- sono individuati i processi$_G$ di interesse;
- sono individuati gli attributi del software più importanti per il progetto.

Per ogni processo vengono descritti:

- gli obiettivi da perseguire;
- le strategie da applicare;
- le metriche da utilizzare.

L'obiettivo è quello di ottenere software e documentazione di qualità soddisfacente.

3.10.2 Strumenti

Gli strumenti utilizzati per la qualità sono:

- forniti dallo standard ISO-12207[2];
- le metriche.

[2]ISO-12207: standard ISO per la gestione del ciclo di vita del software.

3.10.2.1 Metriche

Le metriche sono distinte in tre categorie: processi, documentazione e codifica. Per ciascuna di esse si indica il motivo per cui è stata scelta e la procedura di calcolo.

3.10.2.1.1 Classificazione

Le metriche rispetteranno la seguente notazione:

M[categoria][numero]

dove:

- Categoria: indica la categoria della metrica, più precisamente:
  - **P** per indicare le metriche dei processi;
  - **D** per indicare le metriche dei documenti;
  - **S** per indicare le metriche della codifica del software.
- Numero: identifica in maniera univoca la metrica in ogni categoria, assume un valore intero a due cifre incrementale a partire da 1.

Se nelle formule di calcolo delle metriche è presente il simbolo "#", va inteso come la parola "numero". Un esempio può essere il seguente:

$$\#parole\_doc = numero\ di\ parole\ presenti\ nel\ documento$$

3.10.2.1.2 Metriche per i processi

Tabella 3.10.1: Metriche dei processi

Nome	Codice	Descrizione
Schedule Variance (SV)	MP01	La Schedule Variance indica se una certa attività o processo è in anticipo, in pari, o in ritardo rispetto alla data di scadenza prevista. È calcolata utilizzando la seguente formula: $$SV = DCE - DCP$$ dove: • **DCE**: data conclusione effettiva; • **DCP**: data conclusione pianificata. Se $SV \leq 0$ significa che l'attività o il processo è in pari o in anticipo, invece, se $SV > 0$ significa che l'attività è in ritardo.
Budget Variance (BV)	MP02	Permette di controllare i costi sostenuti alla data corrente rispetto al budget preventivato. Viene calcolata in fase di consuntivo di periodo utilizzando la seguente formula: $$BV[\%] = \frac{CP - CE}{CE} \cdot 100$$ dove: • **CP**: costo preventivato; • **CE**: costo effettivo. Se $BV[\%] \geq 0$ indica che il budget sta venendo speso più lentamente di quanto pianificato, se negativo invece indica che il budget sta venendo speso più velocemente di quanto pianificato.
Produttività (P)	MP03	Rappresenta la produttività media delle risorse impiegate, cioè delle persone coinvolte, nelle diverse fasi del progetto. E' misurata in termini di numero di linee di codice (LOC) sviluppate da una persona nell'unità di tempo stabilita (settimana). E'utilizzata per valutare lo sforzo richiesto per lo sviluppo del progetto a fronte delle sue dimensioni. $$Pmedia = \frac{LOC}{settimana}$$

3.10.2.1.3 Metriche per i documenti

Tabella 3.10.2: Metriche dei documenti

Nome	Codice	Descrizione
Indice Gulpease (IG)	MD01	È un indice di leggibilità di un testo tarato sulla lingua italiana e basato su due variabili linguistiche: la lunghezza della parola e la lunghezza della frase rispetto al numero delle lettere. La formula per il suo calcolo è: $$IG = 89 + \frac{300 \cdot (\#frasi) - 10 \cdot (\#lettere)}{\#parole}$$ Il risultato è un valore compreso nell'intervallo tra 0 e 100, dove il valore 100 indica la più alta leggibilità. Un indice inferiore a 80 indica documenti di difficile leggibilità per chi ha la licenza elementare, inferiore a 60 per chi ha la licenza media, inferiore a 40 per chi ha un diploma superiore.

3.10.2.1.4 Metriche per la codifica

Questa sezione ha lo scopo di fornire delle metriche per garantire un buon livello di qualità del software.

Tabella 3.10.3: Metriche del software

Nome	Codice	Descrizione
Linee di Codice (LOC)	MS01	Rappresenta le dimensioni del codice di un metodo. È espresso in termini di numero di linee di codice, ed è utilizzato per dimensionare la produttività delle persone e da questa lo sforzo richiesto per sviluppare tale funzione.
Numero dei Metodi (NM)	MS02	Numero medio di metodi contenuti nelle classi di un oggetto. Un numero troppo alto di metodi può indicare la necessità di scomporre la classe. Un numero troppo basso deve far riflettere sull'effettiva utilità della classe in esame. $$NM = \frac{\sum \#metodi}{\#classi}$$
Numero di Parametri (NP)	MS03	Numero di parametri passati ad un metodo. Un eccessivo numero di parametri passati ad un metodo può indicare un'eccessiva complessità dello stesso.
Commenti per Linee di Codice (CLC)	MS04	Rapporto fra numero di righe di commento e numero totale di righe (vuote escluse). Un codice ben commentato può essere compreso più facilmente e velocemente, facilitando le operazioni di manutenzione. $$CLC = \frac{\#righe\_commento}{\#tot\_righe}$$
Code Coverage (CC)	MS05	Percentuale delle linee di codice coperte dai test. Avere codice coperto da test riduce la possibilità di introdurre errori nel prodotto. $$CC[\%] = \frac{\#righe\_codice\_testate}{\#tot\_righe\_codice}$$

3.10.3 Verifica

Il processo di verifica ha come scopo la realizzazione di prodotti corretti, coesi e completi. Sono soggetti a verifica i documenti e il software. Il processo di verifica deve rispettare i seguenti punti:

- la verifica deve essere effettuata seguendo procedure ben definite;
- per verificare vi sono criteri chiari e affidabili;
- ogni fase del prodotto viene verificata;
- dopo la verifica il prodotto è in uno stato stabile;
- il prodotto può essere validato.

Il processo di verifica prende in input ciò che è già stato prodotto e lo restituisce in uno stato conforme alle aspettative (stabile). Per ottenere tale risultato ci si affida a processi di analisi e test.

3.10.3.1 Strumenti di verifica

Correzione ortografica

Il controllo ortografico viene eseguito principalmente basandosi sugli strumenti integrati in TEXmaker, il quale fornisce un dizionario italiano e sottolinea in rosso le parole che non vi appartengono.

3.10.3.2 Analisi

Il processo di analisi si suddivide in statica e dinamica.

Analisi statica

L'analisi statica effettua controlli su documenti e codice. Questo tipo di analisi serve per rilevare varie tipologie di anomalie.
Questa attività viene effettuata con l'ausilio di due metodi manuali di lettura (attuati da persone) differenti:

- **Walkthrough**: i vari componenti del team effettuano una lettura scrupolosa di documenti e/o codice alla ricerca di errori, senza sapere inizialmente se ce ne siano;
- **Inspection**: i verificatori usano liste di controllo (checklist) per fare ispezione cercando errori specifici in parti specifiche. Questa tecnica permette quindi di aumentare l'efficienza dello sviluppo, diminuendo i tempi dell'analisi statica.

A seguire sono descritte le liste di controllo utilizzabili per le ispezioni:

Tabella 3.10.4: Errori frequenti nei documenti

Oggetto	Controllo
Formato data	Deve seguire il formato gregoriano YYYY-MM-DD
Sintassi	La frase è troppo complessa e deve essere semplificata
Punteggiatura degli elenchi	Ogni voce termina in ";" eccetto l'ultima che termina in "."
Errori di battitura	Tipici errori di battitura dovuti alla vicinanza delle lettere sulla tastiera, ad esempio la lettera "a" al posto della "s", "i" al posto di "o", "m" al posto di "n"
Gerarchia delle sezioni	Non viene scritta la gerarchia delle sezioni dei documenti in maniera adeguata, ovvero: section, subsection, subsubsection, paragraph, subparagraph

Analisi dinamica

L'analisi dinamica è una tecnica di analisi del prodotto software che richiede la sua esecuzione. Produce una misura della qualità del prodotto, mediante l'esecuzione di test specifici che verificano se il prodotto funziona e se ci sono anomalie.

3.10.3.3 Test

I test sono l'attività fondamentale dell'analisi dinamica: il loro scopo è verificare che il codice scritto funzioni correttamente. I test devono:

- essere ripetibili;
- specificare l'ambiente di esecuzione;
- identificare input e output richiesti;
- avvertire di possibili effetti indesiderati;
- fornire informazioni sui risultati dell'esecuzione.

Ci sono vari tipi di test, ognuno dei quali ha un diverso oggetto di verifica e scopo.

Test funzionali

Test condotti per valutare la conformità di un componente o sistema con requisiti$_G$ funzionali. Ne fanno parte:

- **Test di unità**: si eseguono su unità di software, e si concentrano sul loro funzionamento individuale;
- **Test di integrazione**: verificano se sono rispettati i contratti di interfaccia tra più moduli o sub-system (interni o esterni);
- **Test di accettazione**: gli UAT (User Acceptance Testing) si occupano di verificare il prodotto e, in particolare, il soddisfacimento del cliente. Il superamento di questo test garantisce che il software sia pronto per essere rilasciato.

Test di regressione

Il test di regressione va eseguito ogni volta che viene modificata un'implementazione in un programma. È possibile eseguire nuovamente i test esistenti sul codice modificato, integrando solo le parti che abbiano precedentemente superato il test di unità, per stabilire se le modifiche apportate hanno alterato elementi precedentemente funzionanti.
Se necessario è anche possibile scrivere nuovi test.

Test di sistema

Verificano il comportamento dell'intero sistema.
Lo scopo principale di questi test è la verifica del sistema rispetto alle specifiche tecniche definite nell'*Analisi dei Requisiti*.

3.10.3.3.1 Codifica dei test

La codifica dei test è la seguente:

TX[Tipo_Requisito][Importanza][CodiceUC]

dove:

- **X**: indica il tipo di test. Può assumere i seguenti valori:
  - **U**: indica un test di unità;
  - **I**: indica un test di integrazione;
  - **A**: indica un test di accettazione;
  - **R**: indica un test di regressione;
  - **S**: indica un test di sistema.
- **Tipo_Requisito**: indica il tipo del requisito. Esso può essere:
  - **O**: obbligatorio;
  - **D**: desiderabile;
  - **F**: facoltativo.
- **Importanza**: indica l'importanza del requisito. Viene indicata con:
  - **F**: per indicare un requisito funzionale;
  - **V**: per indicare un requisito di vincolo;
  - **Q**: per indicare un requisito di qualità;
  - **P**: per indicare un requisito prestazionale.
- **CodiceUC**: rappresenta il codice identificativo crescente del componente da verificare.

3.10.4 Validazione

Questo processo avviene tramite test pianificati dal *Progettista* ed in seguito eseguiti dal *Verificatore*. Lo scopo di quest'ultimi è appunto accertare che il prodotto finale corrisponda alle attese, soddisfando tutti i requisiti concordati e i bisogni del committente.
Tali test sono soggetti a tracciamento e verranno riportati all'interno del *Piano di Qualifica*.

4 Processi Organizzativi

4.1 Gestione di Progetto

4.1.1 Ruoli di progetto

Ciascun membro del gruppo, a rotazione, deve ricoprire il ruolo che gli viene assegnato e che corrisponde all'omonima figura aziendale. Nel *Piano di Progetto*$_G$ vengono organizzate e pianificate le attività assegnate agli specifici ruoli. I ruoli che ogni componente del gruppo è tenuto a rappresentare sono descritti di seguito.

4.1.1.1 Responsabile di progetto

Il *Responsabile di progetto* (o semplicemente *Responsabile*) è una figura chiave in quanto ricadono su di lui le responsabilità di pianificazione, gestione, controllo e coordinamento delle risorse e attività del gruppo. Il *Responsabile* si occupa anche di interfacciare il gruppo con le persone esterne facendo da intermediario: sono quindi di sua competenza le comunicazioni con committente e proponente. Questa figura è incaricata anche di analizzare e gestire le criticità, che si incontrano durante il progetto, e di approvare i documenti.

4.1.1.2 Amministratore di progetto

L'*Amministratore* ha il compito di supporto e controllo dell'ambiente di lavoro. Egli deve quindi:

- dirigere le infrastrutture di supporto;
- risolvere problemi legati alla gestione dei processi;
- gestire la documentazione;
- controllare versioni e configurazioni.

4.1.1.3 Analista

L'*Analista* si occupa di analisi dei problemi e del dominio applicativo. Questa figura ha anche il compito di redigere i documenti, in questo caso può essere definito come *Redattore*. Le sue responsabilità sono:

- studio del dominio del problema;
- definizione della complessità e dei requisiti dello stesso;
- redazione del documenti: *Analisi dei Requisiti* e *Studio di Fattibilità*.

4.1.1.4 Progettista

Il *Progettista* gestisce gli aspetti tecnologici e tecnici del progetto. Il *Progettista* deve:

- effettuare scelte efficienti ed ottimizzate su aspetti tecnici del progetto;

- sviluppare un'architettura che sfrutti tecnologie note ed ottimizzate, su cui basare un prodotto stabile e manutenibile.

4.1.1.5 Programmatore

Il *Programmatore* è responsabile della codifica del progetto e delle componenti di supporto che serviranno per effettuare le prove di verifica e validazione sul prodotto. Il *Programmatore* si occupa di:

- implementare le decisioni del progettista;
- creare o gestire componenti di supporto per la verifica e validazione del codice.

4.1.1.6 Verificatore

Il *Verificatore* si occupa di controllare il prodotto del lavoro svolto dagli altri membri del team, sia esso codice o documentazione. Per le correzioni si affida agli standard definiti nelle *Norme di Progetto*, nonché alla propria esperienza e capacità di giudizio. Il *Verificatore* deve:

- ispezionare i prodotti in fase di revisione, avvalendosi delle tecniche e degli strumenti definiti nelle *Norme di Progetto*;
- evidenziare difetti ed errori del prodotto in esame;
- segnalare eventuali errori trovati all'autore dell'oggetto preso in esame o alla persona che ha responsabilità su di esso.

4.1.2 Gestione dei rischi

È compito del *Responsabile* rilevare i rischi e renderli noti, tramite un continuo monitoraggio e una continua identificazione di quest'ultimi.
Qualora dovesse essere **identificato** un nuovo rischio è necessario procedere con i seguenti passaggi:

1. **Classificare** il rischio seguendo la codifica;
2. **Descrivere** una strategia da applicare per gestire il rischio;
3. **Riportare** il rischio nel *Piano di Progetto*.

4.1.2.1 Codifica

La codifica dei rischi è utilizzata per la classificazione. Il codice di un rischio si presenta nella forma:

Ri[**Categoria**][**Grado**][**Numero**]

dove:

- **Categoria**: indica la categoria del rischio, essa può assumere i valori:
  - **O** per i rischi organizzativi;

  - **T** per i rischi tecnologici;
  - **P** per i rischi interpersonali.

- **Grado**: indica il grado di rischio, è la somma tra la probabilità e la gravità. Queste ultime possono assumere i seguenti valori:
  - **0**: bassa;
  - **1**: media;
  - **2**: alta.

  (In questo caso un rischio con una probabilità bassa ma una gravità alta avrà lo stesso grado di un rischio con una probabilità alta e una gravità bassa, ovvero 2)

- **Numero**: insieme a categoria e grado identifica in maniera univoca il rischio, può assumere un valore intero progressivo ad una cifra (0-9).

4.2 Processi di Coordinamento

Di seguito vengono descritte le norme che regolano le comunicazioni e gli incontri del gruppo, che siano tra i membri o con committenti e proponenti.

4.2.1 Gestione Comunicazioni

4.2.1.1 Comunicazioni Interne

Le comunicazioni interne ai membri del gruppo vengono gestite tramite 2 applicazioni:

- Telegram$_G$;
- Discord$_G$.

È stato disposto un gruppo Telegram sul quale si discute di tematiche generali o collettive, garantendo risposte rapide e ordinate in caso di decisioni per votazione, grazie ad apposite funzioni dette bot di Telegram$_G$.
Discord viene usato principalmente per le riunioni tra i membri del gruppo, ma l'applicazione mette anche a disposizione dei canali testuali. Tali canali sono stati suddivisi per tema e vengono usati per le comunicazioni specifiche per agevolare la stesura dei documenti. Vengono suddivisi in:

- **General**: per discussioni riguardanti rotazioni dei ruoli e decisione degli argomenti da discutere nelle riunioni;
- **Links**: per tenere traccia di tutti i link utili al progetto;
- **Analisi-requisiti**: per discutere gli Use Case$_G$ e i requisiti necessari alla stesura dell'*Analisi dei Requisiti*;
- **Norme**: per discutere riguardo le regole del *Way of Working* del gruppo, le norme da seguire e, di conseguenza, la stesura del documento *Norme di Progetto*$_G$;

- **Piano-progetto**: per confrontarsi riguardo il monte ore dei vari ruoli e per facilitare la stesura del documento *Piano di Progetto*;
- **Piano-qualifica**: per discutere di strategie da attuare per garantire qualità attraverso verifica$_G$ e validazione$_G$.

4.2.1.2 Comunicazioni Esterne

Le comunicazioni con soggetti esterni al gruppo sono di competenza del responsabile. Gli strumenti predefiniti sono la posta elettronica, dove viene utilizzato l'indirizzo gruppoafk15@gmail.com. Per comunicare con *Zucchetti SPA* viene usato il servizio Skype$_G$ per le chiamate. Il responsabile ha il compito di tenere informati gli altri componenti del gruppo in caso di assenza.

4.2.2 Gestione Riunioni

Le riunioni possono essere interne o esterne. All'inizio di ogni riunione il *Responsabile* nomina, tra i componenti del gruppo, un *Segretario* che si occuperà di far rispettare l'ordine del giorno. Inoltre ha l'onere della stesura del *Verbale di Riunione*$_G$.

4.2.2.1 Riunioni interne

E' compito del *Responsabile* organizzare riunioni interne al gruppo. Ciò prevede, più nello specifico, la stesura dell'ordine del giorno e stabilire data, orario e luogo di incontro, mediando se necessario con i membri per permettere la presenza di tutti. Le riunioni sono tenute principalmente usando Discord, così da essere il più facilmente raggiungibili. Il *Responsabile* deve inoltre assicurarsi, attraverso la comunicazione mediante i mezzi propri del gruppo, che ogni componente sia pienamente a conoscenza della riunione in tutti i suoi dettagli.
D'altro canto ogni membro del gruppo deve presentarsi puntuale agli appuntamenti, e comunicare in anticipo eventuali ritardi o assenze adeguatamente giustificate.

4.2.2.2 Riunioni esterne

E' nuovamente compito del *Responsabile* organizzare riunioni esterne. Nello specifico egli deve preoccuparsi di contattare l'azienda proponente per fissare gli incontri e qualora sia necessario, tenendo conto anche delle preferenze di date e orario espresse dagli altri membri del gruppo. La partecipazione a tali riunioni deve essere, a meno di casi eccezionali, unanime. Ogni membro del gruppo può, inoltre, esprimere al Responsabile una richiesta, adeguatamente motivata, di fissare una riunione esterna. A questo punto sarà compito dello stesso Responsabile giudicare come valida o meno la richiesta presentatagli ed agire di conseguenza.

4.2.2.3 Verbale di riunione

Ad ogni riunione, interna o esterna, è compito del *Segretario* designato redigere il *Verbale di riunione* corrispondente, che deve essere poi approvato dal *Responsabile*. La struttura del *Verbale* è definita in § 3.5.5.

4.3 Strumenti

Il gruppo, nel corso del progetto, ha utilizzato o utilizzerà i seguenti strumenti:

- **Telegram**: strumento di messaggistica usato per comunicazioni veloci tra i membri;
- **Discord**: per comunicazioni specifiche o per riunioni interne;
- **Git**: sistema di controllo di versionamento;
- **Gitflow**: sistema per agevolare varie operazioni su Git;
- **GitHub**: per il versionamento e il salvataggio in remoto di tutti i file riguardanti il progetto;
- **GanttProject**: software OpenSource$_G$ usato per la realizzazione dei diagrammi di Gantt;
- **Google Doc**: editor di testo cloud, usato per tenere degli appunti modificabili da tutti;
- **Google Drive**: utilizzato per il salvataggio in remoto dei file non sottoposti a versionamento, in modo da essere reperibili a tutti i membri;
- **Google Calendar**: per tenere traccia delle varie scadenze o riunioni fissate;
- **Skype**: servizio che offre possibilità di fare videoconferenze e chiamate VoIP, utilizzato per comunicare con il proponente;
- **Sistema Operativo**: i requisiti non indicano la necessità di usare un sistema operativo specifico, verranno quindi utilizzati Windows, Linux e Mac OS dai diversi membri del team.