# Norme di progetto

v0.9.0


Farmacode

farmacode.swe.unipd@gmail.com

# Registro delle modifiche

| Versione | Data | Scrittori | Revisori | Descrizione |
|---|---|---|---|---|
| 0.9.0 | 14-12-2023 | Bomben Filippo | | Stesura documento da sezione 3.3 a 3.7, eccetto 3.4.2.2 |
| 0.8.0 | 10-12-2023 | Rosson Lorenzo | Bomben Filippo | Completata prima stesura della sezione 2, e modificate alcune parti della sezione 3 |
| 0.7.1 | 5-12-2023 | Rosson Lorenzo | Bomben Filippo | Completata sezione 4 con alcuni miglioramenti |
| 0.7.0 | 2-12-2023 | Rosson Lorenzo | Favaron Riccardo | Realizzata prima stesura sezione 4 |
| 0.6.0 | 25-11-2023 | Rosson Lorenzo | Favaron Riccardo | Realizzata prima stesura sezione 3.2, apportate modifiche alla sezione 3.1.5 |
| 0.5.0 | 21-11-2023 | Baggio Matteo | Carraro Alessandro | Trasferimento da LaTeX a Typst del documento |
| 0.4.0 | 20-11-2023 | Passarella Alessandro | Carraro Alessandro | Completamento stesura sezione 3.1 |
| 0.3.0 | 18-11-2023 | Baggio Matteo | Carraro Alessandro | Completamento stesura sezione 1 |
| 0.2.0 | 15-11-2023 | Baggio Matteo<br>Passarella Alessandro | Carraro Alessandro | Stesura indice |
| 0.1.0 | 12-11-2023 | Bomben Filippo<br>Rosson Lorenzo | Baggio Matteo | Stesura iniziale del documento |

# Indice

# 1) Introduzione al documento

Questo documento è stato creato per identificare le *best practices* di progetto e per stabilire una metodologia di lavoro chiara nel corso dell'attività produttiva. L'obiettivo è garantire una gestione omogenea e coesa del lavoro. Per facilitare il monitoraggio del progresso e consentire un approccio incrementale, vengono registrate le diverse versioni del documento.

## 1.1) Scopo del prodotto

Lo scopo del prodotto è creare un'applicazione dove sia possibile verificare i possibili interessi di un cliente nei confronti di un prodotto. Al giorno d'oggi l'ambito degli e-commerce si sta sempre più espandendo ed evolvendo. La presenza di negozi virtuali permette di accedere a molti dati legati agli acquisti, alle preferenze ed al comportamento degli utenti. Questi dati se analizzati propriamente permettono di prevedere preferenze e comportamenti futuri degli utenti, dando spazio ad operazioni di marketing mirate.

Il prodotto sarà dunque un'applicazione attraverso la quale l'amministrazione di un e-commerce sarà in grado di accedere ai risultati dell'analisi dei dati relativi all'utilizzo della suddetta attività. Il lavoro principale di questa applicazione non sarà dunque svolto dal lato dell'utente, il quale avrà solo accesso ad un'analisi dei dati e potrà garantire feedback sulla loro correttezza, ma sarà svolto da un algoritmo non visibile né accessibile all'utente. Questo algoritmo utilizzerà la tecnologia dell'intelligenza artificiale per analizzare i dati forniti dall'azienda con lo scopo di trovare e definire le correlazioni tra i vari prodotti, tra i vari utenti e tra utenti e prodotti. Queste correlazioni trovate su più livelli di profondità permetteranno di creare un altro set di dati, dal quale l'utente dell'applicazione potrà accedere ai dati che necessita, principalmente questi dati saranno gli N prodotti che potrebbero interessare ad un X utente e gli N utenti che potrebbero essere interessati ad un X prodotto.

Questa applicazione inoltre per comodità d'uso sarà sviluppata sotto la forma di una *webapp* che potrà essere accessibile utilizzando diversi dispositivi, sistemi e browser.

## 1.2) Glossario

Si fa inoltre notare la presenza di un Glossario nel quale sono riportati i termini utilizzati nei documenti. Con questo principio le best practices per la creazione del prodotto riusciranno facilmente ad essere rispettate garantendo il più possibile l'omogeneità del prodotto.

## 1.3) Miglioramenti e maturità del documento

Questo documento è stato creato seguendo un approccio incrementale, il che implica la sua natura adattabile e suscettibile di modifiche nel tempo. Queste modifiche saranno apportate in risposta alle esigenze concordate tra i membri del gruppo e il proponente. Pertanto, questa versione del documento non deve essere considerata come una versione definitiva o completa, ma piuttosto come un punto di partenza che sarà ulteriormente sviluppato e aggiornato per meglio rispondere alle mutevoli esigenze del progetto.

## 1.4) Riferimenti

# 2) Processi primari

## 2.1) Fornitura

### 2.1.1) Descrizione e Scopo

In questa sezione sono elencate tutte le norme che il gruppo è tenuto ad osservare e rispettare per garantire il mantenimento di un rapporto utile e il più trasparente possibile con il proponente e i committenti per l'intera durata del progetto.

Il processo di fornitura definisce l'insieme di attività, compiti e risorse necessari al fornitore per portare a termine con successo il progetto. Il suo obiettivo principale è tracciare e descrivere le attività svolte dai membri del team di sviluppo. Questo tracciamento consente di valutare il lavoro completato, quantificare ciò che ancora deve essere realizzato e confrontare gli avanzamenti con le richieste del proponente, fornendo così una "istantanea" costante dello stato dei lavori e del bilancio di progetto.

Durante questa fase, il gruppo è tenuto a stabilire i contatti con il proponente e a definire i requisiti per il *MVP* (Minimum Viable Product) da concordare con quest'ultimo, basandosi su tempistiche, costi e importanza (intesa come incidenza sulla qualità del prodotto).

### 2.1.2) Rapporti con il proponente

Il team si impegna a mantenere contatti costanti con il proponente per l'intera durata del progetto. Questo impegno non solo assicura in parte il corretto sviluppo dei lavori, ma facilita anche lo scambio di informazioni e feedback tra le parti, contribuendo a garantire il rispetto degli accordi stabiliti. Nella sezione comunicazioni e nella sezione incontri è possibile consultare le norme e modalità con le quali avvengono questi contatti.

### 2.1.3) Documentazione fornita

In questa sezione viene presentanto un elenco contenente tutta la documentazione che deve venire fornita a proponente e/o commitente, per obblighi accademici o per volontà del team.

- Piano di progetto;

  Documento contenente la pianificazione delle attività di progetto, la gestione dell'organizzazione ruolistica ed il bilancio del progetto.

- Piano di qualifica;

  Documento contenente le normative e metriche riguardanti i processi di qualifica in adozione dal gruppo.

- Analisi dei requisiti;

  Documento contenente lo studio dei requisiti del prodotto software, loro classificazione, scenario e gestione.

- Glossario;

  Documento utile per favorire una corretta consultazione della documentazione di progetto. Viene accompagnato da una sua versione online disponibile nel sito vetrina del gruppo.

- Lettera di presentazione relativa;

  Documento accompagnante ogni revisione di progetto, che ne illustra brevemente il contenuto.

### 2.1.4) Strumenti

Segue un elenco degli strumenti utilizzati dal team per il processo di fornitura:

- Suite prodotti google (drive, documenti e fogli), per la condivisione di note, appunti e tabelle di funzione solitamente di tipo organizzativo;
- Google Meet e Zoom per gli incontri verso l'esterno;
- GitHub per l'hosting e versionamento del prodotto software (documentazione inclusa)
- KeyNote per la realizzazione dei diari di bordo.

## 2.2) Sviluppo

### 2.2.1) Descrizione e Scopo

In questa sezione del documento, vengono definite le linee guida e le norme essenziali atte a dirigere le attività di sviluppo in modo accurato e uniforme. L'obiettivo principale è garantire una coerenza completa nei metodi utilizzati, mirando al contempo a promuovere il raggiungimento di una qualità superiore nel prodotto finale. Questa sezione svolge un ruolo cruciale nel plasmare il processo di sviluppo, fornendo una struttura chiara e definita per le operazioni, che permette di mantenere standard solidi in tutte le fasi del ciclo di vita del progetto. La sua importanza sta nel contribuire alla creazione di un ambiente di lavoro orientato al raggiungimento di un risultato finale che soddisfi le aspettative e rispetti gli standard predefiniti.

Il processo di sviluppo di un prodotto software è suddivisibile come segue:
- Analisi dei requisiti;
- Design architetturale;
- Design del software;
- Programmazione e verifica;
- Integrazione.

### 2.2.2) Analisi dei requisiti

Il processo di analisi dei requisiti si colloca come prima fase dello sviluppo software, un momento cruciale in cui è imperativo delineare con precisione e robustezza gli scenari dei casi d'uso e le relative necessità o requisiti del sistema. Questa fase è di fondamentale importanza, perché implica la comprensione approfondita delle richieste degli utenti e degli stakeholder, offrendo una base solida per il progresso del ciclo di sviluppo.

Nel corso dell'analisi dei requisiti, il focus è posto sull'identificazione, la documentazione e la comprensione esaustiva delle funzionalità necessarie che il sistema dovrà incorporare.

Inoltre, durante l'analisi dei requisiti, è essenziale stabilire una comunicazione efficace con il proponente, al fine di garantire che tutte le prospettive e le esigenze rilevanti siano adeguatamente considerate.

#### 2.2.2.1) Casi d'uso

I *Casi d'uso* rappresentano azioni o sequenze di azioni collocabili in specifici scenari, caratterizzate da una richiesta e da una risposta. La loro definizione e realizzazione deve seguire gli standard imposti dal linguaggio UML, ed alcune regole interne al progetto.

Ogni caso d'uso deve essere così composto:
- identificazione e nomenclatura: il formato concordato è il seguente:

UC.x.y - Nome

dove: x, rappresenta il numero identificativo del caso d'uso generico; y, rappresenta l'id del sotto caso relativo. E "Nome" rappresenta il nome del caso d'uso da attribuire in modo chiaro e consono;

- figura;
- attori: utente o componente che può svolgere quella determinata azione o chiedere quel servizio;
- precondizioni: condizioni di utente e/o sistema, necessarie affinchè si verifichi quel caso d'uso/scenario;
- postcondizioni: rappresentano lo stato del sistema e/o utente, dopo che il caso d'uso è stato eseguito;
- scenario principale: in questa sezione si elencano le fasi che caratterizzano il caso d'uso;
- generalizzazioni (se presenti): in questa sezione vanno specificati ed elencati i possibili sotto casi d'uso;
- estensioni (se presenti): in questa sezione vanno elencate eventuali estensioni, nel caso ci possano essere degli scenari alternativi.

Per una migliore comprensione si riporta qui sotto un esempio nella forma standard da adottare per tutti i casi d'uso:

**UC1 - Login**

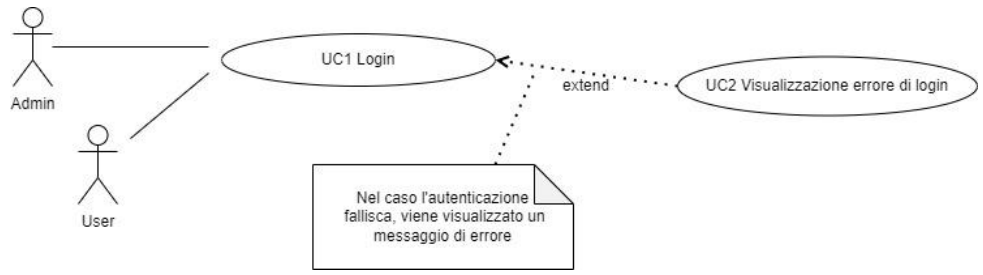


Figura 1: Login

**Attori:**
- Admin;
- Users.

**Precondizioni:**
- L'utente possiede un account valido;
- L'utente non ha già eseguito l'accesso;
- L'utente ha una connessione stabile.

**Postcondizioni:**
- L'utente ha effettuato correttamente l'accesso ed è stato riconosciuto dal sistema.

**Scenario principale:**
- Admin/Users:
  - inserisce la propria email nel campo [email] del modulo di accesso (UC1.1).
  - inserisce la propria password nel campo [password] del modulo di accesso (UC1.2).
- Sistema:
  - Il sistema di autenticazione verifica le credenziali inserite confrontandole con i dati memorizzati nel sistema.
  - Se le credenziali sono corrette, l'utente viene autenticato con successo e reindirizzato alla pagina principale.
  - Se le credenziali sono errate, il sistema di autenticazione visualizza un messaggio di errore per informare l'utente della fallita autenticazione (UC2).

**Generalizzazioni:**

- UC1.1 - Inserimento email.
- UC1.2 - Inserimento password.

**Estensioni:**
- UC2 - Visualizzazione errore di login

#### 2.2.2.2) Requisiti

I requisiti devono possedere un identificativo, composto come segue:

R[Importanza][Tipologia] X

Dove:

- Importanza: Indica il grado di importanza del requisito ed indirettamente la sua incidenza sul progetto. Un requisito può essere:

  1. O –> “Obbligatorio”;

     Un requisito deve venire defenito ed identificato come obbligatorio se ritenuto tale dal proponente, o se considerato tale e soddisfabile, dopo una cauta ed approfondita analisi, dal team di sviluppo.

  2. D –> “Desiderabile”;

     Un requisito deve venire defenito ed identificato come desiderabile, se ritenuto tale dal gruppo, in accordo con il proponente. Un requisito di questa categoria è da considerarsi soddisfabile in un secondo momento, a seconda di tempistche e costi.

  3. OPT –> “Opzionale”.

     Un requisito deve venire defenito ed identificato come opzionale, se ritenuto tale dal gruppo, in accordo con il proponente. Un requisito di questa categoria è da considerarsi di valore aggiunto per il prodotto, anche se secondario, ma insoddisfabile date tempistiche e costi.

- Tipologia: Indica il tipo di requisito in esame. Un requisito può essere:

  1. F –> “Funzionale”;

     I requisiti funzionali descrivono le funzionalità del sistema, le azioni che il sistema può compiere e le informazioni che questo può fornire.

  2. Q –> “di Qualità”;

     I requisiti di qualità descrivono come un sistema deve essere, o come il sistema deve esibirsi, per soddisfare le esigenze dell’utente.

  3. V –> “di Vincolo”;

     I requisiti di vincolo descrivono i limiti e le restrizioni che un sistema deve rispettare per soddisfare le esigenze dell’utente.

  Sono state successivamente concordate altre tre tipologie ritenute dal team, in seguito ad un’attenta analisi, di secondaria importanza data la natura del capitolato. Questi ultimi requisiti non saranno accompagnati da un identificativo, ma verranno posti nelle sezioni apposite in modo da non perderne la tracciabilità.

  1. “d’Ambiente”;

I requisiti d'ambiente si riferiscono alle condizioni e alle risorse necessarie per sviluppare, testare e implementare il software in un ambiente operativo specifico. Questi requisiti forniscono le specifiche riguardanti l'infrastruttura tecnologica e le configurazioni d'ambiente.

1. "di Performance";

I requisiti di performance definiscono le prestazioni e le caratteristiche di rendimento che il sistema deve raggiungere per soddisfare le aspettative degli utenti e del proponente.

1. "di Sicurezza".

I requisiti di sicurezza delineano le misure di sicurezza e i comportamenti attesi per proteggere il sistema da minacce esterne o interne.

#### 2.2.3) Progettazine

L'attività di progettazione segue ed utilizza la fase di analisi dei requisiti per definire ancor più struttura, vincoli e specifiche tecniche del prodotto software in oggetto. La fase di progettazione mira inoltre a facilitare definizione, suddivisione e quindi pianificazione delle attività di codifica del prodotto, beneficiando, se eseguita in modo corretto e vantaggioso, il ciclo di vita del software. La progettazione inizia con la creazione di un *PoC* (Proof of Concept), un prodotto software solitamente usa e getta, che mira a dimostrare la fattibilità del progetto. Durante questa prima fase, vengono scelte le tecnologie da adottare e viene abbozzata una prima struttura del prodotto, andando a definirne le parti, sempre con l'ausilio dei casi d'uso e relativi requisiti, analizzate nella fase precedente. Vengono infine identificate e concordate con il proponente le funzionalità considerate di maggiore importanza da sviluppare in questa prima bozza del prodotto. Successivamente, durante la fase adibita allo sviluppo di un MVP, verranno svolti studi più approfonditi sull'architettura del software in modo da migliorarne la qualità e manutenibilità generale. Tali studi verranno raccolti in un relativo documento allegato alla revisone di *PB* (Product Baseline).

##### 2.2.3.1) RTB

###### 2.2.3.1.1) Bozza di Architettura:

Le scelte concordate per la realizzazione del PoC sono le seguenti: il prodotto è suddivisibile in 5 Layer principali:

- Livello Dati: comprendente il Database relazionale, contente parte del dataset fornitoci dal proponente.
- Livello di Elaborazione: comprendente gli script Python adibiti alla gestione e applicazione, del algoritmo di raccomandazioni.
- Livello di Logica: è formato dalle *API* che permetteranno la comunicazione tra Webapp ed algoritmo, e dalla gestione di quest'utlime tramite script PHP.
- Livello di Presentazione: è composto dalla Webapp, che permetterà di utilizzare il prodotto al utente finale.

##### 2.2.3.2) Algoritmo di raccomandazione:

###### 2.2.3.2.1) Tecnologie scelte:

##### 2.2.3.3) PB

In questa sezione verrano in futuro definite le norme da seguire durante lo sviluppo relative alle scelte architetturali e di design prese nella fase di progettazione del MVP.

### 2.2.4) Programmazione e verifica del software

In questa sezione sono raccolte tutte le norme e regole che i programmatori in carico sono tenuti ad osservare durante il processo di codifica. La programmazione e relativa verifica, è una fase fondamentale, durante la quale chi ne è incaricato, inizia a plasmare e implementare il prodotto che l'utente finale andrà ad utilizzare.

- Linguaggi e ambiente:

Per lo sviluppo del prodotto il team userà vari linguaggi di programmazione a seconda di esigenze e vincoli, imposti sia dal capitolato che dal proponente. Se ne riporta qui sotto un elenco per una più facile consultazione:

1. Python, per l'agortimo di raccomandazione, e uso di sue svariate librerie come surprise, panda e numpy;
2. Mysql per la realizzazione del database relazionale;
3. Php per il backend della webapp;
4. React e Node.js per il frontend dell'interfaccia web.

Allo scopo di avere un ambiente coeso ed organizzato i componenti del gruppo sono tenuti ad utilizzare la configurazione creata appositamente con Anaconda durante lo sviluppo. Anaconda permette di creare configurazioni di Python per semplificare la gestione e la distribuzione di librerie e pacchetti.

- Stile di codifica:

Al fine di poter lavorare in un ambiente omogeneo ed ordinato il team ha deciso di predisporre delle automazioni che migliorino la qualità del codice ed organizzino la sua formattazione.

1. Python: Si è deciso di appoggiarsi a ruff, un sistema automatico di formattazione e analisi statica del codice. Esso è integrato nella repository tramite una GitHub action.

- Lunghezza e complessità:

Le funzioni e i metodi integrati nel codice del prodotto devono aderire rigorosamente agli standard di qualità stabiliti nel contesto del progetto. La filosofia che ogni membro del team si impegna ad adottare si concentra sull'incoraggiare il riuso del codice, sulla facilità del suo mantenimento e sull'ottimizzazione delle prestazioni. Segue un elenco dei principali principi guida:

1. Riuso del Codice: Ogni componente del gruppo è incoraggiato a sviluppare funzioni e metodi modulari che possano essere riutilizzati in diverse parti del progetto, sfavorendo la duplicazione del codice, e facilitando la manutenzione e la gestione delle funzionalità comuni.
2. Efficienza del Codice: Si presta particolare attenzione all'efficienza del codice. Le funzioni dovrebbero essere progettate in modo ottimale per garantire un'esecuzione efficiente. L'attenzione è rivolta alla complessità algoritmica, all'utilizzo appropriato delle risorse e alla minimizzazione di operazioni computazionalmente costose.
3. Testabilità: Ogni funzione dovrebbe essere progettata in modo tale da essere facilmente testabile. L'ideale sarebbe seguire un approccio *TDD* (Test Driven Development), quando e quanto più possibile.

### 2.2.5) Integrazione

La sezione dedicata all'integrazione del sistema e del software delinea il processo mediante il quale diverse componenti, moduli o servizi del progetto vengono combinati e testati per formare un sistema funzionante e coeso. L'obiettivo primario di questo processo è garantire che tutte

le parti del sistema operino in armonia, soddisfacendo i requisiti funzionali e di prestazione stabiliti.

## 2.3) Gestione operativa

### 2.3.1) Descrizione e Scopo

Questa sezione fornirà dettagli sull'installazione del software come i requisiti di sistema e le procedure necessari per eseguire correttamente il prodotto. Inoltre illustrerà una guida all'utilizzo, esponendo le principali funzionalità utilizzabili ed il come interagire con il sistema.

Per quanto riguarda il PoC si rimanda al relativo README.md disponibile nel repository.

### 2.3.2) Utilizzo operativo

### 2.3.3) Accettazione del cliente

## 2.4) Manutenzione

### 2.4.1) Descrizione e Scopo

### 2.4.2) Correzione

### 2.4.3) Adattamento

### 2.4.4) Evoluzione

# 3) Processi di supporto

## 3.1) Documentazione

### 3.1.1) Descrizione e Scopo

La documentazione software è l'insieme di informazioni, raccolte testualmente, volte allo scopo di spiegare a quali funzionalità assolve un software, come è strutturato e implementato e come lo si utilizza. Nel contesto del team di sviluppo è necessaria per facilitare il lavoro dei componenti, tenendo traccia e documentando tutti i processi e attività presenti andando a facilitare anche la manutenzione migliorando la qualità del risultato finale.

È bene quindi che vengano definite delle regole chiare e concise utili per la stesura di un documento, da seguire durante tutto il ciclo di vita del progetto allo scopo di garantire maggiore comprensione.

### 3.1.2) Strumenti

- Typst: scelto per la definitiva formattazione dei documenti per via della comodità con cui effettuare il versionamento dei documenti stessi;
- Overleaf (LaTeX): utilizzato nelle prime fasi del progetto per la realizzazione dei documenti necessari, successivamente sostituito con typst;
- UML: per la creazione di diagrammi UML il team ha deciso di utilizzare StarUML.

### 3.1.3) Grafiche

- Template: i documenti di progetto sono realizzati e caratterizzati da un preciso template, che differisce da quelli utilizzati per verbali e lettere di presentazione. Rispettivamente i file da includere per applicare il relativo template sono:
  1. big_docs.typ, per la documentazione ufficiale di progetto;
  2. verbals.typ, per i verbali, sia esterni che interni;
  3. p_letters.typ, per le lettere di presentazione.
- Tabelle: le tabelle presentano una classica intestazione del contenuto, i nomi delle colonne in grassetto e nessun altra particolarità, si è scelto di utilizzare una filosofia minimale per non appesantire i documenti. Si applicano colori alternati tra righe adiacenti in modo da facilitarne la lettura, rispettivamente "white" per le dispari, e "luma(230)"" per le pari. In alcune occasioni, è possibile se concordato, applicare il colore "c33435" alla riga di intestazione, per motivi puramente estetici.
- Figure: le immagini devono rigorosamente essere accompagnate da relativa didascalia, ed essere ridimensionate in modo ragionevole, senza eccessiva perdità di qualità e contenuto e senza andare ad occupare spazi esagerati.

#### 3.1.4) Norme tipografiche

- Nome file: I nomi dei file hanno tutti una notazione omogenea tra di loro, ovvero, nomi descrittivi del contenuto, la lettera iniziale è sempre maiuscola e il resto tutto minuscolo. Le parole sono separate da degli underscore e la data viene scritta in formato AAAA-MM-GG;
- Sezionamento: divisione in sezioni X.X.X e in caso di ulteriori suddivisioni si utilizza un elenco puntato, la sezione X.X.1 è sempre la descrizione del contenuto di quella sezione. Si cerca sempre di rendere il tutto più semplice possibile per facilitarne la lettura e mantenere ordinato il documento;
- Glossario:
- Stile del testo:

#### 3.1.5) Struttura

I documenti ufficiali hanno una struttura precisa e comune che deve essere rigorosamente rispettata per i motivi citati nella descrizione.

- Prima pagina: sempre composta dal template esclusivo del team, il logo dell'università, l'anno accademico in cui viene svolto il progetto, il nome del documento, il nome del team con la mail e i componenti;
- Registri modifiche (changelog): composti da versionamento, data della modifica effettuata, descrizione della modifica, ruolo dei componenti che hanno effettuato la modifica e i loro nomi.
- Indice: ogni documento presenta un indice nella pagina seguente al registro delle modifiche, la strutture è divisa in sezioni X.X.X con il numero della pagina in cui inizia la sezione. La divisione X.X.X presenta i macro-argomenti suddivisi nei loro vari paragrafi a loro volta suddivisi in sezioni più specifiche;
- Contenuto: esposto con la maggiore chiarezza e semplicità, rigorosamente diviso in sezioni secondo i principi di indicizzazione;
- Pié pagina: solamente il numero della pagina in questione ed il logo del gruppo separati dal contenuto da un' interlinea.

#### 3.1.5.1) Verbali

I verbali differiscono in quanto a struttura rispetto ai documenti di progetto. Vanno rispettate le seguenti sezioni:
Per i verbali esterni:

- Durata e partecipanti, sezione nella quale si elencano i partecipanti all'incontro, e si espone la durata del meeting;
- Oggeto, rappresenta gli argomenti discussi;
- Sintesi, riassume in breve il contenuto delle discussioni.

Per i verbali interni:

- Durata e partecipanti;
- Sintesi;
- Obiettivi fissati, elenco contenente gli obiettivi fissati per il prossimo periodo.

Inoltre i verbali interni relativi all'inizio di un nuovo sprint, dovranno possedere una sezione "Nuova distribuzione ruolistica" contenente una tabella (Ruolo, Nome e cognome) riportante i ruoli per il prossimo periodo.

### 3.1.6) Caratterizzazione

- Formali: Sono i documenti che andranno a formare la documentazione software del prodotto. In quanto tali sono sottoposti a versionamento e a processi di verifica e approvazione. Essi comprendono documenti interni, utili quindi ai membri del team di sviluppo, ed esterni, destinati a proponente e committente.

  Complessivamente ne fanno parte:

  - Interni:
    - Norme di progetto, rappresentano il "way of working";
    - Verbali interni e esterni, a uso consultativo;
  - Esterni:
    - Analisi dei Requisiti;
    - Piano di Progetto;
    - Piano di Qualifica;
    - Glossario;
    - Verbali interni e esterni, attestanti di quanto discusso.
- Informali: Sono i documenti interni non destinati alla divulgazione con esterni e fini a loro stessi. Perciò non necessitano di versionamento. Spesso sono bozze in preparazione a documenti formali, o note e appunti generiche.

## 3.2) Gestione della configurazione

### 3.2.1) Descrizione e Scopo

Il concetto di "gestione della configurazione" abbraccia tutte le pratiche essenziali per gestire lo stato di un prodotto software e di tutti i suoi componenti, compresi sorgenti e documentazione. Questo insieme di norme e procedure non solo fornisce informazioni sullo stato di avanzamento del progetto, ma offre anche un resoconto dettagliato dell'evoluzione nel tempo del prodotto, garantendo nel contempo che il sistema operi secondo le attese. Un'efficace gestione della configurazione è cruciale per preservare l'integrità e le prestazioni del prodotto software durante

il suo avanzamento. Inoltre, dovrebbe facilitare la risoluzione di problematiche e conflitti, assicurando una gestione fluida e efficiente del ciclo di vita del software.

### 3.2.2) Versionamento

Il versionamento è una procedura fondamentale per la gestione di un progetto. Oltre a tracciare i cambiamenti di ogni *artefatto*, documento o sorgente che sia, permette il rispristino di quest'ultimo ad una sua fase precedente rendendo molto più semplice la gestione di errori. Il changelog o "registro delle modifiche", strettamente collegato al concetto di versionamento, espone al lettore, il ciclo di vita dell'artefatto, le modifiche effettuate, le problematiche sorte, e infine anche la distribuzione dei lavori tra i componenti del team di sviluppo.
Ogni documento oltre a essere dotato di un changelog è identificato da un numero di versione così composto:

vX.Y.Z

dove :

- X rappesenta fasi del documento che suddividono e raccolgono i cambiamenti più significativi apportati all'artefatto anche detti "major".
- Y rappresenta modifiche minori come ad esempio la realizzazione di una sezione o feature le quali si pensa non siano sufficienti a stabilire una nuova "fase" del documento. Sono anche identificati con l'appellativo "minor".
- Z rappresenta piccoli aggiustamenti (fixes) o migliorie generali.

Si noti che ogni versione rappresenta non solo un'aggiunta di tipo prettamente produttivo, ma anche al sua revisione.

#### 3.2.2.1) Lato documentazione

Nella documentazione è possibile aggiornare la versione andando semplicemente ad aggiungere un nuovo record nella sezione di changelog del rispettivo file sorgente. Qui sotto un esempio:

```
changelog: (

    "0.5.0", "21-11-2023", p.baggio, p.carraro, "Stesura sezione 3.2",
    "0.4.0", "20-11-2023", p.passarella, p.carraro, "Stesura sezione 3.1",

)
```

Una volta fatto, la compilazione automatica, attuata grazie ad una github action realizzata ad hoc, insieme alle funzionalità di scripting che fornisce typst, andrà a creare effettivamente la tabella del registro delle modifiche con all'interno tutte le informazioni specificate e richieste. Si noti che *p* è una variabile d'ambiente contenente tutti i nominativi dei componenti del gruppo di lavoro e di ulteriori nominativi utili e ripetuti molteplici volte nel corso del progetto.

#### 3.2.2.2) Lato software

### 3.2.3) Repository

Per la gestione della configurazione e versionamento il progetto si poggia sul uso di un repository Github. Qui sotto un link alla documentazione ufficiale:

Github Docs.

#### 3.2.3.1) Struttura

L'attuale struttura del repository è suddivisa in 3 branch:

- main;
- approval;
- sources.

main:
E' definibile come il branch di presentazione, nel quale sono presenti solo artefatti revisionati e approvati dal responsabile di progetto corrente. Su esso è applicata una "branch protection rule" che non ne permette i push diretti e protegge il ramo.

approval:
Come intuibile dal nome, è il branch che rappresenta il main durante lo sviluppo e che garantisce che ciò che entra nel ramo principale sia completamente revisionato e approvato. I suoi contenuti verrano uniti a quelli del main tramite un processo di merge una volta che il responsabile di progetto lo ritenga possibile. Le pull request da qualsiasi ramo verso quello di presentazione verranno infatti reinderizzate a quest'ultimo, che ne andrà a valutare la qualità, accettando il lavoro svolto o rimandandolo al mittente con direttive sul come migliorarlo.

sources:
E' il branch relativo alla produzione della documentazione, perciò contiene solo file di tipo .typ e non è pensato per una sua supervisione esterna. I file sorgenti verranno compilati e resi disponibili automaticamente nel ramo approval.

Nel branch main è disponibile un README.md che ne descrive la struttura di cartelle. Qui sotto un link al repository:

Repository di progetto.

## 3.3) Accertamento della qualità

### 3.3.1) Descrizione e Scopo

La qualità di progetto è l'iniseme delle attività e di accorgimenti eseguiti durante il suo sviluppo per garantire che le performance del progetto siano congruenti con gli obiettivi e i requisiti stabiliti. Questo implica l'attuazione di misure e controlli volti a garantire che gli output del progetto soddisfino gli standard di qualità prefissati e rispettino le aspettative stabilite nel contesto del progetto stesso. La gestione della qualità è un elemento chiave per assicurare il successo complessivo del progetto e la soddisfazione delle parti interessate.
Le attività messe in atto per garantirne la qualità sono:

- Rispettare e comprendere le necessità del proponente, sia in termini di quantità che di qualità, in modo da rilasciare un prodotto il più possibile fedele all'idea del cliente.
- Seguire il piano di qualifica per il sviluppo del progetto, in modo da rientrare nei termini pattuiti, di tempo e di costo.

## 3.4) Qualifica

### 3.4.1) Verifica

La verifica del software è definita come il processo di valutazione del prodotto software, mirato a garantire l'accuratezza dell'esecuzione della fase di sviluppo per costruire il prodotto software desiderato. Questa attività si svolge durante la fase in corso di sviluppo per rilevare difetti e guasti nella fase iniziale del ciclo di vita dello sviluppo, assicurando al contempo che il prodotto

soddisfi i requisiti del cliente. La verifica quindi, è un elemento cruciale per garantire la qualità e la conformità del software alle specifiche stabilite.

#### 3.4.1.1) Analisi statica

La verifica statica, così chiamata poichè non richiede l'esecuzione del prodotto, consiste nell'individuazione e correzione di eventuali problematiche che riguardano convenzioni o metriche stabilite. L'analisi statica riguarda sia il codice del software che la stesura dei documenti e può essere effettuata sia in maniera manuale o grazie all'utilizzo di strumenti per l'automazione.
Esistono inoltre due tipi per la verifica tramite l'analisi statica:

- Walkthrought: viene utilizzato nel momento in cui non si sappia dove viene riscontrata la problematica e consiste in una lettura più ampia scorrendo nella sua interezza il documento/ codice per trovare l'errore. Questo metodo è sicuramente molto efficace ma anche molto dispendioso in termini di risorse.
- Inspection: A differenza del Walkthrought, questo metodo viene utilizzato quando si è a conoscenza di dove potrebbe essere la problematica. E' quindi un approccio più mirato per l'eliminazione dell'errore e molto meno dispendioso.

#### 3.4.1.2) Analisi dinamica

L'analisi dinamica è un tipo di verifica che viene fatta grazie all'esecuzione del prodotto, atto a identificare errori e controllare il corretto funzionamento.

### 3.4.2) Validazione

Lo scopo della validazione è quello di confermare la qualità del prodotto software nella sua interezza, assicurando che i requisiti siano stati implementati correttamente come concordato con il proponente.
Perchè un file venga validato, c'è la necessità che passi i test preposti in base al suo tipo, confermando e attestando la qualità del prodotto.

#### 3.4.2.1) Validazione statica

Per la documentazione è stato implementato un GitHub Actions che verifica la correttezza semantica e sintattica del codice Typst per poi andare a costruire il file in formato pdf da esso. Per quanto riguarda i file di codice, sempre tramite Actions, sono stati implementati, grazie a ruff, sistemi automatici legati al codice del file per la formattazione del codice.

#### 3.4.2.2) Validazione dinamica

## 3.5) Revisioni congiunte con il cliente

### 3.5.1) Descrizione e Scopo

Le revisioni congiunte con il cliente rappresentano un importante aspetto del processo di sviluppo del software, permettendo un coinvolgimento diretto e trasparente con il cliente nell'avanzamento del progetto.
Durante queste sessioni collaborative, gli sviluppatori presentano al cliente gli aggiornamenti, le nuove funzionalità o le modifiche apportate al software. Questo processo consente al proponente di esaminare il prodotto in modo attivo, fornendo feedback tempestivi e suggerimenti oltre ai possibili chiarimenti di natura tecnica o strutturale. Le revisioni congiunte offrono un'opportunità cruciale per allineare le aspettative del cliente con la realizzazione pratica del software,

identificare eventuali incongruenze e garantire che il prodotto finale soddisfi appieno le esigenze e le aspettative del cliente.

## 3.6) Verifiche interne

### 3.6.1) Descrizione

## 3.7) Risoluzione dei problemi

### 3.7.1) Descrizione e Scopo

Durante il progetto il nostro team ha sempre cercato di operare seguendo un modello di miglioramento continuo, in modo da applicare alle best practices, le soluzioni incontrate fino ad ora alle problematiche più riccorrenti.
La risoluzione dei problemi gioca quindi un ruolo fondamentale per l'avanzamento del progetto didattico in quanto permette di monitorare e risolvere i problemi che possono sorgere durante lo sviluppo o il testing.

### 3.7.2) Gestione dei problemi

Il nostro gruppo per risolvere problematiche verificate durante lo sviluppo del progetto ha deciso di adottare una strategia sviluppata in quattro punti:
- Registrazione del problema;
- Valutazione del problema;
- Risoluzione del problema;
- Comunicazione e aggiornamento documentazione.

#### 3.7.2.1) Registrazione del problema

Ogni problema identificato deve essere accuratamente registrato nel sistema di tracciamento dei problemi (ITS). Questo sistema consente di inserire informazioni dettagliate, tra cui:
- Descrizione: breve descrizione, in maniera da far capire il problema che si è riscontarto;
- Stato: indica a che punto si trova la risoluzione del problema: aperto, in lavorazione, pronto per una revisione o chiuso;
- Assegnatario: persona a cui è stata assegnata la risoluzione del problema.
- Data: indica la data in cui è stata aperta la issue;
- Label: indica di cosa si deve occupare la persona preposta, quindi se il problema coinvolge la documentazione o codice.

#### 3.7.2.2) Valutazione del problema

La valutazione del problema è una fase che precede la risoluzione e prevede un attenta valutazione da parte del gruppo. Questa fase rigurda prettamente problematiche riguardanti il codice la cui discussione avviene attraverso meeting interni. Durante gli incontri si toccano tre temi principali: la causa, l'impatto e il rischio.

#### 3.7.2.3) Risoluzione del problema

Una volta valutato il problema, sarà la persona incaricata a risolverlo, comunicando tramite il ITS lo stato di avanzamento della risoluzione al problema e sfruttando la strategia valutata precedentemente.

#### 3.7.2.4) Comunicazione e aggiornamento documentazione

Durante il processo di gestione del problema, è fondamentale mantenere una comunicazione trasparente con il proponente. Questo include la creazione di report periodici sullo stato dei problemi e meeting di aggiornamento.
Deve inoltre, se necessario, essere aggiornata la documentazione in maniera da mantenere una coerenza tra software e documenti.

#### 3.7.3) Gestione dei cambiamenti

La gestione dei cambiamenti è un processo cruciale all'interno dello sviluppo del software, poiché consente di gestire in modo efficace e controllato le modifiche apportate al progetto. Il tener traccia dei cambiamenti è fondamentale in ogni parte del ciclo di vita di un prodotto.
La documentazione dettagliata di ciascun cambiamento, insieme alla sua motivazione contribuisce a costruire una base solida per la manutenzione futura. La gestione dei cambiamenti viene controllata attraverso il changelog presente in ogni documento, e inoltre riportato nel nome del file la sua versione corrente come spiegato nella sezione del versionamento

# 4) Processi organizzativi

## 4.1) Gestione dei processi

#### 4.1.1) Descrizione

In questa sezione vengono trattate tutte le normative utili alla stesura e redazione del documneto "Piano di progetto", gli argomenti che ne fanno parte sono:

- Ruoli e relativa organizzazione: con conseguente descrizione ed esposizione della strategia utilizzata per la rotazione degli stessi.
- Gestione degli incontri e delle comunicazioni: comprendente loro suddivisone in tipologia.
- Gestione dell'organizzazione: informazioni relative alle pratiche per la suddivisone in attività, e loro collocazione temporale, del corso del progetto.

L'obiettivo dei processi organizzativi è quello di arrivare alla creazione del documento denominato "Piano di progetto" nella sua forma il più completa possibile, andando a definire ruoli, attività e loro collocazione nel tempo. Il documento, oltre ad essere utile al team per gestire l'organizzazione e la gestione dei ruoli di ogni componente, fa da bacheca al commitente di quanto appena citato. Il piano di progetto punta a comprendere tutte le pratiche e metodi riguardati il processo organizzativo e di pianificazione, descrivendone l'applicazione.

#### 4.1.2) Ruoli e relativa organizzazione

La suddivisione in ruoli segue le norme definite nel "Regolamento progetto didattico" sottopostoci dal nostro committente. Qui sotto un breve riepilogo:

| Ruolo | Costo orario | Responsabilità |
|---|---|---|
| Responsabile | 30 | • Coordina l'elaborazione di piani e scadenze<br>• Approva il rilascio di prodotti parziali o finali (SW, documenti)<br>• Coordina le attività del gruppo |
| Amministratore | 20 | • Assicura l'efficienza di procedure, strumenti e tecnologie a supporto del way of working |
| Analista | 25 | • Svolge le attività di analisi dei requisiti |

| | | |
|---|---|---|
| Progettista | 25 | • Svolge le attività di progettazione (design) |
| Programmatore | 15 | • Svolge le attività di codifica |
| Verificatore | 15 | • Svolge le attività di verifica |

Si noti come i ruoli possano svolgere anche mansioni al di fuori della loro responsabilità in caso di necessità, ovviamente senza venire meno alle pratiche di tracciabilità adottate normalmente dal team.

La loro assegnazione viene gestita dal Responsabile di progetto corrente, il quale confrontandosi con gli altri componenti del gruppo, va a stabilire una rotazione conforme al regolamento. Ogni membro del team dovrà infatti ricoprire ogni carica almeno una volta.

Segue una descrizione più dettagliata per alcuni ruoli, perchè ritenuti più complessi, e rispettive mansioni:

- Responsabile di progetto:

  Il Responsabile si occupa oltre alle attività già sopra elencate, di:
  1. Gestire il repository;
  2. Gestire le risorse umane;
  3. Comunicare con l'esterno, proponente e committenti;
  4. Gestire le comunicazioni e incontri interni;
  5. Gestire imprevisti;
  6. Gestire il bilancio del progetto.

- Amministratore:

  L'Amministratore si occupa oltre alle attività già sopra elencate, di:
  1. Redigere e attuare i piani e le procedure per la gestione della qualità.

- Verificatore:

  Il verificatore si occupa oltre alle attività già sopra elencate, di:
  1. Dare feedback completi e chiari a chi ha prodotto il lavoro successivamente revisionato.
  2. Assicurare che la qualità di quanto prodotto sia conforme agli standard imposti.

#### 4.1.2.1) Gestione dei "cold start"

Al fine di evitare rallentamenti durante il corso del progetto, dovuti a delle situazioni di "cold start", il team si impegna ad adottare le seguenti pratiche:

- Documentazione dettaglita:

  Ogni membro è tenuto a documentare ogni azione ritenuta non banale e avente valenza e dipendenze future. Prima in documenti informali, questo per non rallentare troppo i tempi, successivamente da integrare nella documentazione ufficiale.

- Formazione e Condivisione delle Conoscenze:

  Ogni membro è tenuto, qualora si ritenga necessario, a condividere, oltre che in forma scritta attraverso la documentazione, verbalmente le conoscenze e le competenze apprese durante gli sviluppi, o pregresse.

- Rotazione Graduale:

  Le rotazioni dei ruoli, quando ritenute necessarie, avverranno in modo graduale. Ciò consentirà a coloro che hanno già sviluppato una certa dimestichezza in un determinato ambito

di supportare chi si avvicina a quel ruolo per la prima volta. In pratica, questa modalità di rotazione riflette l'approccio XP, emulando la pratica del "*pair programming*.

#### 4.1.3) Gestione degli incontri e delle comunicazioni

##### 4.1.3.1) Reperibilità dei membri

Ogni membro del gruppo si impegna ad essere reperibile per riunioni sincrone durante la settimana, dal lunedì al giovedì, nel pomeriggio, il venerdì durante la mattinata. In caso di impossibilità di partecipare alle riunioni nelle date stabilite, è obbligatorio informare tempestivamente il Responsabile di progetto. Inoltre, la disponibilità può essere estesa anche durante il weekend in casi di necessità, solitamente preferendo la domenica al sabato.

Durante il corso di uno sprint ogni mebro è libero di gestire le proprie attività di progetto in modo asincrono, a meno che esse non richiedano la collaborazione di più componenti. Ogni membro si assume responsabilmente la gestione di impegni accademici e personali, rispettando le scadenze imposte dal relativo sprint.

Per facilitare l'assegnazione delle attività in relazione agli impegni di ogni componente, il team ha a disposizione un file "Google Fogli" dove sono visualizzabili le proprie disponibilità giornaliere (inserite al inizio del progetto), e dove nel eventualità è possibile segnare altri impegni inderogabili e sorti in un secondo momento.

##### 4.1.3.2) Comunicazioni

- **Interne**

  Le comunicazioni interne sono quelle coinvolgenti solo il team, o alcuni dei suoi membri, avvengono tramite mezzi quali:

  – Telegram: Il gruppo è usato per comunicazioni di tipo più breve e repentino. Solitamente viene utilizzato per stilare un breve ordine del giorno, e per organizzare incontri interni. Mentre in privato avvengono comunicazioni che interessano solo le parti conivolte in modo da non intasare il gruppo.

  – Discord: Il server è suddiviso in vari canali testuali:
  - General: utilizzato per comunicazioni generali;
  - Link-utility: in esso sono riversati tutti i link utili al gruppo perchè spesso visitati o consultati (ad esempio: link alla tabella condivisa per la gestione costi/ore);
  - Link-brainstorming: racchiude tutti i link a fonti di tipo informativo (ad esempio: link alla documentazione della libreria di python surprise);
  - Domande: è utilizzato per contenere le domande dei vari componenti del team, sia rivolte verso il team stesso, si verso l'esterno (proponente/committenti);
  - Todo-reminder: contiene delle annotazioni su cose da fare nel breve periodo.

  Sono inoltre presenti molteplici canali vocali per permettere di lavorare in piccoli sottogruppi.

- **Esterne**

  Le comunicazioni esterne avvengono tramite i seguenti mezzi:

  – E-mail: Usate per le comunicazioni con proponente e commitenti. Principalmente hanno la funzione di concordare meeting, o di esporre quesiti e dubbi.

##### 4.1.3.3) Incontri o Meetings:

- **Interni**

Per una migliore gestione degli imprevisti e in generale della pianificazione e organizzazione delle attività, il gruppo ha deciso di adottare 2 tipologie differenti di incontri interni: “Scheduled Meeting”, e “Daily Call”. Per questioni di efficenza e praticità si è concordato di adoperare Discord come mezzo tramite, la modalità di questi incontri è quindi “da remoto”.

- **“Scheduled meeting”**

  Sono i meeting interni che solitamente prevedono la messa a verbale. Vengono fissati con cadenza settimanale con data variabile a seconda delle disponibilità dei membri del team, quest’ultima viene regolarmente concordata alla fine del incontro precedente. La loro durata è variabile, e tutte le componenti sono tenute a presenziarvi. Per una migliore gestione del tempo a disposizione è stato deciso di strutturare i meeting come segue:

  – Struttura:

  1) Ordine del giorno: Al inizio di ogni meeting chi ha partecipato alle lezioni del giorno, condivide informazioni utili con il team, aggiornadolo su quanto stato discusso con i docenti.

  2) Retrospettiva: Prima di discutere le varie domande e dubbi, il responsabile di progetto corrente stila un breve riassunto di quanto stato fatto nel ultimo sprint, evidenziandone aspetti positivi e negativi. Per facilitare queste operazioni, il responsabile di progetto è tenuto ad arrivare “preparato” al incontro, in modo che sia tutto più scorrevole.

  3) Domande e dubbi: Successivamente vengono discusse domande e dubbi per le quali si ritenga necessaria una discussione collettiva.

  4) Pianificazione prossima: Infine viene effettuata, avendo un miglior cruscotto sul avanzamento del progetto, una pianificazione più mirata per il prossimo sprint, andando a definire effettivamente le attività, ruotando i ruoli e spartendo queste ultime.

  5) Post meeting: Alla fine di ogni meeting il responsabile di progetto fa un breve resoconto di quanto discusso e lo condivide sul gruppo telegram in modo da aggiornare chi eventualmente non fosse riuscito a presenziare al incontro. Successivamente si adopera anche ad una prima stesura del relativo verbale. Per facilitare queste operazioni ogni meeting interno viene registrato.

- **“Daily Call”**

  Sono incontri di durata mediamente minore, che avvengono giornalmente quando e se ne sorge la necessità. Possono essere richiesti da qualsiasi membro del gruppo, e la partecipazione è richiesta solamente ai sottoinsiemi coinvolti. Solitamente non prevedono la stesura di relativo verbale, ma ciò dipende dagli argomenti discussi e dalla presenza o meno di decisioni importanti. Vengono anche utilizzati per sessioni di lavoro “in pair”.

- **Esterni**

  – Proponente:

  Questi incontri prevedono sempre la stesura di relativo verbale necessitante validazione ed approvazione dal partecipante esterno attraverso la sua firma. Solitamente vengono richiesti dal team. Data e orario, sono concordati a priori durante il meeting precedente, o tramite E-mail, tenendo conto delle disponibilità del proponente e del gruppo. Gli argomenti trattati sono di vario tipo e seguono gli sviluppi del progetto, per facilitarne la discussione vengono esplicitati al proponente tramite mail alcuni giorni prima del incontro. Il mezzo varia a se-

conda delle necessità del proponente, solitamente vengono utilizzate piattaforme come Zoom e Google Meet.

– Committenti:

nd.

#### 4.1.4) Gestione dell'organizzazione

##### 4.1.4.1) Metodologia e pratiche

In modo da migliorare la collaborazione e una migliore gestione del ritmo di avanzamento dei lavori, il team ha preso la decisione di adottare un approccio agile nello sviluppo del progetto, ispirandosi a framework e metodologie ben consolidati come Scrum e XP, ampiamente utilizzati in contesti lavorativi reali.

La filosofia che sottende le strategie di tipo agile è incentrata sull'adozione di pratiche di *Continuous Integration/Continuous Deployment* (CI/CD).

Questa scelta mira a fornire diversi vantaggi e valori aggiunti:

- Favorire il Lavoro di Gruppo:

  Promuove la collaborazione e la comunicazione all'interno del team, incoraggiando la condivisione delle idee e delle competenze.

- Sviluppo Individuale:

  Favorirsce la crescita individuale a livello di conoscenze e competenze, consentendo a ciascun membro di contribuire al massimo delle proprie capacità indipendentemente dal ruolo corrente.

- Miglioramento Continuo:

  Promuove il miglioramento continuo attraverso pratiche di retrospezione, identificando e risolvendo le problematiche in modo tempestivo e continuo.

- Organizzazione Efficace:

  Migliora e facilita l'organizzazione tra i membri del team, assicurando una distribuzione efficace delle responsabilità e delle attività.

- Trasparenza per Proponenti e Committenti:

  Garantisce trasparenza al proponente, consentendo un costante flusso di feedback e una maggiore comprensione del processo di sviluppo. Facilita anche l'analisi da parte del committente per una migliore valutazione del progresso.

L'adozione di pratiche CI/CD si inserisce in questo contesto, contribuendo a garantire una integrazione continua del codice e una distribuzione continua delle nuove funzionalità, riducendo al minimo rischi e migliorando la qualità del software. Questo approccio agile mira a fornire al team una struttura dinamica e flessibile per affrontare le sfide e rispondere alle esigenze del progetto in modo efficiente e tempestivo.

##### 4.1.4.2) Milestone e Sprint

Le tempistiche del periodo di progetto sono scandite da milestone e sprint. Il team, sempre rifacendosi ad un approccio di tipo agile, ha definito conseguentemente queste ultime.

- Milestone:

  Rappresentano le revisioni di progetto, e gli sprint necessari al loro compimento. Ragionevolmente, ogni sprint verrà a posteriori suddiviso in macro attività pianificandone le fasi, la cui granularità e specificità verranno esplorate durante un periodo più a ridosso dello stesso, avendo un istantanea migliore dello stato del progetto.

- Sprint:

  Definiscono finestre di tempo durante le quali il team lavora per portare a termine relative attività, rispettandone le scadenze. La durata degli sprint deve essere tassativamente fissata, in modo che ogni componente percepisca l'incobenza delle scadenze e quindi si adegui di conseguenza, evitando così "pigrizie". Questo senza andare a discapito della salute mentale di ogni componente, la quale andrebbe ad incidere sulla qualità di quanto prodotto. Solo in casi eccezionali, come durante gli inizi del periodo relativo al RTB, considerato, di "assestamento", sono ammesse delle variazioni, perlopiù per facilitare l'adeguamento di ogni membro al respettivo ruolo corrente. In modo da mantenere un workflow scorrevole, si è concordata la durata di 1 settimana per sprint. Così facendo ogni membro ha la possibilità di esplorare più ruoli, e di gestire il proprio tempo in modo più produttivo.

La loro definizione e collocazione temporale è definita nel documento "Piano di progetto", redatto dal responsabile di progetto.

#### 4.1.4.3) Gestione di attività e Issue

Per una migliore gestione delle attività di progetto vengono suddivise in due categorie differenti:

- Macro Issues: Rappresentano le prime attività individuate dal team durante la prima fase di pianificazione generale. Saranno ragionevolmente soggette a modifiche durante il corso del progetto.
- Task: Rappresentano le attività più specifiche individuate a ridosso di un nuovo sprint, e quindi ragionate e definite avendo un cruscotto più chiaro sullo stato di avanzamento del progetto.

Per la gestione delle attività relative ai processi primari e di supporto, si utilizza il sitema integrato di *Issues Tracking System* (ITS), di Github. Il ciclo di vita delle azioni segue i seguenti passaggi:

Creazione: L'attività viene definita come un compito da svolgere e viene registrata come una issue su GitHub. Le issue devono essere collegate alla/e board di progetto, ed al rispettivo sprint utilizzando la milestone allegata. Inoltre devono essere contrassegnate da label identificative consone. Ne segue la lista che ne norma l'uso:

- approval: da utilizzare solo per issue che identificano attività di approvazione, da svolgere dal Responsabile corrente. Solitamente una finale per sprint.
- bug-fix: denota una issue la cui rispettiva attività mira alla correzione di un bug, o altre problematiche.
- code: rappresentano una nuova feature o integrazione nel codice.
- documentation: rappresentano issue legate alla stesura di documentazione.
- revision: da utilizzare solo per issue legate alle attività di revisone.
- RTB: deve essere usata solo per issue rappresentanti sprint legati al periodo di RTB.
- PB: deve essere usata solo per issue rappresentanti sprint legati al periodo di PB.

Infine le issue devono avere un nome significativo e possedere una descrizione definita come segue:

(h3 in markdown) desc: (plain text) testo della descrizione.

Assegnazione: Le issue vengono assegnate in modo da rispettare la configurazione ruolistica corrente. Il responsabile si occupa di svolgere questo compito al inizio di un ogni sprint.

Completamento: L'attività viene completata dalla persona incaricata, per poi essere spostata nello stato "ReadyToReview" nella rispettiva board di progetto, in modo da notificarne la revisione. Successivamente verrà chiusa attraverso l'apposita funzionalità di chiusura delle issue in GitHub da chi ne svolge la revisone.

Verifica: Chi ne è incaricato procede a verificare quanto svolto utilizzando strumenti automatici o meno, a seconda di quanto prodotto (Documentazione, codice, …). Nel caso in cui il verificatore non sia soddisfatto del lavoro svolto, e vi siano grandi modifiche impossibili da apportare da quest ultimo, informerà l'autore e il Responsabile. L'autore dovrà quindi ritornare su quanto fatto e apportare le modifiche suggerite dal verificatore. Una volta finito con esito positivo il processo di verifica il verificatore procederà chiduendo sia la issue relativa all'attività corrisponendete, che la issue relativa alla verifica di tale attività.

Si noti che è stato scelto di avere una issue specifica per la verifica, per rendere ai verificatori più facile organizzare e pianificare il proprio lavoro al inizio di ogni sprint.

Accettazione: Dopo la conferma positiva da parte del Verificatore, a ridosso della fine di ogni sprint, il Responsabile effettua un controllo aggiuntivo, procedendo quindi a chiudere l'issue di approvazione corrispondente, e a eseguire il merge nel branch principale di presentazione.

Le singole attività vengono valutate in base alla loro dimensione e alla pianificazione definita, considerando sia il carico di lavoro che le responsabilità associate.

Una più dettagliata descrizione della gestione delle board di progetto è visionabile nel prossimo sotto-paragrafo.

#### 4.1.4.4) Gestione delle dashboard / Github views.

Per una migliore gestione e visualizzazione delle attività di progetto, le issue devono, al momento della loro creazione, essere "linkate" al rispettivo Github Project (RTB, PB). Per ogni Project sono state predisposte due view differenti:

- Kanban:

  Rappresenta una dashboard, suddivisa in 4 colonne differenti:
  1. Todo: qui si trovano le issue ancora non iniziate;
  2. In progress: in questa sezione vi sono tutte le issue il cui svolgimento sta avanzando;
  3. ReadyToReview: questa colonna è adibita alle attività necessitanti una revisione;
  4. Done: qui si trovano tutte le issue completate.

  Al fine di un utilizzo utile della board ogni membro del gruppo è tenuto a gestire le proprie attività di progetto e a tenere sotto controllo la Project view in modo da avere sempre una visione dello stato si avanzamento dello sprint.

- Gantt:

A differenza di quella precedente, mira a fornire una rappresentazione calendarizzata delle attività di progetto. Ogni issue viene collocata temporalmente in base alla sua data di "presa a carico" (ovvero nel momento in cui il suo stato cambia da todo, a in progress) come inizio e alla sua data di completamento (inclusa la revisione) come fine. Questo approccio consente una visione più chiara e strutturata delle tempistiche di ciascuna attività.

Per una comprensione ancora più approfondita dell'insieme, la board viene suddivisa in milestone, consentendo la visualizzazione singola di ciascuna di esse. Questa suddivisione facilita il monitoraggio e la gestione specifica di ciascun obiettivo intermedio, migliorando ulteriormente la chiarezza e la gestione del progetto nel suo complesso.

## 4.2) Infrastrutture

L'infrastruttura organizzativa costituisce l'insieme degli strumenti impiegati per facilitare e ottimizzare i processi organizzativi, consentendo uno svolgimento pratico ed efficiente delle attività lavorative. Questa struttura svolge un ruolo cruciale nel fornire il supporto necessario per garantire che le operazioni quotidiane come i processi più complessi vengano gestiti con successo.

Questa infrastruttura può comprendere una vasta gamma di elementi, tra cui software specializzato, piattaforme tecnologiche, e strumenti di comunicazione.

### 4.2.1) Strumenti di supporto ai processi

- GitHub:

  E' un servizio di hosting per progetti software, implementazione dello strumento di controllo versione distribuito Git. Viene utilizzato per gestire tutti gli artefatti del progetto.

- Telegram

  Telegram è un servizio di messaggistica istantanea e broadcasting basato su cloud. Viene utilizzato per le comunicazioni interne fra membri del team.

- Discord

  E' una piattaforma di VoIP, messaggistica istantanea e distribuzione digitale. Viene utilizzata per incontri interni fra membri del team

- Google Drive, e suo ecosistema: Fogli e Documenti.

  Google Drive è un servizio web, in ambiente cloud computing, di memorizzazione e sincronizzazione online. Viene utilizzato per la gestione di file informali, che variano tra note e appunti realizzate utilizzando Documenti, e tabelle di vario tipo e uso, realizzate con Fogli.

- Google Meet e Zoom

  Sono applicazioni di teleconferenza. Vengono utilizzate principalmente per incontri esterni.

- Gmail

  E' un servizio di posta elettronica, utilizzato per le comunicazioni con esterni.

## 4.3) Miglioramento

Durante lo svolgimento delle attività e la successiva elaborazione della documentazione, ci poniamo come obiettivo quello di cercare di seguire costantemente il principio di miglioramento continuo. L'obiettivo principale è quello di identificare in modo proattivo le attività, i ruoli

e le opportunità di miglioramento, cercando nuove o alternative soluzioni per affrontare sfide emergenti, o passate. Questo approccio contraddistinguerà ogni fase del processo, dalla pianificazione, all'esecuzione e alla documentazione.

In sintesi, la manutenzione migliorativa dei processi è un ciclo iterativo che ci consente di adattarci dinamicamente alle esigenze mutevoli del progetto, garantendo una crescita continua e una maggiore efficienza nelle nostre attività.

## 4.4) Formazione

Per promuovere un miglioramento continuo nelle attività svolte e garantirne il corretto mantenimento, è essenziale che ogni membro del team attraversi un periodo di formazione in autonomia, che si protragga durante tutto il corso del progetto in modo asincrono. Questo periodo di formazione mira a fornire a ciascun membro le competenze necessarie per comprendere e utilizzare in modo efficace le tecnologie e le metodologie operative coinvolte nel progetto.

### 4.4.1) Complementi all'auto-formazione

- GitHub: https://docs.github.com/en
- Git: https://git-scm.com/doc
- Framework Scrum: https://www.atlassian.com/it/agile/scrum
- Approccio XP:

  https://www.ionos.it/digitalguide/siti-web/programmazione-del-sito-web/extreme-programming