SWEnergy

project.swenergy@gmail.com

# Norme di progetto

**Descrizione:** Elenco delle procedure interne e delle buone pratiche di progetto adottate dal gruppo.

| | |
|---|---|
| **Stato** | Non approvato |
| **Data** | 21/11/2023 |
| **Redattori** | Davide Maffei |
| **Verificatore** | Niccolò Carlesso |
| **Approvatore** | Nessuno |
| **Versione** | 1.4.1 |

Il responsabile: Nessuno

_____

# Registro delle modifiche

| Versione | Data | Redattore | Verificatore | Approvatore | Descrizione |
|---|---|---|---|---|---|
| 1.4.1 | 14/02/2024 | Davide Maffei | Nessuno | Nessuno | Allineamento delle sezioni dei ruoli |
| 1.4.0 | 14/02/2024 | Davide Maffei | Nessuno | Nessuno | Creazione delle sezioni dei processi primari, di supporto e organizzativi |
| 1.3.0 | 8/01/2024 | Carlo Rosso | Niccolò Carlesso | Nessuno | Correzione della sottosezione "Aggiornamento delle "Norme di Progetto"" e aggiunte le sottosezioni "Revisione del codice" e "Codifica" |
| 1.2.0 | 31/12/2023 | Carlo Rosso | Niccolò Carlesso | Nessuno | Ristrutturazione del documento per ruolo, piuttosto che per argomento |
| 1.1.0 | 30/10/2023 | Carlo Rosso | Nessuno | Nessuno | Aggiornamento della sezione dedicata alla documentazione e aggiunta una sezione dedicata agli appunti |
| 1.0.0 | 30/10/2023 | Nessuno | Nessuno | Giacomo Gualato | Approvazione finale del documento |
| 0.2.1 | 29/10/2023 | Alessandro Tigani Sava | Niccolò Carlesso | Nessuno | Modifica procedure in sezione Approvazione di un documento |
| 0.2.0 | 24/10/2023 | Matteo Bando | Niccolò Carlesso | Nessuno | Redazione sezioni Versionamento, Verifica di un documento, Approvazione di un documento |
| 0.1.0 | 23/10/2023 | Alessandro Tigani Sava | Nessuno | Nessuno | Redazione sezioni Introduzione, Strumenti, Creazione e modifica di un documento, Ruoli, Registro delle modifiche |

# Indice

# 1 Introduzione

## 1.1 Scopo del Documento

Questo documento, redatto dal team SWEnergy, definisce le norme e le metodologie adottate per lo sviluppo del progetto "Easy Meal". L'obiettivo è fornire una guida chiara e strutturata che faciliti la collaborazione all' interno del team e garantisca la coerenza e la qualità del lavoro svolto. Le norme qui presentate si ispirano agli standard ISO 12207-1995, adattati alle specificità del progetto universitario in questione.

## 1.2 Struttura del Documento

Le sezioni sezione 2, sezione 3 e sezione 4 del documento riflettono i diversi aspetti e fasi del ciclo di vita del software, suddivisi in processi primari, di supporto e organizzativi, rispettivamente, come delineato dagli standard ISO 12207-1995:

- **Processi Primari:** Questa sezione descrive i processi fondamentali nello sviluppo del software, includendo le fasi di acquisizione, fornitura, sviluppo del prodotto software;
- **Processi di Supporto:** In questa parte vengono trattati i processi che supportano lo sviluppo del software, come la gestione della configurazione, la verifica, l'approvazione, la qualità e la risoluzione dei problemi;
- **Processi Organizzativi:** Questa sezione copre i processi trasversali che aiutano a migliorare e mantenere l'efficienza dell' ambiente di sviluppo, inclusi la gestione dei processi, delle infrastrutture, il miglioramento dei processi e la formazione del personale.

Ciascun processo è descritto dalle seguenti sezioni:

1. **Descrizione:** Fornisce una panoramica generale del processo, fornendo informazioni aggiuntive rispetto al titolo del processo;
2. **Scopo:** Specifica gli obiettivi e le finalità del processo;
3. **Attività:** Elenca le attività principali che compongono il processo;

4. **Strumenti:** Specifica gli strumenti utilizzati per l'attuazione del processo.

Dopo le sezioni dedicate ai processi, il documento include una sezione per ciascun ruolo. Oggi sotto-sezione rappresenta un'attività che il ruolo deve svolgere. La struttura delle attività è la seguente:

- **Descrizione**: riguarda l'introduzione all'attività. In aggiunta, sono contenute le informazioni necessarie per lo svolgimento di qualche attività, per rendere la medesima più chiara e comprensibile;
- ***Trigger***: spiega quando il compito deve essere svolto. Quindi sono elencate le condizioni che devono essere verificate per attivare il compito;
- **Scopo**: descrizione dello stato che si vuole raggiungere, in seguito al completamento del compito;
- **Svolgimento**: contiene l'elenco dei *task* che il ruolo è tenuto a svolgere per completare il compito. Le attività possono essere tra loro dipendenti, oppure indipendenti. Si ritiene che il linguaggio usato sia sufficiente per rendere le attività chiare e comprensibili;
- ***Task***: per ogni *task* viene fornita una breve descrizione. Se necessario, viene fornita una serie di passi da seguire per completare il *task*. I passi da seguire sono elencati in ordine e sono dipendenti tra loro.

## 1.3 Riferimenti

### 1.3.1 Normativi

- ISO/IEC 12207:1995.

### 1.3.2 Informativi

- Gestione di progetto.

# 2 Processi Primari

## 2.1 Acquisizione

Il processo di acquisizione coinvolge la definizione dei requisiti di sistema e *software*, la valutazione e selezione dei potenziali fornitori, e la gestione del contratto con il fornitore selezionato.

### 2.1.1 Scopo

Garantire che il *software* acquisito soddisfi i requisiti stabiliti, rispetti i vincoli di budget e di tempo, e sia conforme agli standard di qualità previsti.

### 2.1.2 Attività

1. **Definizione dei requisiti**: Identificazione delle necessità e delle aspettative degli *stakeholder*.
2. **Selezione del fornitore**: Valutazione delle offerte e scelta del fornitore più adatto.
3. **Gestione del contratto**: Definizione degli accordi contrattuali, monitoraggio della conformità e gestione delle modifiche.
4. **Accettazione del *software***: Verifica e approvazione del *software* consegnato rispetto ai requisiti concordati.

### 2.1.3 Strumenti

- **Zoom**: Strumento di videoconferenza utilizzato per le comunicazioni a distanza con il committente;
- **Teams**: Strumento di videoconferenza utilizzato per le comunicazioni a distanza con il proponente;
- ***Presentazioni* di Google**: Strumento di creazione di presentazioni utilizzato per la comunicazione con il cliente;

- ***Advanced Slides* di Obsidian**: Strumento di creazione di presentazioni utilizzato per la comunicazione con il proponente.

## 2.2 Fornitura

Il processo di fornitura comprende tutte le attività necessarie per consegnare il *software* sviluppato al committente, che in questo contesto è rappresentato dal corpo docente o dai revisori del progetto universitario e da un rappresentante di Imola Informatica, ovvero il proponente. La fornitura si concentra sulla preparazione e presentazione del *software* e della documentazione correlata, in conformità con i requisiti del corso e le aspettative degli *stakeholder*.

### 2.2.1 Scopo

L'obiettivo principale è garantire che il *software* e tutti i materiali di supporto siano pronti per la valutazione finale, rispettando i criteri di accettazione definiti.

### 2.2.2 Attività

1. **Preparazione finale:** Completamento di tutte le attività di sviluppo, *testing* e documentazione.
2. **Revisione della documentazione:** Assicurare che tutta la documentazione sia completa, accurata e pronta per la revisione (vedi sottosezione 3.5).
3. **Presentazione:** Organizzare e condurre una presentazione del progetto, dimostrando le funzionalità del *software* e discutendo la documentazione.
4. **Consegna:** Fornire il *software* e tutta la documentazione correlata ai revisori o ai docenti.

### 2.2.3 Strumenti

Gli strumenti utilizzati in questo processo includono sistemi di *versioning* come Git e strumenti per presentazioni come *Presentazioni* di Google o LaTeX.

*Nota: Poiché questo progetto si inserisce in un contesto universitario, non sono previste attività di supporto o assistenza post-vendita una volta consegnato il software.*

## 2.3 Sviluppo

Questo processo comprende tutte le attività necessarie per trasformare i requisiti in un *software* funzionante e conforme alle aspettative degli *stakeholder*.

### 2.3.1 Scopo

Assicurare la creazione di un *software* che risponda pienamente ai bisogni degli utenti, sia tecnicamente valido, mantenibile e scalabile.

### 2.3.2 Attività

1. **Analisi dei requisiti:** Comprensione e documentazione delle necessità degli utenti e degli altri *stakeholder*.
2. **Progettazione del sistema:** Definizione dell'architettura del sistema e dei principali componenti *software* (vedi sottosezione 9.2).
3. **Implementazione:** Codifica effettiva del *software* in base alla progettazione (vedi sottosezione 10.1).
4. **Testing:** Verifica della correttezza del *software* attraverso test funzionali, di integrazione e di sistema.

### 2.3.3 Strumenti

Per il processo di sviluppo sono utilizzati gli IDE *VSCode* oppure *NeoVim*, il sistema di *versioning* Git e l'organizzazione GitHub per la gestione del codice sorgente e altro materiale di progetto. Sono adottate le *GitHub Actions* per l'automazione di test e *deployment*.

# 3 Processi di Supporto

## 3.1 Documentazione

La documentazione è un processo di supporto essenziale che fornisce un insieme di informazioni e dati strutturati necessari per comprendere, utilizzare, e manutenere il *software*.

La documentazione comprende tutti i materiali scritti o elettronici che descrivono le caratteristiche, le operazioni o l'uso del *software*, come i manuali utente, le specifiche tecniche, i rapporti di test, e i piani di progetto.

### 3.1.1 Scopo

Fornire una chiara comprensione del *software*, facilitare la comunicazione tra i membri del team, consentire un uso efficace del *software* da parte degli utenti e supportare le future attività di manutenzione e sviluppo.

### 3.1.2 Attività

1. **Pianificazione della documentazione:** Definire gli obiettivi, il pubblico e la portata della documentazione.
2. **Redazione dei documenti:** Creare i documenti necessari seguendo le linee guida e gli standard stabiliti (vedi sottosezione 8.1).
3. **Revisione e aggiornamento:** Valutare e aggiornare i documenti per garantirne la precisione e la rilevanza nel tempo (vedi sottosezione 11.1.
4. **Gestione della documentazione:** Organizzare, archiviare e rendere facilmente accessibili i documenti a tutti gli *stakeholder* interessati.

### 3.1.3 Strumenti

Gli strumenti utilizzati per la creazione dei documenti sono:

- **LaTeX**: linguaggio di *markup* per la creazione di documenti (www.latex-project.org);
- **VisualStudio Code**: GUI con integrazioni per la creazione di documenti scritti in LaTeX e per la gestione delle repository git (code.visualstudio.com)
  - **LaTeX Workshop**: estensione utilizzata in VisualStudio Code per la compilazione e la scrittura dei documenti.

## 3.2 Gestione della Configurazione

La gestione della configurazione è un processo di supporto che assicura il controllo delle versioni e la tracciabilità dei componenti *software* durante tutto il ciclo di vita del progetto. Questo processo si occupa di mantenere la coerenza delle prestazioni, dei dati funzionali e delle informazioni fisiche di un sistema e dei suoi componenti. Si focalizza sulla gestione di modifiche e configurazioni per prevenire disordine e confusione.

### 3.2.1 Scopo

Questo processo ha lo scopo di assicurare che tutti i componenti del *software* siano identificati, versionati e tracciati nel corso del tempo, facilitando così la gestione delle modifiche e migliorando la qualità del prodotto *software*.

### 3.2.2 Attività

1. **Identificazione della configurazione:** Definire e documentare le caratteristiche funzionali e fisiche dei componenti *software*.
2. **Controllo della configurazione:** Gestire le modifiche attraverso un processo formale di valutazione, approvazione e implementazione.
3. **Registrazione e rapporto dello stato di configurazione:** Tenere traccia di tutte le modifiche apportate ai componenti *software* e documentare lo stato corrente di configurazione.
4. **Verifica:** Assicurare che i componenti *software* siano conformi ai requisiti e che le modifiche siano implementate correttamente.

### 3.2.3 Strumenti

- **Git**: Sistema di controllo versione distribuito utilizzato per il tracciamento delle modifiche al codice sorgente;
- **GitHub**: Piattaforma di hosting per progetti *software* che fornisce strumenti di collaborazione e controllo versione e traccia delle *issue*;

- **GitHub Actions**: Strumento di automazione per l'esecuzione di *workflow* personalizzati.

## 3.3 Accertamento della Qualità

L'accertamento della qualità è un processo di supporto fondamentale che garantisce che il *software* soddisfi i requisiti di qualità stabiliti e le aspettative degli *stakeholder*. Questo processo include la definizione, implementazione, valutazione e manutenzione delle procedure e delle politiche di qualità per assicurare che il *software* prodotto sia di alta qualità.

### 3.3.1 Scopo

Assicurare che il *software* e le pratiche di sviluppo rispettino gli standard di qualità prefissati, migliorando così la soddisfazione del cliente e l'affidabilità del prodotto.

### 3.3.2 Attività

1. **Definizione delle Politiche di Qualità:** Stabilire gli standard e le metriche di qualità in base ai requisiti del progetto e alle aspettative degli *stakeholder*.
2. **Implementazione delle Procedure di Qualità:** Applicare le politiche attraverso metodi concreti e pratiche di sviluppo, come revisioni del codice e test.
3. **Valutazione della Conformità:** Verificare periodicamente che il *software* e i processi di sviluppo rispettino le politiche di qualità stabilite.
4. **Manutenzione e Miglioramento Continuo:** Aggiornare le politiche e le procedure di qualità in base ai *feedback* e ai risultati delle valutazioni per promuovere il miglioramento continuo (vedi sottosezione 7.2).

### 3.3.3 Strumenti

Gli strumenti utilizzati nel processo di accertamento della qualità possono includere *software* autoprodotti di automazione dei test e di raccolta dei dati.

## 3.4 Verifica

Il processo di verifica è essenziale per assicurare che il codice prodotto sia conforme alle aspettative e agli standard definiti. Questo processo coinvolge una serie di attività dettagliate per valutare la qualità e la correttezza del codice.

### 3.4.1 Scopo

Questo processo assicura che il codice sia non solo funzionale ma anche conforme agli standard qualitativi stabiliti, contribuendo significativamente alla qualità generale del prodotto *software* e dei documenti.

### 3.4.2 Attività di Verifica

A seconda che il prodotto da controllare sia un documento o del codice sorgente, sono previste le seguenti attività di verifica:

- **Verifica del Documento:** valutare la correttezza e la completezza del documento rispetto agli standard e alle linee guida stabilite (vedi sottosezione 11.1).
- **Verifica del Codice:** valutare la correttezza e la qualità del codice sorgente rispetto agli standard e alle linee guida stabilite (vedi sottosezione 11.2).

## 3.5 Approvazione

Il processo di approvazione si concentra sulla conferma che i requisiti e il sistema *software* o prodotto finito soddisfino il loro uso inteso specifico. La approvazione può essere condotta in fasi precedenti dello sviluppo.

### 3.5.1 Scopo

Questo processo assicura che il sistema *software* o il prodotto finito siano adeguatamente validati rispetto al loro uso previsto, contribuendo significativamente all'affidabilità e alla soddisfazione dell'utente finale.

### 3.5.2 Attività

L'implementazione del processo di approvazione include le seguenti attività principali:

1. **Identificazione:** valutare se il progetto richieda uno sforzo di approvazione e il grado di indipendenza organizzativa di tale sforzo.
2. **Pianificazione:** stabilire un processo di approvazione per validare il sistema o il prodotto *software* se il progetto lo richiede. Selezionare i compiti di approvazione, inclusi i metodi, le tecniche e gli strumenti associati.
3. **Esecuzione:** eseguire i compiti di approvazione pianificati (vedi sottosezione 6.5).
4. **Rapporti:** inoltrare i rapporti di approvazione al committente e al proponente.

## 3.6 Revisioni Congiunte con il Cliente

Le revisioni congiunte con il cliente sono incontri strutturati tra il team di sviluppo e gli *stakeholder* o i clienti per esaminare il progresso del prodotto *software*, discutere problemi e trovare soluzioni congiunte.

### 3.6.1 Scopo

L'obiettivo di queste revisioni è assicurare che il prodotto *software* in sviluppo rispecchi fedelmente i requisiti e le aspettative del committente e del proponente, e che eventuali discrepanze o incomprensioni siano risolte tempestivamente.

### 3.6.2 Attività

1. **Preparazione della Revisione:** Organizzare l'incontro, definire l'agenda e preparare il materiale da presentare al committente o al proponente (vedi sottosezione 6.2).
2. **Conduzione della Revisione:** Presentare il lavoro svolto, discutere i progressi e raccogliere *feedback* dagli *stakeholder*.
3. **Riscontro ai Feedback:** Analizzare e discutere i *feedback* ricevuti per determinare le azioni correttive necessarie.

4. **Pianificazione delle Azioni Correttive:** Definire un piano per implementare le modifiche richieste o per risolvere problemi identificati durante la revisione (vedi sottosezione 6.3).

5. ***Follow-up*:** Monitorare l'attuazione delle azioni correttive e organizzare revisioni successive se necessario.

### 3.6.3 Partecipanti

Includono membri del team di sviluppo, rappresentanti del cliente o degli *stakeholder*, e possono includere anche esperti di dominio o utenti finali.

### 3.6.4 Documentazione

Tutti gli aspetti salienti della revisione, compresi i *feedback*, le decisioni prese e le azioni correttive pianificate, devono essere documentati all'interno dei verbali esterni e resi disponibili a tutti i partecipanti per riferimento futuro.

## 3.7 Verifiche Ispettive Interne

Le verifiche ispettive interne sono processi attraverso i quali il team di progetto esegue revisioni sistematiche e ispezioni dei propri processi e prodotti *software*, al fine di identificare e correggere gli errori prima che il prodotto sia rilasciato o passi alla fase successiva.

### 3.7.1 Scopo

L'obiettivo delle verifiche ispettive interne è migliorare la qualità dei processi e dei prodotti *software*, riducendo gli errori, aumentando l'efficienza e garantendo la conformità agli standard di progetto.

### 3.7.2 Attività

Le attività tipiche coinvolte nelle verifiche ispettive interne includono:

1. **Pianificazione delle Ispezioni:** Definire gli obiettivi, lo scopo, la portata e il programma delle ispezioni.

2. **Preparazione:** Raccogliere e rivedere i documenti, il codice e altri artefatti da ispezionare.
3. **Conduzione delle Ispezioni:** Eseguire le ispezioni secondo le procedure stabilite, utilizzando *checklist* o linee guida specifiche per identificare gli errori e le aree di miglioramento.
4. **Riunione di Ispezione:** Discutere i risultati delle ispezioni con il team, identificare le cause degli errori e decidere le azioni correttive.
5. **Implementazione delle Azioni Correttive:** Apportare le modifiche necessarie per risolvere gli errori identificati durante le ispezioni.
6. ***Follow-up*:** Verificare che tutte le azioni correttive siano state implementate correttamente e che gli errori siano stati risolti.

### 3.7.3 Partecipanti

Le verifiche ispettive interne coinvolgono diversi ruoli all'interno del team di progetto, tra cui verificatori, analisti, progettisti e programmatori, ciascuno con responsabilità specifiche nel processo di ispezione.

### 3.7.4 Documentazione

Tutti i risultati delle ispezioni, comprese le scoperte, le decisioni prese e le azioni correttive pianificate, devono essere documentati e archiviati per future referenze e valutazioni della qualità.

## 3.8 Risoluzione dei Problemi

La risoluzione dei problemi si occupa della gestione sistematica dei problemi riscontrati nel *software* o nei processi di sviluppo, dalla loro identificazione alla loro risoluzione e documentazione.

### 3.8.1 Scopo

Identificare e risolvere i problemi in modo efficiente per minimizzare l'impatto sul progetto, migliorando la qualità del prodotto e del processo.

### 3.8.2 Attività

1. **Identificazione del Problema:** Riconoscere e documentare i problemi o le discrepanze riscontrate nel *software* o nei processi.
2. **Analisi del Problema:** Valutare il problema per comprenderne le cause radice e determinare l'impatto sul progetto.
3. **Pianificazione delle Azioni Correttive:** Sviluppare un piano di azioni per risolvere il problema, includendo modifiche al *software* o ai processi.
4. **Implementazione delle Azioni Correttive:** Applicare le soluzioni identificate per correggere il problema.
5. **Verifica e Chiusura:** Verificare che la soluzione abbia risolto efficacemente il problema e documentare l'esito e le lezioni apprese.

### 3.8.3 Documentazione

Documentare ogni problema riscontrato, le azioni intraprese per risolverlo e i risultati ottenuti, per mantenere una tracciabilità e fornire un riferimento per problemi futuri.

### 3.8.4 Strumenti

- ***Discussion* di GitHub**: Strumento per la gestione ed il mantenimento di discussioni su problemi e soluzioni.

# 4 Processi Organizzativi

I processi organizzativi sono fondamentali per garantire l'efficienza e l' efficacia dei processi di ciclo di vita del *software* all'interno dell' organizzazione del progetto. Essi forniscono supporto trasversale a tutti i progetti e contribuiscono alla gestione delle risorse, al miglioramento continuo e alla formazione del personale.

## 4.1 Gestione dei Processi

La gestione dei processi comprende le attività di pianificazione, monitoraggio e controllo dei processi di ciclo di vita del *software* all'interno del progetto, assicurando che siano condotti in modo efficace ed efficiente.

### 4.1.1 Scopo

Il principale obiettivo della gestione dei processi è migliorare la qualità del *software* prodotto e l'efficienza dello sviluppo, attraverso la standardizzazione dei processi e l'implementazione delle migliori pratiche.

### 4.1.2 Attività

1. **Pianificazione dei Processi:** Definire gli obiettivi, le procedure e i piani per l'esecuzione e il controllo dei processi di ciclo di vita del *software* (vedi sottosezione 6.3 e sottosezione 6.4).
2. **Monitoraggio e Controllo:** Tenere traccia dei progressi rispetto ai piani stabiliti e intervenire in caso di deviazioni, per assicurare l'allineamento con gli obiettivi di progetto.
3. **Valutazione dei Processi:** Analizzare periodicamente l'efficacia e l'efficienza dei processi attuati, identificando aree di miglioramento.
4. **Miglioramento dei Processi:** Implementare azioni correttive e miglioramenti basati sui risultati delle valutazioni, per ottimizzare i processi di ciclo di vita del *software*.
5. **Formazione e Sviluppo del Team:** Assicurare che tutti i membri del team abbiano le competenze e le conoscenze necessarie per attuare efficacemente i processi definiti.

### 4.1.3 Strumenti

Per tracciare le attività sono utilizzati i *project* di GitHub, inoltre viene anche adottato Git come sistema di controllo versione. Infine, sono utilizzati *Discord* e *Telegram* per la comunicazione interna.

## 4.2 Gestione delle Infrastrutture

La gestione delle infrastrutture si occupa dell'organizzazione e della manutenzione delle infrastrutture tecniche necessarie per supportare lo svolgimento efficace dei processi di ciclo di vita del *software*.

### 4.2.1 Scopo

Assicurare che l'ambiente tecnologico sia adeguatamente configurato, gestito e manutenuto per supportare le attività di sviluppo, *testing*, *deployment* e operatività del *software*.

### 4.2.2 Attività

1. **Valutazione delle Necessità:** Identificare i requisiti infrastrutturali basati sulle esigenze del progetto, incluse le piattaforme di sviluppo, gli ambienti di *testing* e i sistemi di produzione.
2. **Configurazione e Implementazione:** Configurare e implementare le infrastrutture tecniche necessarie, inclusi *hardware*, reti, sistemi operativi e servizi.
3. **Manutenzione e Aggiornamento:** Eseguire la manutenzione regolare delle infrastrutture per assicurare prestazioni ottimali e applicare aggiornamenti di sicurezza e funzionalità.
4. **Monitoraggio e *Troubleshooting*:** Monitorare le infrastrutture per identificare e risolvere tempestivamente eventuali problemi o malfunzionamenti.
5. **Gestione della Sicurezza:** Implementare misure di sicurezza appropriate per proteggere le infrastrutture e i dati da accessi non autorizzati e da altre minacce.

### 4.2.3 Strumenti

Sono utilizzati programmi autoprodotti e le *GitHub Actions* per l'automazione di tutte le attività che lo permettono.

#### 4.2.4 Documentazione

Mantenere una documentazione dettagliata sulle configurazioni delle infrastrutture, sulle procedure operative standard per garantire trasparenza e facilitare la gestione.

## 4.3 Miglioramento del Processo

Il miglioramento del processo si basa sul Ciclo di Miglioramento Continuo PDCA.

#### 4.3.1 Scopo

Lo scopo del processo consiste nell'ottimizzare i processi organizzativi e incrementare l'efficacia e l'efficienza nel ciclo di vita del software.

#### 4.3.2 Attività

1. ***Plan***: Definire gli obiettivi specifici di miglioramento, identificare le attività necessarie per raggiungerli, stabilire le scadenze e assegnare le responsabilità. Questo include la selezione di metriche di processo per misurare l'efficacia delle azioni di miglioramento (vedi sottosezione 7.1).

2. ***Do***: Implementare le attività pianificate, seguendo i piani stabiliti. Questo può includere la formazione del personale, l'aggiornamento delle procedure o l'introduzione di nuovi strumenti e tecnologie.

3. ***Check***: Monitorare e valutare l'esito delle azioni di miglioramento rispetto agli obiettivi prefissati, utilizzando le metriche di processo definite nella fase di pianificazione. Analizzare i dati raccolti per identificare le tendenze, le deviazioni e le aree che necessitano di ulteriori miglioramenti.

4. ***Act***: Sulla base dei risultati ottenuti nella fase di valutazione, intraprendere azioni correttive per consolidare i miglioramenti ottenuti e indirizzare le aree che non hanno raggiunto gli obiettivi prefissati. Questa fase può anche includere la standardizzazione di nuove pratiche di successo e la modifica dei piani di miglioramento per i cicli futuri.

5. **Ciclicità del Processo**: Ripetere il ciclo PDCA per garantire un miglioramento continuo dei processi, adattando gli obiettivi e le strategie in base ai risultati ottenuti e alle nuove priorità identificate.

## 4.4 Formazione del Personale

La formazione del personale è un processo organizzativo critico che mira a sviluppare le competenze e le conoscenze dei membri del team, garantendo che siano adeguatamente equipaggiati per contribuire efficacemente al progetto.

### 4.4.1 Scopo

Incrementare le competenze tecniche e metodologiche del team, promuovere l'innovazione e migliorare la qualità del lavoro svolto, attraverso un approccio di apprendimento continuo e adattivo.

### 4.4.2 Attività

1. **Analisi dei Bisogni Formativi:** Valutare le esigenze di formazione del team, identificando le lacune nelle competenze e nelle conoscenze.

2. **Pianificazione della Formazione:** Sviluppare un piano di formazione che includa obiettivi di apprendimento, metodi formativi, risorse necessarie e calendario delle attività formative.

3. **Erogazione della Formazione:** Implementare le attività formative attraverso *workshop*, seminari, corsi *online*, *mentoring* e auto-studio, adattando l'approccio in base alle preferenze e ai bisogni del team (vedi sottosezione 9.1).

4. **Valutazione dell'Impatto:** Misurare l'efficacia della formazione attraverso *feedback*, valutazioni e analisi delle prestazioni, per garantire che gli obiettivi di apprendimento siano stati raggiunti.

5. **Miglioramento Continuo:** Utilizzare i *feedback* e i risultati delle valutazioni per perfezionare continuamente le iniziative formative, assicurando che restino rilevanti e utili.

### 4.4.3 Risorse

L'accesso a risorse formative come piattaforme di *e-learning*, libri, articoli, e la partecipazione a conferenze e *workshop* esterni o interni sono incoraggiati e supportati dall'organizzazione.

### 4.4.4 Cultura dell'Apprendimento

Promuovere una cultura dell'apprendimento all'interno del team, incoraggiando la condivisione delle conoscenze, la curiosità e l'iniziativa personale nell'esplorazione di nuove competenze e tecnologie.

# 5 Tutti

## 5.1 Lavoro sul progetto

### 5.1.1 Descrizione

Questo compito descrive i passi da seguire per svolgere qualunque altro compito assegnato dal responsabile di progetto.

### 5.1.2 *Trigger*

- Il responsabile di progetto assegna un compito ad un membro del gruppo.

### 5.1.3 Scopo

- Svolgere il compito assegnato;
- Risulta conclusa un'*issue* nella *repository* corrispondente;
- Il compito è stato verificato e convalidato da una persona diversa da chi lo ha svolto.

### 5.1.4 Svolgimento

Di seguito sono descritte le attività da svolegere per effettuare un compito assegnato:

- **Analisi**: si crea una *issue* nella *repository* nella quale verrà svolto il compito. La *issue* sarà assegnata a se stessi. All'interno della issue si descrive l'insieme degli obiettivi da raggiungere affinché l'attività sia completata. L'attività deve essere collegata al *project* corrispondente alla fase di sviluppo in cui si trova il progetto. In aggiunta, deve essere marcata con un *tag* che ne identifichi la tipologia;
- **Appunti**: si inseriscono i file degli appunti nel proprio *branch* personale all'interno della *repository* `appunti-swe`. Nella *repository*, deve essere presente un `README.md` contenente l'organizzazione della cartella per permettere agli altri membri di orientarsi;
- **Svolgimento**: si svolge l'attività assegnata, al meglio delle proprie capacità e cercando di rispettare le scadenze;

- **Verifica**: si chiede ad un membro del gruppo, tendenzialmente al verificatore, di controllare la conformità del lavoro svolto.

# 6 Responsabile

## 6.1 Organizzare un *meeting* interno

Il responsabile organizza i *meeting* interni, ovvero le *stand-up*. Le *stand-up* sono riunioni brevi, della durata di circa 30 minuti, che si svolgono su *Discord*. In esse sono trattati i seguenti argomenti:

- ***Brainstorming*:** è riassunto brevemente il lavoro svolto nella settimana;
- **Problemi riscontrati:** sono esposti i problemi riscontrati durante la settimana;
- ***To-do list*:** sono discussi i compiti da svolgere nella settimana successiva;
- ***Dubbi*:** sono evidenziati i dubbi riguardo alle attività da svolgere;
- **Restrospettiva:** i membri del gruppo espongono i problemi le cose positive e negative non inerenti alle attività, riscontrati durante la settimana e le possibili soluzioni. I problemi possono, per esempio, riguardare l'organizzazione del lavoro o la comunicazione tra i membri del gruppo o con il proponente.

### 6.1.1 *Trigger*

- Ogni venerdì, per dare tempo al responsabile di preparare il materiale per la *stand-up*.

### 6.1.2 Scopo

- Rendere la comunicazione tra i membri del gruppo più efficace ed efficiente;
- Creare della documentazione usufruibile in caso di dubbi o problematiche;
- Formalizzare le decisioni prese durante la riunione.

### 6.1.3 Svolgimento

- **Pianificazione:** il responsabile deve decidere quando svolgere la *stand-up*. Di seguito i passi:

  1. **Anticipare la data:** nella *stand-up* precedente il responsabile si informa sulle disponibilità dei membri del gruppo rispello alla prossima *stand-up*;
  2. **Pianificare la data:** il responsabile propone delle date e degli orari per la prossima *stand-up* e le propone sul gruppo *Telegram* del gruppo. I membri del gruppo esprimono la loro preferenza attraverlo un sondaggio.

- **Ordine del giorno:** il responsabile stila un ordine del giorno, ovvero una lista degli argomenti da trattare durante la riunione. Di seguito i passi:

  1. **Template:** il responsabile utilizza il template delle *stand-up* situato nella *repository* `appunti-swe/stand-up/template-stand-up.md`;
  2. **Brainstorming:** il responsabile si informa con i membri del gruppo attraverso *Telegram* in merito ai punti che bisogna trattare durante la riunione;
  3. ***To-do list*:** il responsabile stila la lista delle attività da svolgere nella settimana successiva. La lista viene poi discussa e approvata durante la riunione.

- **Verbale della riunione:** il responsabile redige il verbale della riunione, in cui vengono riportati gli argomenti trattati e le decisioni prese. Di seguito i passi per redigere il verbale interno:

  1. **Appunti:** l'ordine del giorno (il punto precedente) viene utilizzato come base per stilare il verbale interno;
  2. **Template:** viene copiata la cartella di template dei verbali interni e viene rinominata seguendo il formato: `YYYY-MM-DD_I`;
  3. **Stesura:** poichè si tratta di un documento, si rimanda alla sottosezione che illustra come redigere un documento (vedi sottosezione 8.1).

### 6.1.4 Strumenti

- ***Discord*:** per svolgere la riunione;

- ***Telegram*:** per comunicare con i membri del gruppo;
- ***GitHub*:** per la gestione del codice sorgente e altro materiale di progetto.

## 6.2 Organizzare un *meeting* esterno

Il responsabile organizza i *meeting* esterni, ovvero i SAL tenuti tra il proponente e SWEnergy. I *SAL* sono riunioni brevi, della durata di circa 30 minuti, che hanno luogo su *Teams*. In esse sono trattati i seguenti argomenti:

- **Riassunto:** il responsabile riassume le attività svolte dal gruppo durante lo sprint;
- **Problemi riscontrati:** il responsabile espone i problemi riscontrati durante lo sprint;
- ***To-do list*:** sono discussi i compiti da svolgere nella settimana successiva tra il gruppo e il proponente;
- ***Dubbi*:** il responsabile esponge i dubbi riguardo alle attività da svolgere;
- **Restrospettiva:** il responsabile guida la discussione sulla qualità del prodotto e soprattutto del processo.

### 6.2.1 *Trigger*

- La domanica precedente al SAL.

### 6.2.2 Scopo

- Rendere la comunicazione tra i membri del gruppo più efficace ed efficiente;
- Creare della documentazione usufruibile in caso di dubbi o problematiche;
- Formalizzare le decisioni prese durante la riunione.

### 6.2.3 Svolgimento

- **Pianificazione:** il responsabile deve decidere quando svolgere un SAL. Di seguito i passi:

1. **Anticipare la data:** nel *SAL* precedente il responsabile e il proponente concordano la data del prossimo *SAL*;
2. **Pianificare l'ora:** il responsabile contatta su *Telegram* il proponente, gli condivide l'ordine del giorno e concorda l'ora del *SAL*. Le due attività sono svolte in concomitanza, perché si può prevedere la durata del *SAL* solo dopo aver stilato l'ordine del giorno.

- **Ordine del giorno:** il responsabile stila l'ordine del giorno, ovvero una lista degli argomenti da trattare durante la riunione. Di seguito i passi:

  1. **Template:** il responsabile utilizza il template dei SAL precedenti, situato nella *repository* `appunti-swe`;
  2. **Brainstorming:** il responsabile si informa con i membri del gruppo attraverso le *stand-up* in merito allo *status quo* del progetto;
  3. ***To-do list*:** il responsabile stila la lista delle attività da svolgere nello sprint successivo. La lista viene poi discussa e approvata durante la riunione.

- **Verbale della riunione:** il responsabile deve redigere il verbale della riunione, in cui vengono riportati gli argomenti trattati e le decisioni prese. Di seguito i passi per redigere il verbale interno:

  1. **Appunti:** l'ordine del giorno (il punto precedente) viene utilizzato come base per stilare il verbale esterno;
  2. **Template:** viene copiata la cartella di template dei verbali esterni e viene rinominata seguendo il formato: `YYYY-MM-DD_E`;
  3. **Stesura:** poichè si tratta di un documento, si rimanda alla sottosezione che illustra come redigere un documento (vedi sottosezione 8.1).

#### 6.2.4 Strumenti

- **Telegram:** per comunicare con il proponente e con i membri del gruppo;
- **Teams:** per svolgere il *SAL*;
- **GitHub:** per condividere l'ordine del giorno e il verbale esterno;

- **Mail:** per la condivisione di documenti e per la comunicazione con il proponente.

## 6.3 Pianificazione delle attività

Il responsabile pianifica le attività da svolgere durante lo sprint e le suddivider tra i membri del gruppo. Inoltre, aggiorna il piano di progetto in base alle attività svolte e a quelle da svolgere. La pianificazione avviene tramite l'uso dei diagrammi di Gantt disponibili su *GitHub*.

### 6.3.1 *Trigger*

- Comincia una nuova iterazione, che sia uno sprint$^G$ od un mini-sptrint$^G$ .

### 6.3.2 Scopo

- Aggiornare il piano di progetto in base alle attività svolte e a quelle da svolgere.
- Pianificare le attività da svolgere durante l'iterazione corrente;
- Guidare lo svolgimento delle attività;
- Produrre la documentazione che permette di tenere traccia delle attività svolte e da svolgere.

### 6.3.3 Svolgimento

- **Creazione delle *issue***: il responsabile crea delle issue che descrivono le attività da svolgere e guidano i lavoratori nella loro esecuzione. Di seguito sono riportati i passi per definire le *issue*:
  1. **Identificazione**: il responsabile identifica le attività da svolgere e le aggiunge su *GitHub*;
  2. **Priorità**: il responsabile assegna una priorità alle issue in base all'urgenza e all'importanza;
  3. **Scadenza**: il responsabile assegna una data di scadenza alle issue in base alla priorità e alla durata dell'attività. L'attività viene quindi inserita nel *project*

di *Github* corrispondente alla milestone di riferimento. In questo modo viene aggiornato il diagramma di Gantt;

4. **Perfezionamento**: il responsabile guida la discussione in merito alle *issue* durante le riunioni. In questo modo sono aggiornate priorità, scadenza e descrizione;
5. **Assegnazione**: il responsabile assegna le issue ai membri del gruppo in base alle loro competenze e disponibilità.

## 6.4 Aggiornamento del "Piano di progetto"

### 6.4.1 *Trigger*

- Inizio di uno sprint;
- Fine di uno sprint;

### 6.4.2 Scopo

- Formalizzare la pianificazione delle attività da svolgere durante lo sprint;
- Disambiguare la pianificazione;
- Aggiornare le informazioni relative ai rischi e al modello di sviluppo;
- Aggiornare le informazioni utili alla verifica dello stato di avanzamento del progetto;

### 6.4.3 Svolgimento

- **Rischi e modello di sviluppo**: il responsabile aggiorna le informazioni in esse contenute in base all'esperienza maturata durante il periodo da responsabile;
- **Pianificazione**: il responsabile aggiorna la sezione di pianificazione rispettando la struttura già definita nel documento. Eventualmente può proporre modifiche alla struttura di pianificazione di perido. Queste sono discusse nelle riunioni interne. Di seguito sono riportati i passi da seguire per aggiornare la sezione di pianificazione:
    1. **Creazione**: nella cartella `sprint` viene aggiunto un nuovo file `<numero_dello_sprint>.tex`;

2. **Diagramma di Gantt**: il responsabile copia il diagramma di Gantt sviluppato nel *project* di GitHub;
3. **Spiegazione del diagramma**: per ciascuna attività riportata nel diagramma di Gantt, il responsabile riporta chi se ne occupa e la durata prevista;
4. **Preventivo**: il responsabile riporta in forma tabellare le ore preventivate per ciascuna persona divisa per ruolo e calcola le ore ed il costo totali per il periodo;

- **Consuntivo**:

1. **Riassunto delle attività svolte**: il responsabile legge i *commit* e le *issue* chiuse durante lo sprint e ne riporta un riassunto nel documento;
2. **Consuntivo**: il responsabile riporta in forma tabellare le ore effettivamente impiegate per ciascuna persona divise per ruolo e calcola le ore ed il costo totali per il periodo. Le ore effettive si trovano nel file excel condiviso su *Google Drive*.
3. **Gestione dei ruoli**: il responsabile riporta in un diagramma a torta la distribuzione delle ore per ruolo effettivamente impiegate durante lo sprint.

- **Modifica di un documento**: dal momento che l'aggiornamento del documento "Piano di progetto" rientra nella casistica di modifica di un documento, si rimanda alla sezione che illustra come redigere un documento (vedi 8.1).

#### 6.4.4 Strumenti

- ***GitHub*:** per la gestione del codice sorgente e altro materiale di progetto;
- ***Google Drive*:** per la gestione dei file excel.
- **Preventivi:** si tratta di un programma autoprodotto che permette di calcolare in modo automatico le ore e i costi totali per il periodo.

### 6.5 Approvare un documento

#### 6.5.1 *Trigger*

- Un documento viene completato rispetto alla fase attuale del progetto.

#### 6.5.2 Scopo

- Assicurarsi che il documento soddisfi i requisiti ad esso associati;
- Convalidare il contenuto ed il completameto del documento.

#### 6.5.3 Svolgimento

- **Approvazione:**
  1. **Seconda verifica**: il responsabile verifica il documento (vedi sottosezione 11.1);
  2. **Aggiornamento della versione**: dopo che il documento viene corretto dall'autore, il responsabile aggiorna la sua versione ed il suo stato;
  3. **Versione**: sia $X.Y.Z$ la versione del documento, dopo l'approvazione, il valore di $X$ viene incrementato di $1$, mentre $Y$ e $Z$ vengono azzerati.

# 7 Amministratore

## 7.1 Aggiornamento delle "Norme di progetto"

#### 7.1.1 *Trigger*

- Si discute di qualche processo da aggiungere o modificare durante un *meeting*.

#### 7.1.2 Scopo

- Mantenere il documento coerente rispetto al modello di sviluppo e ai processi adottati da SWEnergy;
- Formalizzare i processi adottati da SWEnergy, per chiarire eventuali dubbi e per facilitare l'individuazione di attività e processi da svolgere;
- Mostrare l'evoluzione dell'organizzazione del lavoro di SWEnergy;
- Evidenziare i dubbi e le lacune intestini ai processi di sviluppo.

### 7.1.3 Svolgimento

- **Identificazione delle attività**: in quale modo l'amministratore ed il gruppo possono individuare le attività da includere nel documento. Di seguito sono riportati i passi da seguire:

  1. **Nuovo compito**: durante gli incontri, SWEnergy si rende conto che alcune attività si presentano di frequente;
  2. **Ipotesi**: SWEnergy ipotizza il flusso di lavoro da svolgere per completare il compito. Sono stesi degli appunti che verranno poi inseriti nel documento "Norme di progetto";
  3. **Sperimentazione**: i componenti del gruppo che svolgono l'attività, sperimentano diverse tecniche per completare il compito, partendo dall'ipotesi iniziale;
  4. **Perfezionamento**: i componenti che hanno svolto l'attività, spiegano al gruppo il processo seguito. SWEnergy lo discute e lo valuta;
  5. **Formalizzazione**: l'amministratore inserisce il compito nel documento "Norme di progetto". Nota bene: viene modificato un documento, quindi si rimanda alla sottosezione che illustra come redigere un documento (vedi sottosezione 8.1).

- **Aggiornamento delle attività**: in seguito ad una discussione organica a SWEnergy, l'amministratore modifica il compito nel documento "Norme di progetto". Nota bene: viene modificato un documento, quindi si rimanda alla sottosezione che illustra come redigere un documento (vedi sottosezione 8.1).

## 7.2 Aggiornamento del "Piano di qualifica"

### 7.2.1 *Trigger*

- Termina uno sprint;

### 7.2.2 Scopo

- Mantenere sotto controllo la qualità del prodotto;
- Mantenere sotto controllo l'andamento del progetto;

- Documentare quanto qui sopra, per poterlo mostrare al committente durante le revisioni e per evidenziarne l'evoluzione nel tempo.

### 7.2.3 Svolgimento

- **Identificazione di una metrica**: in quale modo l'amministratore ed il gruppo possono individuare le metriche utili a controllare e valutare la qualità del prodotto. Di seguito sono descritti i passi da seguire:

  1. **Nuova metrica**: durante gli incontri, uno dei componenti di SWEnergy propone una nuova metrica da adottare per valutare la qualità del prodotto;
  2. **Discussione**: i componenti del gruppo discutono in merito alla metrica proposta: se è utile, se è applicabile ed in quale modo verificare i risultati ottenuti e formalizzarli;
  3. **Formalizzazione**: l'amministratore inserisce la metrica di qualità nel documento "Piano di qualifica". Nota bene: viene modificato un documento, quindi si rimanda alla sottosezione che illustra come redigere un documento (vedi sottosezione 8.1).

- **Aggiornamento di una metrica**: In seguito ad una discussione organica a SWEnergy, l'amministratore modifica qualche caratteristica di una metrica nel documento "Piano di qualifica". Nota bene: viene modificato un documento, quindi si rimanda alla sottosezione che illustra come redigere un documento (vedi sottosezione 8.1).

- **Misurazione**: l'amministratore misura i risultati ottenuti applicando le metriche di qualità. Di seguito sono descritti i passi di aggiornamento del documento "Piano di qualifica":

  1. **Nuovi risultati**: alla fine di ogni sprint, l'amministratore e il gruppo valutano i risultati di qualità ottenuti applicando le metriche concordate;
  2. **Discussione**: I risultati sono discussi durante la retrospettiva e, se ritenuto opportuno, sono modificati gli obiettivi di qualità adottati da SWEnergy;
  3. **Inserimento dei risultati**: l'amministratore inserisce i risultati ottenuti nel documento "Piano di qualifica". Nota bene: viene modificato un documento, quin-

di si rimanda alla sottosezione che illustra come redigere un documento (vedi sottosezione 8.1).

# 8 Analista

## 8.1 Redazione di un documento

L'analista redige i documenti, in particolare l'analista redige l'"Analisi dei requisiti".

### 8.1.1 *Trigger*

- Sono presenti dei dubbi o delle lacune in merito a qualcosa;
- Risulta necessario formalizzare qualche concetto o qualche argomento.

### 8.1.2 Scopo

- Risolvere i dubbi e le lacune riguardo a un argomento, o almeno formalizzare i dubbi e le lacune;
- Formalizzare la definizione di un concetto o di un argomento, per renderlo chiaro ed inequivoco.

### 8.1.3 Struttura del documento

A ciascun documento corrisponde un'omonima cartella che viene creta all'interno della cartella che rappresenta la fase in cui si trova il progetto, quando viene prodotto il documento. La cartella della fase si trova nella *repository* `doc-latex` dell'organizzazione GitHub del gruppo. Il nome della cartella è il nome del documento che deve rispettare le seguenti regole:

- deve avere la prima lettera maiuscola;
- sono previsti gli spazi tra le parole e le parole successive alla prima sono in minuscolo.

Di seguito la struttura della cartella:

```
/ (Nome del documento)
└── main.tex
```

```
└── sec
    ├── registro_modifiche.tex
    ├── introduzione.tex
    └── le_altre_sezioni.tex
```

#### 8.1.4 `main.tex`

Di seguito la struttura del file `main.tex`:

- **Import dei template**: sono importati i template per la creazione del documento. I template sono: `copertina.tex`, `header_footer.tex` e `variable.tex`. In aggiunta, sono importati i modelli specifici per il documento che si sta redigendo;
- **Inizializzazione delle variabili:** sono inizializzate le variabili che verranno utilizzate nel documento;
- **Struttura del documento:** attraverso l'uso degli `input` viene gestita la struttura del documento.

#### 8.1.5 Svolgimento

- **Modifica di un documento**: l'analista aggiorna il documento in base alle modifiche richieste dal verificatore e in base alle informazioni necessarie per la redazione del documento. Di seguito sono elencati i passi per completare l'attività:
  1. ***Pull***: l'analista effettua il *pull* della *repository* `doc-latex` per avere l'ultima versione della *repository*;
  2. ***Checkout:*** l'analista effettua il *checkout* del *branch* verso il *branch* chiamato come il documento che si sta redigendo;
  3. **Struttura:** l'analista modifica il `main.tex` in base alle modifiche necessarie;
  4. **Gestione dei *file*:** l'analista crea, elimina o rinomina i *file* nella cartella `sec` in modo tale che siano rispecchiate le modifiche apportate al `main.tex`;
  5. **Contenuto:** l'analista modifica i *file* nella cartella `sec` in base alle modifiche necessarie;
  6. ***Push*:** l'analista effettua un *commit* e il *push*;

7. ***Pull request*:** l'analista può creare una *pull request* verso il `main`, per chiedere al verificatore, la verifica del documento;
8. **Verifica:** l'analista informa il verificatore che il documento è pronto per la verifica;
9. **Correzione:** l'analista corregge il documento in base alle segnalazioni del verificatore;
10. **Chiusura:** l'analista effettua un *push* del branch inserendo nel messaggio di *commit* la parola `close` seguita dal numero della issue che si sta risolvendo;
11. **Secondo merge:** l'analista può concludere la *pull request* con il *main*.

### 8.1.6 Strumenti

Gli strumenti utilizzati per la creazione dei documenti sono:

- **LaTeX**: linguaggio di *markup* per la creazione di documenti (www.latex-project.org);
- **VisualStudio Code**: GUI con integrazioni per la creazione di documenti scritti in LaTeX e per la gestione delle repository git (code.visualstudio.com)
  - **LaTeX Workshop**: estensione utilizzata in VisualStudio Code per la compilazione e la scrittura dei documenti.

# 9 Progettista

## 9.1 Organizzare un *workshop*

I progettisti sono tenuti a sperimentare nuove tecnologie, per produrre le PoC. SWEnergy non norma il processo di sperimentazione e produzione delle PoC, d'altra parte, ritiene che sia importante spiegare i risultati ottenuti dalle PoC al resto del *team*. I *workshop* sono un'insieme di appunti, presentazioni e codice per illustrare i risultati ottenuti dalle PoC.

### 9.1.1 *Trigger*

- Qualche membro del gruppo non conosce qualche tecnologia da implemetare.

### 9.1.2 Scopo

- Condividere le conoscenze tecniche tra i membri del gruppo;
- Documentare le conoscenze tecniche acquisite;
- Imparare ad usare la nuova tecnologia;
- Provvedere affinché tutti i membri di SWEnergy abbiano una conoscenza di base e sufficiente per adottare la tecnologia all'interno del progetto.

### 9.1.3 Svolgimento

- **Bozza di appunti:** i progettisti sono tenuti a produrre dei *markdown* per spiegare e riassumere i contenuti delle PoC. I *file* così prodotti sono organizzati come il progettista meglio crede, all'interno del *repository* `appunti-swe`.
- **Appunti web:** a partire dagli appunti sopra prodotti, sono organizzati i *workshop*. Di seguito sono elencati i passi da seguire per pubblicare gli appunti di un *workshop*:
  1. Creare una cartella all'interno del *repository* `Project-SWEnergy.github.io` con il nome del *workshop* da organizzare;
  2. Creare un `readme.md` all'interno della cartella appena creata, che collega gli appunti all'interno della cartella tra loro;
  3. Effettuare il *push* delle modifiche sul *repository* remoto.
- **Presentazione:** i progettisti sono tenuti a produrre una presentazione per presentare gli appunti prodotti e le PoC realizzate. Si noti che la presentazione non ha una descrizione prescrittiva perché, a seconda del contenuto e delle conoscenze tecnologiche del progettista, può essere realizzata con diversi strumenti. Viene consigliato l'uso di *Obsidian* e del *plugin Advanced Slides*.

## 9.2 Progettazione

I progettisti si occupano della progettazione del *software*, ovvero devono definire l'architettura del sistema, i moduli e le interfacce tra di essi. Inoltre, devono definire i test di unità e di integrazione, in modo da verificare che il sistema funzioni correttamente.

### 9.2.1 *Trigger*

- Dopo l'RTB avviene la fase di progettazione più vasta, ma non in dettaglio;
- Ogni volta che viene implementata una nuova funzionalità, viene progettata la struttura del *software* che la implementa in modo dettagliato.

### 9.2.2 Scopo

- Definire l'architettura del sistema;
- Definire i moduli e le interfacce tra di essi;
- Definire i test di unità e di integrazione.

### 9.2.3 Svolgimento

- **Progettazione ad alto livello:** il progettista definisce l'architettura del sistema, i moduli e le interfacce tra di essi. Di seguito i passi che vengono seguiti:
  1. **Ripasso dei requisiti:** il progettista studia i requisiti e le specifiche del sistema;
  2. **Studio delle PoC:** il progettista studia le PoC per individuare i problemi e le soluzioni adottate;
  3. **Descrizione:** partendo dalle PoC, il progettista crea degli appunti che evidenzino la struttura da realizzare;
  4. **Definizione dell'architettura:** a partire dalla descrizione del sistema, il progettista crea i diagrammi delle classi, per guidare lo sviluppo del sistema;
  5. **Appunti integrativi:** il progettista crea degli appunti per motivare le scelte fatte e per supplire alle mancanze dei diagrammi delle classi;

6. **Test di integrazione:** il progettista definisce i test di integrazione, in modo da verificare che il sistema funzioni correttamente.

- **Progettazione di dettaglio:** il progettista definisce i dettagli di implementazione di una nuova funzionalità. Di seguito i passi da seguire:

1. **Scelta della funzionalità:** il progettista sceglie la funzionalità da implementare;
2. **Studio dell'architettura:** il progettista studia l'architettura del sistema, per capire come la nuova funzionalità si inserisce nel sistema;
3. **Definizione delle interfacce:** il progettista definisce le interfacce tra i moduli;
4. **Descrizione:** il progettista crea degli appunti integrativi, per guidare lo sviluppo del programmatore e per motivare le scelte fatte;
5. **Definizione dei test di unità:** il progettista definisce i test di unità, in modo da verificare che la nuova funzionalità sia implementata correttamente.

#### 9.2.4 Strumenti

- **StarUML:** per la creazione dei diagrammi delle classi;
- **GitHub:** per la condivisione dei diagrammi delle classi e degli appunti;

# 10 Programmatore

## 10.1 Codifica

Il programmatore scrive il codice sorgente che compone l'applicativo. Il codice sorgente è scritto in linguaggio TypeScript.

#### 10.1.1 *Trigger*

- Viene completata la progettazione di una *feature*;

#### 10.1.2 Scopo

- Implementare le funzionalità richieste dal proponente;
- Soddisfare qualche requisito;

### 10.1.3 Svolgimento

- **Progettazione:** il programmatore deve produrre dei commenti o degli appunti che descrivano la struttura del codice che andrà a scrivere nella prossima attività. Questi commenti devono poi essere riorganizzati e riportati nella *issue* corrispondente;
- **Test:** il programmatore deve scrivere dei test per verificare il corretto funzionamento del codice che andrà a scrivere;
- **Codifica di una funzione o metodo:** di seguito sono elencati i passi che il programmatore deve seguire per la codifica del prodotto software:
  1. **Pull:** il programmatore esegue un *pull* del codice sorgente dal *repository* remoto;
  2. **Branch:** il programmatore crea un nuovo branch di lavoro a partire dal branch `dev`;
  3. **Commenti:** il programmatore scrive lo scopo della funzione o del metodo che andrà a codificare e ne descrive la firma;
  4. **Codifica:** il programmatore scrive il codice che compone il corpo della funzione o del metodo;
  5. **Test:** il programmatore esegue i test di verifica. In caso di fallimento, il programmatore deve correggere il codice e ripetere la verifica;
  6. **Iterazione:** se il programmatore vuole scrivere altre funzioni torna al punto 3, altrimenti prosegue con il punto successivo;
  7. **Push:** il programmatore esegue un *push* del codice sorgente sul *repository* remoto.
  8. **Verifica:** il programmatore segnala al verificatore che il codice è pronto per essere verificato.
  9. **Correzione:** se il verificatore segnala degli errori, il programmatore deve correggere il codice e torna al passo precendente. Altrimenti, il programmatore può procedere al passo successivo.
  10. **Chiusura:** il programmatore effettua il *merge* del *branch* di lavoro con il *branch* `dev` e chiude il *ticket* di *GitHub* corrispondente.

# 11 Verificatore

## 11.1 Verificare del documento

Il verificatore deve verificare che i documenti prodotti mentre svolge il suo ruolo siano conformi alle norme stabilite in questa sotto-sezione.

### 11.1.1 *Trigger*

- Viene prodotto un incremento su di un documento;
- Un componente di SWEnergy segnala la necessità di una verifica.

### 11.1.2 Scopo

- Evidenziare gli errori in un documento e segnalarli all'autore del documento;
- Assicurarsi che il documento soddisfi le norme qui sotto descritte;
- Convalidare l'incremento di un documento per garantirne l'integrità agli altri componenti di SWEnergy.

### 11.1.3 Norme

- **Correttezza grammaticale:** il testo deve essere privo di errori grammaticali;
- **Correttezza lessicale:** il testo deve essere privo di errori lessicali;
- **Correttezza ortografica:** il testo deve essere privo di errori ortografici;
- **Correttezza sintattica:** il testo deve essere sintatticamente corretto;
- **Correttezza di contenuto:** il testo deve essere privo di errori di contenuto;
- **Correttezza della struttura:** in ogni documento che contiene il registro delle modifiche, deve essere anche presente un'introduzione che spiega la struttura del documento medesimo, coerente con la struttura del documento;
- **Completezza**: il documento deve essere completo di tutte le sezioni opportune;

- **Coerenza**: il contenuto del documento deve essere coerente con il suo scopo, con le norme qui descritte e con il contenuto di eventuali documenti correlati;
- **Chiarezza espositiva**: il documento deve essere scritto in modo chiaro e comprensibile;

### 11.1.4 Svolgimento

Per verificare la correttezza di un documento, il verificatore deve completare le seguenti attività:

- **Correzione dei refusi:** il verificatore deve correggere i refusi presenti nel documento. Sono considerati refusi gli errori della tipologia grammaticale, lessicale, ortografica e sintattica;
- **Verifica del contenuto:** il verificatore deve verificare che il contenuto del documento sia corretto e coerente con il suo scopo. Di seguito sono riportati i passi da seguire:
  1. **Lettura del documento:** il verificatore deve leggere il documento per comprendere il contenuto del documento;
  2. **Appunti degli errori**: durante la lettura il verificatore prende nota di eventuali errori;
  3. **Ricerca delle soluzioni**: il verificatore deve trovare una soluzione per ogni errore trovato;
  4. **Spiegazione degli errori**: il verificatore deve segnalare all'autore del documento gli errori trovati e le relative soluzioni;
  5. **Aggiornamento della versione**: dopo che il documento viene corretto dall'autore, il verificatore deve aggiornare la versione del documento;
  6. **Versione**: sia $X.Y.Z$ la versione del documento, dopo la verifica, il valore di $Z$ viene incrementato di $1$, se le modifiche apportate al documento si limitano al contenuto e non modificano la struttura del documento, ovvero l'indice non viene modificato; altrimenti il valore di $Y$ viene incrementato di $1$ e $Z$ viene azzerato.

## 11.2 Verifica del codice

Il verificatore deve effettuare dei controlli di conformità sul codice prodotto. Questo controllo deve essere effettuato in modo sistematico e ripetitivo.

### 11.2.1 *Trigger*

- Viene prodotto un incremento sulla *code base*;
- Un componente di SWEnergy segnala la necessità di una verifica.

### 11.2.2 Scopo

- Evidenziare gli errori nel codice e segnalarli al programmatore;
- Assicurarsi che il codice soddisfi le norme qui sotto descritte;
- Convalidare l'incremento di codice per garantirne l'integrità agli altri componenti di SWEnergy

### 11.2.3 Norme

- **Commenti:** per ciascuna funzione o metodo, deve essere spiegato lo scopo. In particolare, deve essere sempre presenta la spiegazione dei parametri in ingresso e del valore di ritorno;
- **Test:** per ciascuna funzione o metodo, deve essere presente almeno un test che ne verifica il corretto funzionamento e fornisce un esempio di utilizzo;
- **Nomi:** i nomi delle variabili devono essere significativi e devono essere scritti in lingua italiana. Di seguito sono riportate le regole di forma per ciascun tipo di variabile:
  - **Variabili:** devono essere scritte in minuscolo e devono essere separate da underscore (es. `nome_variabile`);
  - **Costanti:** devono essere scritte in maiuscolo e devono essere separate da underscore (es. `NOME_COSTANTE`);
  - **Interfacce:** la prima lettera di ogni parola è maiuscola e le parole sono unite senza spazi (es. `NomeInterfaccia`);

- **Classi:** la prima lettera di ogni parola è maiuscola e le parole sono unite senza spazi (es. `NomeClasse`);
- **Metodi:** devono essere scritte in minuscolo e devono essere separate da underscore (es. `nome_metodo`);
- **Funzioni:** devono essere scritte in minuscolo e devono essere separate da underscore (es. `nome_funzione`);
- ***File*:** devono essere scritte in minuscolo e devono essere separate da underscore (es. `nome_file`). In ogni *file* ci può essere al più una classe o un'interfaccia che ha lo stesso nome del *file*.

#### 11.2.4 Svolgimento

- **Correzione del codice:** il verificatore deve controllare che per ciascuna funzione o metodo sia presente una descrizione dello scopo, dei parametri in ingresso e del valore di ritorno. Di seguito sono riportati i passi da seguire:
  1. **Commenti:** il verificatore legge i commenti della funzione e ne intuisce lo scopo;
  2. **Funzionamento:** il verificatore legge il corpo della funzione o del metodo e ne verifica il funzionamento staticamente;
  3. **Test:** il verificatore verifica che sia presente almeno un test per la funzione o il metodo;
  4. **Nomi:** il verificatore controlla che i nomi definiti dal programmatore rispettino le regole di forma definite precedentemente;
  5. **Correzioni:** il verificatore riporta gli errori riscontrati al programmatore;
  6. **Aggiornamento della versione:** dopo che il codice viene corretto dal programmatore, il verificare deve aggiornare la versione del codice;
  7. **Versione:** sia $X.Y.Z$ la versione del codice, dopo la verifica, il valore di $Z$ viene incrementato di $1$, se le modifiche apportate al codice non aggiungono nuove funzionalità. Se invece le modifiche apportate al codice aggiungono nuove funzionalità, il valore di $Y$ viene incrementato di $1$ e il valore di $Z$ viene reimpostato a $0$. Una funzionalità coincide con un requisito.