SWEmming Pool

swemming.pool@gmail.com

Norme di Progetto

Informazioni sul documento

Responsabile	Elia Pasquali
Redattori	Sebastiano Sanson, Enrico Bacci Bonivento
Verificatori	Fabio Pantaleo
Uso	Interno
Destinatari	*SyncLab*
	Prof. Vardanega Tullio
	Prof. Cardin Riccardo
	Gruppo *SWEmming Pool*

Sommario

Questo documento descrive gli strumenti, le regole e le convenzioni che il gruppo deve rispettare durante lo svolgimento del progetto.

Registro delle modifiche

Versione	Data	Nominativo	Ruolo	Descrizione
2.0.0	2023-05-02	Ennio Italiano	Verificatore	Verificati processi primari.
1.0.1	2023-04-26	Sebastiano Sanson	Analista	Completati processi primari.
1.0.1	2023-03-22	Nicolò Trinca	Verificatore	Verifica processi di sviluppo.
1.0.0	2023-03-21	Sebastiano Sanson	Responsabile	Completati processi di sviluppo.
1.0.0	2023-02-09	Fabio Pantaleo	Verificatore	Verifica del documento.
0.2.3	2023-01-18	Elia Pasquali	Responsabile	Effettuate modifiche a gestione delle infrastrutture.
0.2.2	2023-01-17	Enrico Bacci Bonivento	Amministratore	Effettuate alcune modifiche e aggiunte.
0.2.1	2023-01-09	Nicolò Trinca	Analista	Aggiunte metriche di qualità.
0.2.0	2023-01-08	Enrico Bacci Bonivento	Amministratore	Ampliamento dei processi.
0.1.0	2022-11-29	Sebastiano Sanson	Amministratore	Ampliamento dei processi.
0.0.6	2022-11-27	Enrico Bacci Bonivento Sebastiano Sanson	Amministratore	Ampliamento dei processi.
0.0.5	2022-11-24	Enrico Bacci Bonivento Sebastiano Sanson	Amministratore	Prima stesura processi organizzativi.
0.0.4	2022-11-22	Enrico Bacci Bonivento Sebastiano Sanson	Amministratore	Prima stesura processi di supporto.
0.0.3	2022-11-21	Enrico Bacci Bonivento Sebastiano Sanson	Amministratore	Prima stesura processi primari.
0.0.2	2022-11-20	Enrico Bacci Bonivento Sebastiano Sanson	Amministratore	Stesura introduzione.
0.0.1	2022-11-18	Elia Pasquali	Responsabile	Creata struttura del documento in LaTeX.

Contenuti

1 Introduzione **1**
- 1.1 Scopo del documento . . . 1
- 1.2 Scopo del progetto . . . 1
- 1.3 Glossario . . . 1
- 1.4 Riferimenti . . . 1
  - 1.4.1 Riferimenti normativi . . . 1
  - 1.4.2 Riferimenti informativi . . . 1

2 Processi Primari **2**
- 2.1 Fornitura . . . 2
  - 2.1.1 Scopo . . . 2
  - 2.1.2 Attività . . . 2
- 2.2 Sviluppo . . . 2
  - 2.2.1 Scopo . . . 2
  - 2.2.2 Attività . . . 2
    - 2.2.2.1 Analisi dei requisiti . . . 2
    - 2.2.2.2 Progettazione . . . 3
      - 2.2.2.2.1 Requirements and Technology Baseline . . . 3
      - 2.2.2.2.2 Product Baseline . . . 4
    - 2.2.2.3 Codifica . . . 4
      - 2.2.2.3.1 Norme di codifica . . . 4
      - 2.2.2.3.2 Qualità del codice . . . 5
      - 2.2.2.3.3 Strumenti di codifica . . . 5
    - 2.2.2.4 Test . . . 6

3 Processi di Supporto **6**
- 3.1 Scopo . . . 6
- 3.2 Gestione della documentazione . . . 6
  - 3.2.1 Implementazione . . . 6
  - 3.2.2 Design e sviluppo . . . 7
    - 3.2.2.1 Documentazione . . . 7
    - 3.2.2.2 Verbali . . . 7
    - 3.2.2.3 Norme tipografiche . . . 8
    - 3.2.2.4 Convenzioni nomenclatura file . . . 8
    - 3.2.2.5 Formato data . . . 8
    - 3.2.2.6 Elenchi . . . 8
    - 3.2.2.7 Glossario . . . 8
    - 3.2.2.8 Stile del testo . . . 8
    - 3.2.2.9 Sigle . . . 8
  - 3.2.3 Produzione documenti . . . 9
  - 3.2.4 Elementi grafici . . . 9
    - 3.2.4.1 Tabelle . . . 9
    - 3.2.4.2 Immagini . . . 9
    - 3.2.4.3 Diagrammi UML . . . 9
  - 3.2.5 Strumenti . . . 10
- 3.3 Gestione della configurazione . . . 10
  - 3.3.1 Utilizzo di Git . . . 10
- 3.4 Gestione del controllo qualità . . . 10
  - 3.4.1 Scopo . . . 10
  - 3.4.2 Denominazione metriche . . . 10

3.5 Gestione di verifica . . . 10
3.5.1 Scopo . . . 10
3.5.1.1 Analisi statica . . . 11
3.5.1.2 Analisi dinamica . . . 11
3.6 Gestione di validazione . . . 11
3.6.1 Scopo . . . 11

4 Processi Organizzativi **11**
4.1 Gestione Organizzativa . . . 11
4.1.1 Scopo . . . 11
4.1.2 Attività di gestione . . . 12
4.1.3 Gestione delle comunicazioni . . . 12
4.1.4 Gestione delle riunioni . . . 12
4.1.4.1 Riunioni interne . . . 12
4.1.4.2 Riunioni esterne . . . 12
4.1.5 Ruoli di progetto . . . 13
4.1.5.1 Responsabile di progetto . . . 13
4.1.5.2 Amministratore di progetto . . . 13
4.1.5.3 Analista . . . 13
4.1.5.4 Progettista . . . 13
4.1.5.5 Programmatore . . . 13
4.1.5.6 Verificatore . . . 13
4.2 Gestione delle Infrastrutture . . . 13
4.2.1 Ticketing . . . 13
4.2.2 Gestione dei Rischi . . . 14
4.2.3 Strumenti . . . 14
4.2.4 Configurazione degli Strumenti . . . 14
4.2.4.1 Tipi di file . . . 15
4.3 Formazione del Personale . . . 15

5 Standard ISO/IEC 12207:1997 **15**
5.1 Processi Primari . . . 16
5.1.1 Acquisition . . . 16
5.1.2 Supply . . . 16
5.1.3 Development . . . 16
5.1.4 Maintenance . . . 17
5.2 Processi Organizzativi . . . 17
5.3 Processi di Supporto . . . 18
5.3.1 Gestione della documentazione . . . 18
5.3.2 Gestione della configurazione . . . 18
5.3.3 Gestione della qualità . . . 18
5.3.4 Verifica . . . 18
5.3.5 Validazione . . . 18
5.3.6 Risoluzione dei problemi . . . 18
5.3.7 Usabilità . . . 18
5.3.8 Valutazione del prodotto . . . 19

6 Standard di qualità ISO/IEC 9126 **19**
6.1 Modello della qualità del software . . . 19
6.2 Qualità interna . . . 21
6.2.1 Metriche per la qualità interna . . . 21
6.3 Qualità esterna . . . 21

6.3.1 Metriche per la qualità esterna 21
6.4 Qualità in uso 21
6.4.1 Metriche per la qualità in uso 21

7 Metriche della qualità **21**
7.1 Metriche per la qualità di prodotto 21
7.1.1 MPD01: Indice di Gulpease 21
7.1.2 MPD02: Copertura requisiti obbligatori 22
7.1.3 MPD03: Copertura requisiti desiderabili 22
7.1.4 MPD04: Copertura requisiti opzionali 22
7.1.5 MPD05: Facilità di utilizzo 22
7.1.6 MPD06: Versioni browser supportate 23
7.1.7 MPD07: Solidity Statement Coverage 23
7.1.8 MPD08: Solidity Branch Coverage 23
7.1.9 MPD09: Solidity Function Coverage 23
7.1.10 MPD10: Solidity Line Coverage 23
7.1.11 MPD11: Frontend Statement Coverage 23
7.1.12 MPD12: Frontend Branch Coverage 23
7.1.13 MPD13: Frontend Function Coverage 24
7.1.14 MPD14: Frontend Line Coverage 24
7.1.15 Obiettivi qualità di prodotto 24
7.2 La qualità di processo 24
7.2.1 MPC01: Earned Value (EV) 25
7.2.2 MPC02: Actual Cost (AC) 25
7.2.3 MPC03: Planned Value (PV) 25
7.2.4 MPC04: Cost Variance (CV) 25
7.2.5 MPC05: Schedule Variance (SV) 25
7.2.6 MPC06: Estimated At Completion (EAC) 25
7.2.7 MPC07: Estimate To Complete (ETC) 26
7.2.8 MPC08: Requirements Stability Index (RSI) 26
7.2.9 MPC09: Passed Tests 26
7.2.10 MPC10: Metrics Satisfied 26
7.2.11 MPC11: Risks Found 26
7.2.12 Obiettivi qualità di processo 27

1 Introduzione

1.1 Scopo del documento

Il seguente documento ha l'obiettivo di definire le linee guida per tutti i processi del gruppo *SWEmming Pool*. Al suo interno sono presenti le norme, le tecnologie e gli strumenti che il team intende adottare.

Ogni membro del gruppo si impegna ad adottare queste "misure" per lo svolgimento di ogni attività finalizzata al progetto.

1.2 Scopo del progetto

Al giorno d'oggi poter garantire l'autenticità e la veridicità di una recensione$_{\mathbf{G}}$ risulta essere un problema; il capitolato *Trustify* si pone come obiettivo quello di creare un servizio che, attraverso l'uso di uno *smart contract*$_{\mathbf{G}}$, permetta di effettuare una recensione collegata in modo univoco ad un pagamento.

Il prodotto atteso sarà composto da un contratto digitale e una *web app*$_{\mathbf{G}}$ che, attraverso il *wallet*$_{\mathbf{G}}$ *MetaMask*$_{\mathbf{G}}$ ne permetta l'interazione con esso, e da un server *API REST*$_{\mathbf{G}}$ che consenta ad un *e-commerce*$_{\mathbf{G}}$ di mostrare le recensioni ricevute all'interno del proprio sito.

1.3 Glossario

Al fine di evitare ambiguità nella terminologia usata all'interno del seguente documento è stato redatto un glossario, in cui vengono riportate le definizioni di termini tecnici, rilevanti o con un significato particolare.

Per indicare la presenza di un termine all'interno del glossario si è scelto di contrassegnarlo con $_{\mathbf{G}}$. Per non appesantire la lettura della documentazione verrà così contrassegnata solo la prima occorrenza di ogni termine in ciascun documento.

Per una consultazione completa si rimanda al *Glossario v2.0.0*.

1.4 Riferimenti

1.4.1 Riferimenti normativi

- Capitolato d'appalto C7: **Trustify - Authentic and verifiable reviews platform**: https://www.math.unipd.it/~tullio/IS-1/2022/Progetto/C7.pdf
  [Ultimo accesso: 23 maggio 2023]

1.4.2 Riferimenti informativi

- **Standard ISO/IEC 12207:1995**: https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf
  [Ultimo accesso: 23 maggio 2023];
- **Piano di Progetto** - *Piano di Progetto - v2.0.0*;
- **Piano di Qualifica** - *Piano di Qualifica - v2.0.0*;
- **Software Engineering - Ian Sommerville - 9th Edition (2010)**;
- **Angular coding style guide**: https://angular.io/guide/styleguide
  [Ultimo accesso: 2023-03-21];
- **Solidity coding style guide**: https://docs.soliditylang.org/en/v0.8.19/style-guide.html
  [Ultimo accesso: 2023-03-21];

2 Processi Primari

2.1 Fornitura

2.1.1 Scopo

Lo scopo del processo di fornitura è quello analizzare e definire le operazioni e risorse necessarie per fornire il prodotto finale in modo tale che rispetti i requisiti imposti dal committente.

2.1.2 Attività

- **Avvio:** si procede con un'analisi preliminare dei capitolati. Vengono effettuate riunioni tra i membri del gruppo per discutere le proposte e scambiarsi le opinioni.
- **Lettera di candidatura:** si redige una lettera di candidatura in cui si indica il termine ultimo di consegna e il preventivo dei costi per la realizzazione del progetto;
- **Aggiudicazione appalto:** fase in cui si riceve ufficialmente in carico il progetto;
- **Piano di Progetto:** il responsabile e gli amministratori redigono il *Piano di Progetto* che contiene le seguenti informazioni:
  - Analisi dei rischi.
  - Analisi dei costi.
  - Pianificazione delle attività.
  - Modello di sviluppo.
- **Piano di Qualifica:** i verificatori redigono il *Piano di Qualifica* che contiene le linee guida da adottare per garantire la qualità prefissata per il prodotto finale. Il documento conterrà i seguenti punti:
  - Qualità di processo.
  - Qualità di prodotto.
  - Specifiche dei test.
  - Standard di qualità.

2.2 Sviluppo

2.2.1 Scopo

Questo processo contiene tutte le azioni e attività finalizzate ad analisi dei requisiti, design, codifica e test.

2.2.2 Attività

2.2.2.1 Analisi dei requisiti

Viene effettuata la stesura dell'*Analisi dei Requisiti*, che comprende elenco e analisi dei requisiti funzionali, non funzionali e di qualità. Questo documento si prefigge l'obiettivo di descrivere in modo chiaro e preciso tutte le caratteristiche richieste esplicitamente o implicitamente dal committente. Al suo interno vengono analizzati:

- **Casi d'uso:** consiste in una descrizione di un insieme di azioni che si possono eseguire su un sistema per ottenere un determinato risultato. Le caratteristiche di un caso d'uso sono:
  - codice identificativo.

  - nominativo.
  - diagramma UML.
  - attore primario.
  - attori secondario.
  - pre-condizioni.
  - post-condizioni.
  - scenario principale.
  - estensioni.

  Il codice utilizzato in ciascun caso d'uso è formato dalle iniziali di Use Case seguite dal numero progressivo del caso d'uso in questione e i suoi eventuali sotto-casi relativi.

- **Requisiti:** ciascun requisito viene definito seguendo lo standard di codifica: **R[Importanza][Tipologia][Codice]**
  - importanza: può essere **1** obbligatorio, **2** desiderabile e **3** opzionale.
  - tipologia: può essere **F** per funzionale, **NF** per non funzionale, **Q** per qualità e **V** per vincolo.
  - codice: identificatore univoco del requisito in forma gerarchica.

Alla versione *v1.0.2* del documento *Analisi dei Requisiti* la struttura è la seguente:

- **Introduzione**: contiene una breve descrizione del documento e del prodotto software;
- **Descrizione generale**: contiene una descrizione degli obiettivi del prodotto software;
- **Casi d'uso**: contiene l'elenco dei vari casi d'uso con i relativi diagrammi e descrizioni;
- **Requisiti**: contiene l'elenco dei requisiti con le relative descrizioni e tracciamento con i casi d'uso;

2.2.2.2 Progettazione

Questa fase prevede la realizzazione della specifica architetturale del prodotto finale, partendo dai requisiti definiti e individuando le diverse parti corrispettive, che dovranno essere poi raggruppate fino ad arrivare ad ottenere un unico sistema.
Inoltre, in parallelo allo sviluppo dell'architettura, si procede alla definizione dei test di unità, di integrazione e di sistema.
Tale fase dovrà essere documentata in un documento denominato *Specifica Tecnica*, alla versione *v1.0.0* la struttura del documento è la seguente:

- **Introduzione**: contiene una breve descrizione del documento e del prodotto software;
- **Architettura del prodotto**: contiene una descrizione dettagliata dell'architettura del prodotto software, nello specifico vengono illustrate l'*Architettura logica*, l'*Architettura di deployment* e l'*Architettura di dettaglio* che comprende i *Diagrammi delle classi* e i *Design pattern* utilizzati;
- **Requisiti soddisfatti**: contiene l'elenco dei requisiti che sono stati soddisfati;

2.2.2.2.1 Requirements and Technology Baseline

Fase in cui viene studiato il problema per individuare i requisiti opportuni per il suo soddisfacimento e le tecnologie da utilizzare per la realizzazione del prodotto finale.
Si conclude con lo sviluppo di un *Proof of Concept* (*PoC*) che permetta di verificare la fattibilità del progetto.

2.2.2.2.2 Product Baseline
Fase in cui viene definita l'architettura del prodotto finale, infine viene effettuata la codifica, seguita dalla fase di testing.

2.2.2.3 Codifica
Fase in cui si procede con la codifica del prodotto finale, concretizzando così i risultati delle fasi precedenti.

2.2.2.3.1 Norme di codifica
Il codice sorgente deve essere scritto in modo chiaro e leggibile, seguendo le seguenti norme:

- **Web App** (le cui convenzioni di stile sono imposte dal framework *Angular*):
  - **Nomenclatura**
    - **Cartelle**: le cartelle devono avere il nome dell'elemento, che deve essere in minuscolo e, se composto da piu parole, queste devono essere separate con -. Ad esempio, i file del componente `nome-componente` saranno contenuti in una cartella con lo stesso nome;
    - **File**: il nome dei file deve essere in minuscolo e, se composto da piu parole, queste devono essere separate con -. Inoltre, in base alla tipologia di elemento, il file deve avere un'estensione specifica prima di quella del linguaggio:
      - `.component`: per i componenti;
      - `.service`: per i servizi;
      - `.pipe`: per i filtri;
      - `.directive`: per le direttive.
    - **Classi**: la nomenclatura delle classi è basata sulle regole di *upper CamelCase* e deve essere coerente con il nome dell'elemento corrispondente. Ad esempio, la classe rappresentante il componente `nome-componente` sarà chiamata `NomeComponente`;
    - **Metodi**: la nomenclatura dei metodi è basata sulle regole di *upper camelCase*;
    - **Variabili e argomenti delle funzione**: la nomenclatura delle variabili e degli argomenti delle funzioni è basata sulle regole di *lower camelCase*;
    - **Costanti**: i nomi delle costanti deve essere espressi in maiuscolo e, se composti da più parole, queste devono essere separate con _;
    - **Commenti**: i commenti dovranno essere inseriti prima dell'inizio di un costrutto e presentati in lingua italiana.
  - **Struttura progetto**: i file del progetto sono organizzati e suddivisi in cartelle, in base all'elemento a cui appartengono. Tutti i file e cartelle devono essere contenuti all'interno della cartella *src*. Ad esempio, il componente `nome-componente` avrà la seguente struttura:
    - `nome-componente.component.ts`: file contenente la logica del componente;
    - `nome-componente.component.html`: file contenente il template *HTML* del componente;
    - `nome-componente.component.css`: file contenente gli stili *CSS* del componente;
    - `nome-componente.component.spec.ts`: file contenente i test del componente;

    È inoltre possibile creare altri componenti figli all'interno della cartella di un componente.
  - **Indentazione**: l'indentazione deve essere di 4 spazi.

Smart Contract (le cui convenzioni di stile sono imposte dal linguaggio *Solidity*):

- **Nomenclatura**:
  - **File**: i nomi dei file seguono le regole di *upper CamelCase* e devono essere coerenti con i nomi dei contratti corrispondenti; se all'interno di un file si trovano più contratti, il nome di tale file deve corrispondere a quello del contratto principale;
  - **Contratti e Librerie**: la nomenclatura dei contratti e delle librerie è basata sulle regole di *upper CamelCase* e deve essere coerente con il nome del file corrispondente;
  - **Metodi**: la nomenclatura dei metodi è basata sulle regole di *lower camelCase*;
  - **Variabili**; la nomenclatura delle variabili è basata sulle regole di *lower camelCase*;
  - **Argomenti delle funzione**: la nomenclatura degli argomenti delle funzioni è basata sulle regole di *lower camelCase* e il carattere iniziale deve essere _;
  - **Costanti**: la nomenclatura delle costanti deve essere espressa in maiuscolo e, se composta da più parole, queste devono essere separate con _;
  - **Commenti**: i commenti dovranno essere inseriti prima dell'inizio di un costrutto e presentati in lingua italiana.
- **Struttura progetto** (convenzione imposta dal framework *Truffle*):
  - *contracts*: cartella contenente i contratti;
  - *build*: cartella contenente i file compilati;
  - *test*: cartella contenente i test;
- **Indentazione**: l'indentazione deve essere di 4 spazi.

API REST (alcune convenzioni che seguono regole di *Java*):

- **Nomenclatura**:
  - **File**: la nomenclatura dei file è basata sulle regole di *upper CamelCase*;
  - **Classi**: la nomenclatura delle classi è basata sulle regole di *upper CamelCase*;
  - **Metodi**: la nomenclatura dei metodi è basata sulle regole di *upper CamelCase*;
  - **Variabili e argomenti delle funzione**: la nomenclatura delle variabili e degli argomenti delle funzioni sono basate sulle regole di *lower camelCase*;
  - **Costanti**: la nomenclatura delle costanti deve essere espressa in maiuscolo e, se composte da più parole, queste devono essere separate con _;
  - **Commenti**: i commenti dovranno essere inseriti prima dell'inizio di un costrutto e presentati in lingua italiana.
- **Struttura progetto** (convenzioni imposte dal framework *Spring Boot*):
  - *src/main/java*: cartella contenente il codice sorgente;
  - *src/main/resources*: cartella contenente i file di configurazione;
  - *src/test/java*: cartella contenente i test.
- **Indentazione**: l'indentazione deve essere di 2 spazi.

2.2.2.3.2 Qualità del codice

La qualità del codice è definita dalle metriche, che fanno riferimento al *Piano di Qualifica v2.0.0* e dai requisiti di qualità, descritti nell'*Analisi dei Requisiti v2.0.0.*

2.2.2.3.3 Strumenti di codifica

Gli strumenti utilizzati per la codifica e di supporto ad essa sono:

- **Visual Studio Code**: *IDE* utilizzato per la stesura del codice per la *web app* e dello *smart contract*;

- **IntelliJ IDEA**: *IDE* utilizzato per la stesura del codice per l'*API REST*;
- **Angular**: *framework* utilizzato per la realizzazione della *web app*;
- **Solidity**: linguaggio utilizzato per la realizzazione dello *smart contract*;
- **Spring Boot**: *framework* utilizzato per la realizzazione dell'*API REST*;
- **Node.js**: *runtime environment* utilizzato per l'esecuzione di codice *JavaScript*;
- **Metamask**: *wallet* utilizzato per effettuare delle transazioni;
- **Truffle**: *framework* utilizzato per la realizzazione e la compilazione dello *smart contract*;

2.2.2.4 Test

Fase in cui si procede con la verifica della qualità del prodotto finale, attraverso l'esecuzione di test di sistema, ideati e implementati nelle fasi precedenti.

I test di sistema sono stati descritti nel documento *Piano di Qualifica v.2.0.0*, dove sono tracciati con i relativi requisiti e viene indicato il loro stato di implementazione. Il loro scopo è quello di determinare se il prodotto soddisfa i requisiti richiesti e se rispetta le norme di qualità stabilite.

3 Processi di Supporto

3.1 Scopo

In questa sezione vengono descritte le procedure per le seguenti fasi del progetto:

- documentazione;
- configurazione;
- qualità;
- verifica;
- validazione;
- controllo costi;
- risoluzione dei problemi.

3.2 Gestione della documentazione

Vengono documentate tutte le informazioni riguardanti il ciclo di vita di ciascun processo o attività. Queste attività si dividono nelle sezioni riportate di seguito.

3.2.1 Implementazione

Per tutti i documenti redatti vengono stabiliti titolo, scopo, destinatario, procedure di tutti i ruoli e le scadenze per le revisioni intermedie e finali.

3.2.2 Design e sviluppo

3.2.2.1 Documentazione

È stato creato un *template* per uniformare lo stile di tutti i documenti prodotti e agevolarne la produzione.
Nello specifico sono presenti:

- **Prima pagina** contenente:
  - logo;
  - indirizzo email del gruppo;
  - titolo del documento;
  - responsabile, redattori e verificatori;
  - uso (interno o esterno);
  - destinatari;
  - sommario.
- **Registro delle modifiche**: in ogni documento viene inserita una tabella contenente il registro delle modifiche che riporta:
  - versione;
  - data;
  - autore;
  - breve descrizione della modifica.
- **Indice**: ha lo scopo di fornire una visione gerarchica dei contenuti del documento e di facilitare la ricerca di informazioni specifiche;
- **Contenuto principale**: qui è presente l'effettivo contenuto del documento.

3.2.2.2 Verbali

È stato creato un template per uniformare lo stile di tutti i verbali prodotti e agevolarne la produzione. Nello specifico sono presenti:

- **Prima pagina** contenente:
  - logo;
  - indirizzo email del gruppo;
  - titolo del documento;
  - responsabile, redattori e verificatori;
  - uso (interno o esterno);
  - destinatari;
  - sommario.
- **Indice**;
- **Ordine** contenente:
  - ora di inizio/termine;
  - luogo;
  - partecipanti interni e esterni;
  - ordine del giorno.
- **Resoconto**: contiene una descrizione dettagliata degli argomenti trattati e delle decisioni prese.

3.2.2.3 Norme tipografiche

3.2.2.4 Convenzioni nomenclatura file
Escludendo l'estensione, le regole di nomenclatura dei file sono le seguenti:

- i nomi dei file devono essere completamente in minuscolo;
- viene utilizzato il carattere underscore _ per separare le parole;
- omissione delle preposizioni.

3.2.2.5 Formato data
Viene utilizzato il formato **YYYY-MM-DD** per le date.

- **YYYY** rappresenta l'anno (4 cifre);
- **MM** rappresenta il mese (2 cifre);
- **DD** rappresenta il giorno (2 cifre).

3.2.2.6 Elenchi

- **Elenco puntato**: viene utilizzato per elencare elementi in cui non ha importanza l'ordine;
- **Elenco ordinato**: viene utilizzato per elencare elementi in cui ha importanza l'ordine o la priorità.

3.2.2.7 Glossario
Le convenzioni adottate per fare riferimento a termini specifici presenti nel glossario sono le seguenti:

- un termine presente nel glossario viene marcato con $_{G}$;
- in ogni documento un termine viene marcato solo alla sua prima occorrenza.

3.2.2.8 Stile del testo

- `Monospaced`: viene utilizzato per i comandi e snippet;
- *Italic*: viene utilizzato per i termini di dominio e per citare altri documenti;
- **Bold**: viene utilizzato per i termini da enfatizzare, da definire e per citare eventuali standard.

3.2.2.9 Sigle
Nella stesura della documentazione vengono utilizzate alcune sigle per termini ricorrenti. Vengono scritte in maiuscolo le iniziali dei sostantivi e in minuscolo le iniziali delle eventuali preposizioni.

Le sigle adottate sono le seguenti:

- **AdR**: *Analisi dei Requisiti*;
- **MU**: *Manuale Utente*;
- **MS**: *Manuale Sviluppatore*;
- **PdP**: *Piano di Progetto*;

- **PdQ**: *Piano di Qualifica*;
- **NdP**: *Norme di Progetto*;
- **ST**: *Specifica Tecnica*;
- **SdF**: *Studio di Fattibilità*;
- **RTB**: *Requirements and Technology Baseline*;
- **PB**: *Product Baseline*;
- **CA**: *Customer Acceptance*;
- **PoC**: *Proof of Concept.*

Per definire le sigle dei ruoli si è deciso di aggiungere la seconda lettere in minuscolo per evitare omonimia:

- **Am**: Amministratore;
- **An**: Analista;
- **Pr**: Programmatore;
- **Pt**: Progettista;
- **Re**: Responsabile di progetto;
- **Ve**: Verificatore.

3.2.3 Produzione documenti

Per agevolare la produzione dei documenti è stato creato un template che contiene tutte le informazioni necessarie per la redazione di un documento; a tale scopo è stata definita una cartella `componenti_comuni` che contiene:

- `src`: contiene i file `command.tex` e `packages.tex` che contengono comandi e pacchetti necessari per la produzione e compilazione del documento;
- `img`: contiene tutte le immagini utilizzate nel template.

3.2.4 Elementi grafici

3.2.4.1 Tabelle

In ciascun documento prodotto, qualora fosse presente una tabella, essa viene accompagnata da una didascalia contenete la numerazione della tabella stessa e un breve titolo che ne riassuma il contenuto.

3.2.4.2 Immagini

In ogni immagine inserita all'interno dei documenti prodotti viene fornita una breve didascalia per descriverne chiaramente il contenuto

3.2.4.3 Diagrammi UML

I diagrammi *UML* inseriti rispettano lo standard UML 2.0. Anch'essi vengono accompagnati da una breve didascalia.

3.2.5 Strumenti

Per la redazione dei documenti vengono utilizzati i seguenti strumenti:

- LaTeX: per la redazione dei documenti;
- *VSCode*: per la gestione del codice sorgente e della documentazione;
- *StarUML*: per la creazione dei diagrammi *UML*;
- *GanttProject*: per la creazione dei diagrammi di Gantt;

3.3 Gestione della configurazione

3.3.1 Utilizzo di Git

Per la gestione della configurazione del progetto viene utilizzato il sistema di versionamento **Git**. Il gruppo ha adottato il modello di branching **GitFlow** per la gestione delle versioni del progetto.

3.4 Gestione del controllo qualità

3.4.1 Scopo

Il processo di Gestione della Qualità ha l'obiettivo di definire indici precisi per tutte le attività che riguardano la verifica e la validazione. Questo permette di garantire che il livello di qualità desiderato venga rispettato. Per maggiori informazioni sulle metriche utilizzate, si fa riferimento al *Piano di Qualifica v2.0.0*. Questo documento fornirà una trattazione dettagliata su come vengono misurati i risultati e garantire la qualità nei progetti.

3.4.2 Denominazione metriche

Le metriche utilizzate sono identificate tramite un codice univoco:

M[Utilizzo]-[Acronimo]

Nella quale **[Utilizzo]** indica se la metrica è:

- **PC**: per la qualità di processo;
- **PD**: per la qualità di prodotto.

Mentre **[Acronimo]** indica il nome della metrica.

3.5 Gestione di verifica

3.5.1 Scopo

Il processo di verifica ha lo scopo di garantire che il prodotto finale soddisfi i requisiti richiesti, fornendo un risultato corretto, coeso e completo. Tale processo riceve in input quanto prodotto fino a quello momento e lo restituisce in uno stato conforme alle aspettative. Per ottenere ciò vengono utilizzati i processi di test e analisi.

3.5.1.1 Analisi statica

Consiste nell'analizzare il codice sorgente e la documentazione del prodotto per verificarne la correttezza (assenza di difetti e/o errori).

I metodi per effettuare analisi statica sono catalogabili in due categorie: manuali e automatici. Quelli manuali sono due:

- **Walkthrough:** mira a rivelare la presenza di difetti tramite una lettura critica a largo spettro e senza assunzioni di alcun genere. Il Walkthrough va pianificato e, in seguito, va discusso quanto individuato per correggere i difetti, documentando le attività svolte;
- **Inspection:** mira a rivelare la presenza di difetti tramite una lettura mirata del prodotto da parte di verificatori distinti dai programmatori secondo una strategia che si focalizza sui punti ritenuti critici.

3.5.1.2 Analisi dinamica

Consiste nell'analizzare il prodotto in esecuzione per verificare che il comportamento sia conforme a quanto previsto e che non si presentino anomalie o malfunzionamenti. I criteri da seguire per definire i test per l'analisi dinamica vanno inseriti nel *Piano di Qualifica*. Un test ha lo scopo di far fallire il programma e se ci riesce, quel test va aggiunto alla suite dei test del progetto in modo permanente.

Ci sono vari tipi di test:

- **Test di unità:** *unit test*, sono test che vengono eseguiti su singole unità di codice, per verificare che il codice funzioni correttamente;
- **Test di integrazione:** *integration test*, sono test che vengono eseguiti su più unità di codice, per verificare una corretta interazione tra le componenti;
- **Test di sistema:** *system test*, sono test che verificano la copertura dei requisiti e vengono definiti durante la fase di analisi dei requisiti;
- **Test di regressione:** *regression test*, sono test che vengono eseguiti dopo una modifica del codice per verificare che le modifiche non abbiano introdotto errori.

3.6 Gestione di validazione

3.6.1 Scopo

La validazione è la conferma finale che il prodotto sia conforme alle attese del committente. Dopo di essa il prodotto può essere rilasciato.

4 Processi Organizzativi

4.1 Gestione Organizzativa

4.1.1 Scopo

L'obiettivo di tale sezione è quello di definire e gestire i processi di progetto secondo lo standard **ISO/IEC 12207:1995**. Nello specifico:

- Stabilire il lavoro da svolgere in modo da raggiungere gli obiettivi del progetto;
- Stabilire quale modello di sviluppo adottare;
- definire scadenze, assegnare i compiti, quantificare i rischi e le risorse necessarie;
- Stabilire i criteri di qualità e di verifica;

- Definire le modalità e strumenti di comunicazione;
- Definire i ruoli di progetto e i relativi compiti.

4.1.2 Attività di gestione

- Definizione dell'obiettivo;
- Istanziazione dei processi;
- Pianificazione delle scadenze, costi e risorse;
- Assegnazione dei compiti e ruoli;
- Esecuzione dei processi;
- Revisione e valutazione dei risultati.

4.1.3 Gestione delle comunicazioni

- **Comunicazioni interne**: comunicazioni tra i membri del gruppo;
  - **Comunicazione orale**: la comunicazione orale è il metodo più veloce e diretto di comunicazione. È importante che questo tipo di comunicazione sia chiara e concisa, in modo da evitare equivoci e malintesi.
    Lo strumento adottato dal gruppo a tale scopo è *Discord*.
  - **Comunicazione via chat**: la comunicazione via chat è utile per comunicare informazioni urgenti, chiarire dubbi e fissare incontri.
    Lo strumento adottato dal gruppo è *Telegram*.
- **Comunicazioni esterne**: comunicazioni con il committente e con il proponente.

  Lo strumento principale per le comunicazione con soggetti esterni al gruppo è la posta elettronica all'indirizzo *swemming.pool@gmail.com*. La comunicazione via mail è utile per chiarire dubbi e fissare incontri e viene usata per le comunicazioni formali.

  Inoltre l'azienda *SyncLab* ha fornito un canale *Discord* apposito per le comunicazioni via chat con il gruppo.

4.1.4 Gestione delle riunioni

4.1.4.1 Riunioni interne

È compito del responsabile di progetto, in concordanza con il gruppo, organizzare le riunioni interne. Gli incontri possono affrontare argomenti di vario tipo: definizione delle attività da svolgere, allineamento su ciò che è stato fatto con revisione e validazione, ecc.

Si ritiene valida una riunione quando sono presenti almeno quattro membri del gruppo.

Viene redatto un verbale dopo ogni riunione in cui si sono approvate le attività svolte nel periodo precedente.

4.1.4.2 Riunioni esterne

È compito del responsabile di progetto, in concordanza con il gruppo e il proponente, organizzare le riunioni esterne, che avranno luogo su *Google Meet*.

Viene redatto un verbale al termine di ogni riunione esterna in modo da riportarne il contenuto, che dovrà poi essere approvato dal responsabile.

4.1.5 Ruoli di progetto

Di seguito vengono descritti i ruoli di progetto e le relative responsabilità; a fini accademici, ciascun membro del gruppo deve ricoprire ogni ruolo almeno una volta.

4.1.5.1 Responsabile di progetto
Il responsabile di progetto e la figura principale in quanto deve pianificare e coordinare tutte le attività del gruppo. Deve inoltre occuparsi delle comunicazioni con il proponente e con il committente, gestire le risorse e dare l'approvazione finale dei documenti.

4.1.5.2 Amministratore di progetto
L'Amministratore di Progetto è incaricato di gestire, controllare e curare le risorse e l'ambiente di lavoro. Si occupa della redazione del documento *Norme di Progetto* e si assicura della messa in opera delle regole definite al suo interno; deve inoltre tenere traccia del versionamento del prodotto.

4.1.5.3 Analista
L'analista è un ruolo essenziale nella fase iniziale in cui viene analizzato e studiato il problema in modo da definirne i requisiti, le funzionalità e le caratteristiche. Deve dunque avere una conoscenza del dominio applicativo e una notevole esperienza personale; deve inoltre occuparsi della redazione del documento *Analisi dei Requisiti*.

4.1.5.4 Progettista
Il progettista segue il lavoro dell'analista, occupandosi di trovare una soluzione al problema che soddisfi i requisiti definiti. Deve dunque avere una competenza tecnica e tecnologica aggiornata.

4.1.5.5 Programmatore
Il programmatore si occupa della realizzazione del prodotto, codificandone le soluzioni stabilite dal progettista. È inoltre responsabile di definirne i test per verificarne la correttezza l'implementazione.

4.1.5.6 Verificatore
Il verificatore è un ruolo che accompagna tutto lo svolgimento del progetto, in quanto si occupa di verificare e validare il prodotto in modo che soddisfi le norme e le attese prefissate.

4.2 Gestione delle Infrastrutture

4.2.1 Ticketing

Il ticketing è un sistema di gestione delle attività che permette di tenere traccia delle attività svolte e di quelle da svolgere. In questa maniera il responsabile di progetto ha sempre sotto occhio lo stato di avanzamento del progetto.

Si è scelto di usare lo strumento fornito da *GitHub* per la gestione dei ticket; tramite l'integrazione tra l'*Issue Tracking* e le sue *Project Board*, questo si presenta sotto forma di tabella in cui ciascuna attività si trova necessariamente in uno stato definito tra quelli elencati di seguito:

- **To do**: attività da svolgere;
- **In progress**: attività in corso di svolgimento;
- **In review**: attività in attesa di revisione;
- **Done**: attività completata.

4.2.2 Gestione dei Rischi

Il responsabile di progetto deve tenere traccia dei possibili rischi e delle relative azioni correttive. Le tipologie di rischio posso essere:

- **RT: Rischi Tecnologici**: sono quelli che possono essere associati a una tecnologia o a un processo di sviluppo;
- **RO: Rischi Organizzativi**: sono quelli che possono essere associati a una mancanza di organizzazione o di pianificazione;
- **RI: Rischi Interpersonali**: sono quelli che possono essere associati a una mancanza di comunicazione o di collaborazione tra i membri del gruppo.

4.2.3 Strumenti

- *Telegram*: per le comunicazione rapide tra i membri del gruppo;
- *Discord*: per le riunioni (interne ed esterne);
- *Google Suite*: per la gestione delle email, delle bozze di documenti, diari di bordo e del calendario;
- *Git*: per il controllo di versionamento;
- *GitHub*: per la condivisione e salvataggio remoto del codice sorgente e documentazione.

A livello di sistema operativo non sono presenti vincoli prefissati dal proponente, pertanto ogni membro del gruppo utilizza il sistema operativo con cui ha maggior familiarità. Vengono utilizzati *Windows*, *Linux* e *MacOS*.

4.2.4 Configurazione degli Strumenti

Il gruppo come prima cosa ha creato un canale *Discord* per le comunicazioni. Successivamente ha creato un account *Google* per poter usufruire dei vari servizi offerti da *Google Suite*, quali *Gmail*, *Google Drive* e *Google Calendar*. Infine è stata creata una organizzazione del gruppo su *GitHub*, all'interno della quale si è deciso di creare dei *repository* divisi tra pubblici e privati:

- Repository pubblici:
  - **documentazione**: per la condivisione della documentazione definitiva.
- Repository privati:
  - **docs**: per gestire la documentazione in fase di redazione;
  - **proof-of-concept**: per la condivisione del codice sorgente del *Proof of Concept*;
  - **trustify-smartcontract**: per gestire la codifica della parte legata al backend;
  - **trustify-webapp**: per gestire la codifica della parte legata al frontend;
  - **trustify-api**: per gestire la codifica della parte legata all'API REST.

L'indirizzo della organizzazione è https://github.com/SWEmming-Pool.

4.2.4.1 Tipi di file Le cartelle pubbliche di documentazione contengono solamente *pdf*. Nelle cartelle utilizzate per la redazione sono contenuti anche i vari file `.tex`, le immagini a supporto dei documenti, i file relativi agli strumenti di sviluppo (come ad esempio i file di configurazione delle *GitHub Action* o i `.gitignore` per nascondere i file superflui) e gli script di build o per la gestione dei container (per uno sviluppo su ambiente isolato e riproducibile).

Le cartelle contenenti codice sorgente per i prodotti software seguiranno la stessa metodologia applicata per i documenti, in modo da contenere strumenti supporto come script di build e container, oltre ai classici file di configurazione di *GitHub* come *.gitignore* e *GitHub Action*.

4.3 Formazione del Personale

Ciascun membro del gruppo è tenuto a provvedere alla propria formazione, eventualmente con l'aiuto degli altri componenti del team, al fine di garantire una qualità di lavoro adeguata. Nello specifico il gruppo dovrà fare riferimento alla seguente documentazione;

- *LaTeX*:
  - overleaf;
  - LaTeXpedia;
- *Git*: documentazione ufficiale;
- *GitHub*: documentazione ufficiale;
- *Smart contracts*: documentazione Ethereum;
- *Blockchain Ethereum*: documentazione ufficiale;
- *RPC*: Infura;
- *API REST*: AWS;
- *Wallet*: documentazione Ethereum;
- *Angular*: documentazione ufficiale;
- *Spring*: documentazione ufficiale;
- *web3j*: documentazione ufficiale;
- *web3js*: documentazione ufficiale;

5 Standard ISO/IEC 12207:1997

La norma ISO/IEC 12207:1997 è uno standard internazionale che fornisce linee guida per la gestione del ciclo di vita del software. Lo standard definisce un insieme di processi, attività e ruoli per lo sviluppo, la manutenzione e la gestione del software, e si applica a tutti i tipi di software, inclusi i sistemi informativi, i software embedded e i software di controllo.
I processi sono divisi in tre categorie:

- Processi primari;
- Processi organizzativi;
- Processi di supporto.

5.1 Processi Primari

Comprendono le attività direttamente legate allo sviluppo del software. Le attività sono le seguenti:

- Acquisition;
- Supply;
- Development;
- Maintenance.

5.1.1 Acquisition

Questo processo ha lo scopo di ottenere il prodotto richiesto dal cliente ed è suddiviso nelle seguenti attività:

- Preparazione dell'acquisizione;
- Scelta del fornitore;
- Monitoraggio dei fornitori;
- Accettazione del cliente.

5.1.2 Supply

Questo processo ha lo scopo di fornire al cliente il prodotto/servizio che soddisfa i requisiti concordati.

È suddiviso nelle seguenti attività:

- Proposal preparation;
- Contract;
- Planning;
- Execution and control;
- Review and evaluation;
- Release and completion.

5.1.3 Development

Questo processo ha lo scopo di sviluppare un prodotto software che indirizzi le esigenze del cliente. Le attività del processo sono suddivise rispetto al ruolo dello sviluppatore e a quello del cliente e sono le seguenti:

- Requirements elicitation;
- System requirements analysis;
- System architecture design;
- Software requirements analysis;
- Software architecture design;

- Software construction (code and unit test);
- Software integration;
- Software testing;
- System integration;
- System testing;
- Software installation.

5.1.4 Maintenance

È svolto simultaneamente alla precedente fase di Development.

Questo processo ha lo scopo di modificare il prodotto software dopo il suo rilasci per correggere i difetti, migliorare le sue prestazioni o altri attributi o adattarlo a cambiamenti nell'ambiente operativo.

È suddiviso nelle seguenti attività:

- Defect or request for change analysis;
- Change implementation;
- Review/acceptance of changes;
- Migration;
- Software withdraw.

5.2 Processi Organizzativi

Sono fondamentali per garantire che l'organizzazione coinvolta nello sviluppo del software sia in grado di fornire un ambiente adeguato e supportare efficacemente gli altri processi del ciclo di vita del software. Le attività sono le seguenti:

- **Formazione del personale:** questo processo ha lo scopo di fornire all'organizzazione risorse umane adeguate e di mantenere le loro competenze consistenti con le necessità del business;
- **Miglioramento del processo:** questo processo ha lo scopo di stabilire, valutare, controllare e migliorare il ciclo di vita del software;
- **Gestione delle infrastrutture:** questo processo ha lo scopo di mantenere un'infrastruttura stabile ed affidabile, necessaria a supportare le prestazioni di qualsiasi processo. L'infrastruttura può includere hardware, software, metodi, tools, tecniche, standard ed utilità per lo sviluppo, operatività o manutenzione;
- **Gestione dei processi:** Questo processo ha lo scopo di organizzare, monitorare e controllare l'avvio e le prestazioni di un processo per il raggiungimento dei loro obiettivi in accordo con quelli di business dell'organizzazione. Il processo è stabilito da una organizzazione per assicurare la consistente applicazione di pratiche per l'uso dall'organizzazione e nei progetti.

5.3 Processi di Supporto

I processi di supporto forniscono supporto a questi processi primari, come la gestione della configurazione, la gestione della qualità e la gestione dei test. Inoltre, essi forniscono il supporto necessario all'organizzazione, come la gestione delle risorse umane e la gestione delle infrastrutture.

Le attività sono le seguenti:

- Documentazione;
- Gestione delle versioni e della configurazione;
- Risoluzione dei problemi;
- Verifica e validazione.

5.3.1 Gestione della documentazione

Questo processo garantisce lo sviluppo e la manutenzione delle informazioni prodotte e registrate relativamente al prodotto software.

5.3.2 Gestione della configurazione

Questo processo ha lo scopo di definire e mantenere l'integrità di tutti i componenti della configurazione e di renderli accessibili a chi ne ha diritto.

5.3.3 Gestione della qualità

Questo processo ha lo scopo di assicurare che tutti i prodotti di fase siano conformi con i piani e gli standard definiti.

5.3.4 Verifica

Questo processo ha lo scopo di confermare che ciascun prodotto o servizio realizzato da un processo soddisfi i requisiti specificati.

Il processo di Verifica deve essere integrato nei processi di Sviluppo, Fornitura e Manutenzione.

5.3.5 Validazione

Questo processo ha lo scopo di confermare che i requisiti siano rispettati quando uno specifico prodotto sia utilizzato nell'ambiente destinatario.

5.3.6 Risoluzione dei problemi

Questo processo ha lo scopo di assicurare che tutti i problemi individuati siano analizzati e risolti secondo trend riconosciuti.

5.3.7 Usabilità

Questo processo ha lo scopo di assicurare che siano prese in considerazione, ed opportunamente indirizzate, le considerazioni espresse dalle parti interessate relativamente alla facilità d'uso del prodotto finale da parte degli utenti a cui è rivolto, al supporto che ne riceverà, alla formazione, all'incremento della produttività, alla qualità del lavoro, all'accettazione del prodotto stesso.

5.3.8 Valutazione del prodotto

Questo processo ha lo scopo principale di assicurare, tramite test e misure, che il prodotto soddisfi le necessità esplicite ed implicite degli utilizzatori del prodotto stesso.

6 Standard di qualità ISO/IEC 9126

ISO/IEC 9126 è uno standard internazionale per la valutazione della qualità software. Il gruppo *SWEmming Pool* ha deciso di fare riferimento a questo standard poiché considerato un caposaldo in materia di qualità. Questo standard permette di diffondere una comprensione comune degli obiettivi di un progetto software.

6.1 Modello della qualità del software

Il modello di qualità stabilito dallo standard si articola in sei caratteristiche principali, ognuna delle quali a sua volta presenta delle sotto-caratteristiche, misurabili tramite delle metriche di qualità. Di seguito sono esposte le sei caratteristiche sopra citate:

- **Funzionalità:** insieme di attributi riguardanti un insieme di funzioni e le loro proprietà. Tali funzioni mirano a soddisfare requisiti stabiliti o implicitamente dedotti. Le sotto-caratteristiche della funzionalità sono:
  - **Adeguatezza:** capacità del prodotto di fornire un insieme di funzioni in grado di svolgere compiti e soddisfare obiettivi prefissati;
  - **Accuratezza:** capacità del prodotto di fornire i risultati desiderati con la precisione richiesta;
  - **Interoperabilità:** capacità del prodotto di interagire ed operare con uno o più sistemi;
  - **Sicurezza:** capacità del prodotto di proteggere informazioni e dati monitorando gli accessi ad essi;
  - **Aderenza alle funzionalità:** grado di adesione del prodotto agli standard scelti dal gruppo, alle convenzioni e ai regolamenti.
- **Affidabilità:** insieme di attributi riguardanti la capacità del prodotto di mantenere un dato livello di performance sotto condizioni di esecuzione prestabilite. Le sotto-caratteristiche dell'affidabilità sono:
  - **Maturità:** capacità del prodotto di evitare errori e risultati non corretti durante l'esecuzione;
  - **Tolleranza ai guasti:** capacità del prodotto di conservare il livello di prestazioni anche in caso di malfunzionamenti o di uso inappropriato del prodotto;
  - **Recuperabilità:** capacità di un prodotto di ripristinare il livello di prestazioni e di recupero delle informazioni rilevanti, a seguito di un malfunzionamento. Il periodo di inaccessibilità del prodotto a seguito di un errore è valutato proprio dalla recuperabilità;
  - **Aderenza all'affidabilità:** grado di adesione del prodotto a standard, regole e convenzioni inerenti all'affidabilità.
- **Efficienza:** insieme di attributi riguardanti il rapporto tra il livello delle prestazioni e la quantità di risorse usate durante la loro esecuzione, sotto condizioni prestabilite. Le sotto-caratteristiche dell'efficienza sono:

  - **Comportamento rispetto al tempo:** capacità di un prodotto di fornire appropriati tempi di risposta e di elaborazione e quantità di lavoro eseguendo le funzionalità richieste in date condizioni di lavoro;
  - **Utilizzo delle risorse:** capacità di un prodotto di usare appropriati numero e tipo di risorse durante la fase di esecuzione sotto condizioni di utilizzo date;
  - **Aderenza all'efficienza:** grado di adesione del prodotto a standard, regole e convenzioni inerenti all'efficienza.

- **Usabilità:** insieme di attributi riguardanti lo sforzo necessario all'utilizzo del prodotto e la valutazione individuale su tale uso da parte di un insieme di utenti. Le sotto-caratteristiche dell'usabilità sono:
  - **Comprensibilità:** facilità di comprensione dei concetti base del prodotto, per permettere all'utente di capire se il prodotto è appropriato;
  - **Apprendibilità:** facilità con cui un utente medio può comprendere il funzionamento del prodotto ed imparare ad usarlo;
  - **Operabilità:** misura della possibilità degli utenti di usare il prodotto in vari contesti e di adattarlo ai propri scopi;
  - **Attrattività:** misura della gradevolezza e dell'essere "attraente" del prodotto durante l'uso;
  - **Aderenza all'usabilità:** grado di adesione del prodotto a standard, regole e convenzioni inerenti all'usabilità.

- **Manutenibilità:** insieme di attributi riguardanti lo sforzo richiesto per apportare modifiche specifiche al prodotto. Le sotto-caratteristiche della manutenibilità sono:
  - **Analizzabilità:** misura della difficoltà incontrata nel diagnosticare un errore nel prodotto;
  - **Modificabilità:** facilità di apportare modifiche al prodotto originale o di introdurre nuove funzionalità; per modifiche si intendono cambiamenti al codice, alla progettazione o alla documentazione;
  - **Stabilità:** capacità del prodotto di evitare effetti indesiderati a causa di modifiche;
  - **Provabilità:** capacità del prodotto di essere verificato;
  - **Aderenza alla manutenibilità:** grado di adesione del prodotto a standard, regole e convenzioni inerenti alla manutenibilità.

- **Portabilità:** insieme di attributi riguardanti la capacità del software di essere trasferito da un ambiente di esecuzione ad un altro. Le sotto-caratteristiche della portabilità sono:
  - **Adattabilità:** capacità del prodotto di essere adattato per differenti ambienti operativi senza richiedere azioni specifiche diverse da quelle previste dal prodotto per quell'attività; include la scalabilità delle capacità interne del prodotto;
  - **Installabilità:** capacità del prodotto di essere installato in un dato ambiente;
  - **Coesistenza:** capacità di un prodotto di coesistere con altre applicazioni in ambienti comuni e condividere le risorse;
  - **Sostituibilità:** capacità di sostituire un altro software specifico indipendente, per lo stesso scopo e nello stesso ambiente di sviluppo;
  - **Aderenza alla portabilità:** capacità del prodotto di aderire a standard e convenzioni relative alla portabilità.

6.2 Qualità interna

6.2.1 Metriche per la qualità interna

Le metriche interne si applicano al prodotto non eseguibile durante la progettazione e la codifica. Sono anche dette misure statiche. Le misure effettuate permettono di prevedere il livello di qualità esterna ed in uso del prodotto finale, poiché gli attributi interni influiscono su quelli esterni e quelli in uso. Le metriche interne permettono di individuare eventuali problemi che potrebbero inficiare la qualità finale del prodotto prima che sia realizzato il prodotto eseguibile.

6.3 Qualità esterna

6.3.1 Metriche per la qualità esterna

Le metriche esterne misurano i comportamenti del prodotto sulla base dei test, dall'operatività e dall'osservazione durante la sua esecuzione, in funzione degli obiettivi stabiliti. Le metriche esterne sono scelte in base alle caratteristiche che il prodotto finale dovrà dimostrare durante la sua esecuzione.

6.4 Qualità in uso

La qualità in uso rappresenta il punto di vista dell'utente sul prodotto. Il livello di qualità in uso è raggiunto quando sono state conseguite sia la qualità esterna che quella interna.

6.4.1 Metriche per la qualità in uso

La qualità in uso permette di abilitare specificati utenti ad ottenere dati obiettivi con efficacia, produttività, sicurezza e soddisfazione:

- **Efficacia:** capacità del prodotto di consentire agli utenti di raggiungere gli obiettivi specificati con accuratezza e completezza;
- **Produttività:** capacità di consentire agli utenti di adoperare un'appropriata quantità di risorse rispetto all'efficacia ottenuta in uno dato contesto d'uso;
- **Soddisfazione:** capacità del prodotto di corrispondere alle aspettative degli utenti;
- **Sicurezza:** capacità del prodotto di raggiungere accettabili livelli di rischio di danni a persone, software, apparecchiature tecniche e all'ambiente d'uso.

7 Metriche della qualità

7.1 Metriche per la qualità di prodotto

La presente sezione espone le metriche selezionate dal gruppo *SWEmming Pool* per misurare il raggiungimento degli obiettivi di qualità del prodotto.

7.1.1 MPD01: Indice di Gulpease

Indice che riporta il grado di leggibilità di un testo redatto in lingua italiana. La formula adottata è la seguente:

$$GULP = 89 + \frac{300 * (totale\ frasi) - 10 * (totale\ lettere)}{(totale\ parole)}$$

L'indice così calcolato può assumere valori tra 0 e 100:

- **GULP $<$ 80:** indica una leggibilità difficile per un utente con licenza elementare;
- **GULP $<$ 60:** indica una leggibilità difficile per un utente con licenza media;
- **GULP $<$ 40:** indica una leggibilità difficile per un utente con licenza superiore.

Per facilitare il calcolo di questo indice, il gruppo si è appoggiato al sito https://farfalla-project.org/readability_static/.

7.1.2 MPD02: Copertura requisiti obbligatori

Indice che misura in ogni istante la percentuale di requisiti obbligatori soddisfatti. La formula adottata è la seguente:

$$CROB = \frac{ROBC}{ROB} * 100$$

dove:

- **ROBC:** indica il numero di requisiti obbligatori coperti dall'implementazione;
- **ROB:** indica il numero complessivo di requisiti obbligatori.

7.1.3 MPD03: Copertura requisiti desiderabili

Indice che misura in ogni istante la percentuale di requisiti desiderabili soddisfatti. La formula adottata è la seguente:

$$CRD = \frac{RDC}{RD} * 100$$

dove:

- **RDC:** indica il numero di requisiti desiderabili coperti dall'implementazione;
- **RD:** indica il numero complessivo di requisiti opzionali.

7.1.4 MPD04: Copertura requisiti opzionali

Indice che misura in ogni istante la percentuale di requisiti opzionali soddisfatti. La formula adottata è la seguente:

$$CROP = \frac{ROPC}{ROP} * 100$$

dove:

- **ROPC:** indica il numero di requisiti opzionali accettati coperti dall'implementazione;
- **ROP:** indica il numero complessivo di requisiti opzionali accettati.

7.1.5 MPD05: Facilità di utilizzo

Numero di click necessari con cui l'utente raggiunge la funzionalità cercata.

7.1.6 MPD06: Versioni browser supportate

Percentuale di versioni di browser supportate dal prodotto. Calcolabile con la seguente formula:

$$BS = \frac{Bsup}{Btot} * 100$$

dove:

- **Bsup** indica il numero di browser in cui il prodotto può essere eseguito;
- **Btot** indica il numero complessivo di browser presi in considerazione.

Dove in **Btot** i browser presi in considerazione sono i seguenti:

- Chrome;
- Firefox;
- Brave;
- Opera.

7.1.7 MPD07: Solidity Statement Coverage

Viene utilizzato per calcolare il numero di istruzioni nel codice sorgente che sono state eseguite.

Lo scopo dello Statement Coverage è quello di coprire tutti i possibili percorsi, righe e istruzioni nel codice sorgente.

Il valore è stato calcolato per mezzo di solidity-coverage.

7.1.8 MPD08: Solidity Branch Coverage

Lo scopo del Branch Coverage è garantire che ogni condizione decisionale venga eseguita almeno una volta.

Il valore è stato calcolato per mezzo di solidity-coverage.

7.1.9 MPD09: Solidity Function Coverage

Indica la misura in cui le funzioni presenti nel codice sorgente sono coperte durante il test.

Tutte le funzioni presenti nel codice sorgente vengono testate durante l'esecuzione del test.

Il valore è stato calcolato per mezzo di solidity-coverage.

7.1.10 MPD10: Solidity Line Coverage

Il Line Coverage di un programma è il numero di righe eseguite diviso per il numero totale di righe.

Vengono considerate solo le righe che contengono istruzioni eseguibili. Il valore è stato calcolato per mezzo di solidity-coverage.

7.1.11 MPD11: Frontend Statement Coverage

Viene utilizzato per calcolare il numero di istruzioni nel codice sorgente che sono state eseguite.

Lo scopo dello Statement Coverage è quello di coprire tutti i possibili percorsi, righe e istruzioni nel codice sorgente.

7.1.12 MPD12: Frontend Branch Coverage

Lo scopo del Branch Coverage è garantire che ogni condizione decisionale venga eseguita almeno una volta.

7.1.13 MPD13: Frontend Function Coverage

Indica la misura in cui le funzioni presenti nel codice sorgente sono coperte durante il test.

Tutte le funzioni presenti nel codice sorgente vengono testate durante l'esecuzione del test.

7.1.14 MPD14: Frontend Line Coverage

Il Line Coverage di un programma è il numero di righe eseguite diviso per il numero totale di righe.

Vengono considerate solo le righe che contengono istruzioni eseguibili.

7.1.15 Obiettivi qualità di prodotto

Obiettivo	Descrizione	Metriche
Monitoraggio documentazione		
Leggibilità documenti	I documenti devono essere comprensibili agli utenti.	MPD01
Correttezza linguistica	Tutti gli errori grammaticali devono essere corretti.	MPD02
Monitoraggio software		
Funzionalità	Capacità del prodotto di offrire tutte le funzioni individuate nell'*Analisi dei Requisiti*, perseguendo accuratezza e adeguatezza.	MPD03 MPD04 MPD05
Usabilità	Capacità di essere comprensibile in modo da rendere piacevole l'esperienza dell'utente. Le funzionalità devono essere compatibili con le aspettative.	MPD06
Portabilità	Capacità di poter funzionare in diversi ambienti di esecuzione. Gli obiettivi da perseguire sono adattabilità e sostituibilità.	MPD07
Copertura test	Il codice prodotto dovrà essere verificato nella sua interezza al fine di garantire la corretta implementazione dei requisiti individuati.	MPD08 MPD09 MPD10 MPD11 MPD12 MPD13 MPD14 MPD15

Tabella 1: Obiettivi qualità di prodotto

7.2 La qualità di processo

La presente sezione espone le metriche selezionate dal gruppo *SWEmming Pool* per misurare il raggiungimento degli obiettivi di qualità del processo.

Parametri per comprendere le metriche scelte:

- **Budget At Completion (BAC)**: È il budget stimato in preventivo per il completamento del progetto;
- **Actual Calendar Days (ACD)**: È il numero effettivo di giorni impiegati in un determinato periodo;
- **Planned Calendar Days (PCD)**: È la durata preventivata di un determinato periodo;
- **Number of Requirements Added (NRA)**: Numero di requisiti aggiunti in un determinato periodo;
- **Number of Requirements Changed (NRC)**: Numero di requisiti modificati in un determinato periodo;
- **Number of Requirements Removed (NRR)**: Numero di requisiti rimossi in un determinato periodo;
- **Total Number of Initial Requirements (TNIR)**: Numero totale dei requisiti all'inizio di un determinato periodo.

7.2.1 MPC01: Earned Value (EV)

È il valore del lavoro effettuato fino ad un determinato periodo.

7.2.2 MPC02: Actual Cost (AC)

È il costo effettivamente sostenuto fino ad un determinato periodo.

7.2.3 MPC03: Planned Value (PV)

È il costo stimato per il completamento di un determinato periodo.

7.2.4 MPC04: Cost Variance (CV)

Metrica che indica, in percentuale, se il gruppo è regolare rispetto al budget prestabilito. La formula adottata è la seguente:

$$CV = (\frac{EV}{AC} - 1)100$$

Se il valore risulta negativo si è fuori budget.

7.2.5 MPC05: Schedule Variance (SV)

Metrica che indica, in percentuale, se il gruppo è regolare rispetto i limiti temporali imposti. La formula adottata è la seguente:

$$SV = (1 - \frac{ACD}{PCD}) * 100$$

Se il valore risulta negativo si è indietro rispetto i tempi preventivati.

7.2.6 MPC06: Estimated At Completion (EAC)

Revisione del valore stimato per la realizzazione del progetto in un determinato periodo, ossia il BAC rivisto allo stato corrente del progetto. La formula adottata è la seguente:

$$EAC = AC + ETC$$

7.2.7 MPC07: Estimate To Complete (ETC)

Valore stimato del costo del lavoro rimanente in un determinato periodo. La formula adottata è la seguente:

$$ETC = BAC - EV$$

7.2.8 MPC08: Requirements Stability Index (RSI)

Indice che misura la variazione dei requisiti nel tempo. La formula adottata è la seguente:

$$RSI = (1 - \frac{NRA + NRC + NRR}{TNIR})100$$

7.2.9 MPC09: Passed Tests

Indica la percentuale dei test che sono stati superati fino ad un determinato periodo. La formula adottata è la seguente:

$$Passed\ Tests = \frac{Number\ of\ Passed\ Tests}{Tot.\ Number\ of\ Tests}100$$

7.2.10 MPC10: Metrics Satisfied

Indice che rappresenta la percentuale di metriche di qualità soddisfatte in un determinato periodo. La formula adottata è la seguente:

$$Metrics\ Satisfied = \frac{Number\ of\ Metrics\ Satisfied}{Tot.\ Number\ of\ Metrics} * 100$$

7.2.11 MPC11: Risks Found

Indica il numero di rischi incontrati in un determinato periodo.

7.2.12 Obiettivi qualità di processo

Obiettivo	Descrizione	Metriche
Processi primari		
Fornitura	Processo che consiste nel decidere procedure e risorse adatte a soddisfare le necessità del cliente.	MPC01 MPC02 MPC03 MPC04 MPC05 MPC06 MPC07
Sviluppo	Processo che ha lo scopo di sviluppare un prodotto software che indirizzi le esigenze del cliente.	MPC08
Processi di supporto		
Verifica	Processo che ha lo scopo di confermare che ciascun servizio realizzato soddisfi i requisiti specificati.	MPC09
Gestione della qualità	Processo che ha lo scopo di assicurare che il prodotto e i servizi offerti siano conformi agli standard definiti.	MPC10
Processi organizzativi		
Gestione organizzativa	Processo che si occupa di esporre le modalità di coordinamento del gruppo.	MPC11

Tabella 2: Obiettivi qualità di processo