

SEVEN CLICKERS

You Wish We Click

7clickersgroup@gmail.com

Norme di Progetto

Versione	1.0.7
Stato	Verificato
Uso	Interno
Approvazione_g	Rino Sincic
Redazione	Mirko Stella Giacomo Mason Gabriele Mantoan Marco Brigo
Verifica_g	Giacomo Mason Gabriele Mantoan
Distribuzione	<i>Seven Clickers</i> Prof. Vardanega Tullio Prof. Cardin Riccardo

Descrizione

Questo documento contiene le *Norme di Progetto* seguite dal gruppo Seven Clickers per il progetto ShowRoom3D

Registro delle modifiche

Vers.	Data	Autore	Ruolo	Descrizione
1.0.7	05-05-23	Elena Pandolfo Giacomo Mason	Progettista Verificatore	Modifica sezione Branching e verifica
1.0.6	24-04-23	Gabriele Mantoan Giacomo Mason	Verificatori	Stesura sezione Verifica del codice e verifica
1.0.5	24-04-23	Tommaso Allegretti Giacomo Mason	Progettista Verificatore	Stesura paragrafo sprint e verifica
1.0.4	18-04-23	Elena Pandolfo Gabriele Mantoan	Amministratore Verificatore	Stesura sezione Metodo di lavoro e verifica
1.0.3	17-04-23	Gabriele Mantoan Giacomo Mason	Amministratore Verificatore	Aggiunta sezione Codifica
1.0.2	06-04-23	Elena Pandolfo Gabriele Mantoan	Amministratore Verificatore	Modifica a sezione Versionamento e verifica
1.0.1	31-03-23	Elena Pandolfo Gabriele Mantoan	Amministratore Verificatore	Aggiunta sezione Riferimenti e verifica
1.0.0	21-03-23	Rino Sincic	Responsabile	Approvazione _g documento
0.4.0	19-03-23	Gabriele Mantoan Giacomo Mason	Verificatori	Verifica _g
0.3.2	19-03-23	Marco Brigo	Progettista	Aggiornamento metriche
0.3.1	18-03-23	Elena Pandolfo	Amministratore	Modificata la sezione Branching e la norma sulle date
0.3.0	01-03-23	Mirko Stella	Verificatore	Verifica _g documento
0.2.6	06-02-23	Marco Brigo	Verificatore	Aggiunta sezione Accertamento della qualità e metriche di qualità
0.2.5	05-02-23	Mirko Stella	Programmatore	Aggiunte sezioni <i>Piano di Qualifica</i> , <i>Glossario</i> , <i>Norme di Progetto</i> in Documentazione
0.2.4	01-02-23	Mirko Stella	Programmatore	Aggiunta sezione <i>Studio di Fattibilità</i> , modificata sezione Verifica _g della documentazione
0.2.3	31-01-23	Mirko Stella	Programmatore	Aggiunte sezioni <i>Lettera di Presentazione</i> , <i>Diario di Bordo</i> , modificata sezione <i>Glossario</i> , spostata sezione <i>Verbale</i> da Convenzioni generali a Tipi di documento
0.2.2	31-01-23	Elena Pandolfo	Verificatore	Aggiunte sezioni Validazione _g e Gestione della configurazione

0.2.1	22-01-23	Elena Pandolfo	Verificatore	Aggiunte sezioni Fornitura e Sviluppo e sistemato l'indice
0.2.0	06-01-23	Elena Pandolfo Mirko Stella	Verificatore e Responsabile	Verifica _g Documento
0.1.3	06-01-23	Marco Brigo	Verificatore	Inserita sezione standard ISO/IEC 9126
0.1.2	10-12-22	Mirko Stella	Verificatore	Modifica a sezione Milestone _g , Projects Board, Issue _g Tracking System
0.1.1	03-12-22	Marco Brigo	Analista	Prima stesura sezione Approvazione _g e Strumenti per la stesura, aggiornate norme su Branching, aggiunta sezione su Pull Request, Milestone _g , Projects Board, aggiornata sezione Issue _g Tracking System, tolta sezione Jira _g
0.1.0	22-11-22	Giacomo Mason Tommaso Allegretti	Verificatori	Verifica _g documento
0.0.3	16-11-22	Gabriele Mantoan	Verificatore	Modificate norme riguardo verifica _g e vita delle issue _g ; aggiunte norme riguardo Jira _g e indici di <i>Glossario</i>
0.0.2	13-11-22	Gabriele Mantoan	Verificatore	Aggiunte norme riguardo Issue _g tracking system, ruoli e nomi dei branch liberi
0.0.1	06-11-22	Mirko Stella Giacomo Mason	Analista Verificatore	Creazione documento

Indice

1	Introduzione	4
1.1	Scopo del documento	4
1.2	Scopo del prodotto _g	4
1.3	Glossario	4
1.4	Riferimenti	4
1.4.1	Riferimenti normativi	4
1.4.2	Riferimenti informativi	4
2	Processi Primari	5
2.1	Fornitura	5
2.1.1	Scopo	5
2.1.2	Descrizione	5
2.1.3	Rapporti con il proponente _g	5
2.1.4	Metriche	5
2.2	Sviluppo	5
2.2.1	Scopo	5
2.2.2	Descrizione	6
2.2.3	<i>Analisi dei Requisiti</i>	6
2.2.3.1	Struttura del documento	6
2.2.3.2	Casi d'uso	6
2.2.3.3	Requisiti	7
2.2.4	Progettazione	7
2.2.4.1	Requirments and Tecnology Baseline	7
2.2.5	Codifica	7
2.2.5.1	Stile di codifica	8
2.2.5.2	Norme linguaggi adottati	8
2.2.5.3	Norme React.js	9
2.2.5.4	Organizzazione Codice	9
2.2.5.5	Strumenti di sviluppo	10
2.2.5.6	Ambiente di lavoro	10
2.2.5.7	Configurazione Ambiente di lavoro	10
2.2.6	Metriche	10
3	Processi di Supporto	11
3.1	Documentazione	11
3.1.1	Scopo	11
3.1.2	Descrizione	11
3.1.3	Ciclo di vita _g di un documento	11
3.1.4	Struttura generale	11
3.1.5	Convenzioni	12
3.1.5.1	Date	12
3.1.5.2	Nomi di persona	12
3.1.5.3	Elenchi puntati	12
3.1.5.4	Stile del testo	12
3.1.5.5	Immagini	13
3.1.5.6	Tabelle	13
3.1.5.7	<i>Glossario</i>	13
3.1.6	Strumenti per la stesura	13
3.1.7	Documentazione interna	14
3.1.7.1	<i>Verbale</i>	14
3.1.7.2	<i>Studio di Fattibilità</i>	14
3.1.7.3	<i>Norme di Progetto</i>	14

3.1.8	Documentazione esterna	15
3.1.8.1	<i>Lettera di Presentazione</i>	15
3.1.8.2	<i>Diario di Bordo</i>	18
3.1.8.3	<i>Piano di Progetto</i>	19
3.1.8.4	<i>Piano di Qualifica</i>	20
3.1.8.5	<i>Glossario</i>	20
3.1.9	Metriche	20
3.2	Gestione della configurazione	21
3.2.1	Scopo	21
3.2.2	Descrizione	21
3.2.3	Versionamento	21
3.2.3.1	Sistemi software utilizzati	21
3.3	Accertamento della qualità	21
3.3.1	Scopo	21
3.3.2	Descrizione	21
3.3.3	Attività per il controllo di qualità	21
3.3.3.1	Attuazione del controllo di qualità	22
3.3.4	Denominazione requisiti nel <i>Piano di Qualifica</i>	22
3.3.5	Denominazione degli obiettivi di qualità nel <i>Piano di Qualifica</i>	22
3.3.6	Metriche	22
3.4	Verifica _g	23
3.4.1	Scopo	23
3.4.2	Descrizione	23
3.4.3	Verifica _g della documentazione	23
3.4.4	Verifica del codice	24
3.4.4.1	Codice identificativo	25
3.4.4.2	Continuous Integration	25
3.4.4.3	Procedimento di verifica del software	25
3.4.4.4	Strumenti	25
3.4.5	Metriche	26
3.5	Validazione _g	26
3.5.1	Scopo	26
4	Processi Organizzativi	27
4.1	Gestione Organizzativa	27
4.1.1	Scopo	27
4.1.2	Descrizione	27
4.1.3	Pianificazione	27
4.1.3.1	Ruoli	27
4.1.3.2	Sprint	28
4.1.3.3	Metodo di lavoro	28
4.1.4	Gestione delle comunicazioni	29
4.1.4.1	Comunicazioni Interne	29
4.1.4.2	Comunicazioni Esterne	29
4.1.5	Gestione delle riunioni	29
4.1.5.1	Riunioni Interne	29
4.1.5.2	Riunioni Esterne	30
4.1.6	Metriche	30
4.2	Gestione Infrastrutture	30
4.2.1	Scopo	30
4.2.2	Descrizione	30
4.2.3	GitHub _g	30
4.2.3.1	Repository _g	30
4.2.3.2	Branching	30

4.2.3.3	Commits	31
4.2.3.4	Pull Requests	32
4.2.3.5	Milestone _g	32
4.2.3.6	Projects Board	32
4.2.3.7	Issue _g Tracking System	33
4.2.4	Discord _g	34
4.2.5	Metriche	34
5	Standard di qualità ISO/IEC 9126	35
5.1	Qualità esterne	35
5.2	Qualità interne	35
5.3	Modello di qualità	35
5.3.1	Funzionalità	35
5.3.2	Affidabilità	36
5.3.3	Efficienza	36
5.3.4	Usabilità	36
5.3.5	Manutenibilità	36
5.3.6	Portabilità	37
5.4	Qualità in uso	37
6	Standard di qualità ISO/IEC 12207:1995	38
6.1	Processi primari	38
6.1.1	Acquisizione	38
6.1.2	Fornitura	38
6.1.3	Sviluppo	39
6.1.4	Gestione operativa	39
6.1.5	Manutenzione	39
6.2	Processi di supporto	40
6.2.1	Documentazione	40
6.2.2	Gestione della configurazione	40
6.2.3	Accertamento della qualità	40
6.2.4	Verifica _g	40
6.2.5	Validazione _g	41
6.2.6	Revisione congiunta	41
6.2.7	Audit	41
6.2.8	Risoluzione dei problemi	41
6.3	Processi organizzativi	41
6.3.1	Gestione organizzativa	41
6.3.2	Gestione delle infrastrutture	42
6.3.3	Miglioramento del processo _g	42
6.3.4	Formazione del personale	42
7	Metriche di qualità	43
7.1	Metriche per la qualità di processo _g	43
7.1.1	MPC01: Planned Value (PV)	43
7.1.2	MPC02: Actual Cost (AC)	43
7.1.3	MPC03: Estimated at Completion (EAC)	43
7.1.4	MPC04: Earned Value (EV)	43
7.1.5	MPC05: Estimated to Complete (ETC)	44
7.1.6	MPC06: Cost Variance (CV)	44
7.1.7	MPC07: Schedule Variance (SV)	44
7.1.8	MPC08: Budget Variance (BV)	44
7.1.9	MPC09: Requirements Stability Index (RSI)	44
7.1.10	MPC10: Indice di Gulpease	45

7.1.11	MPC11: Metriche soddisfatte	45
7.1.12	MPC12: Code coverage	45
7.1.13	MPC13: Rischi non previsti	45
7.2	Metriche per la qualità di prodotto _g	45
7.2.1	MPD01: Percentuale requisiti soddisfatti	45
7.2.2	MPD02: Densità di fallimenti durante l'esecuzione	46
7.2.3	MPD03: Tempo medio di risposta	46
7.2.4	MPD04: Tempo di caricamento	46
7.2.5	MPD05: Facilità di apprendimento	46
7.2.6	MPD06: Complessità ciclomatica	46
7.2.7	MPD07: Densità dei commenti	47
7.2.8	MPD08: Browser supportati	47

Elenco delle figure

1	Esempio di <i>Lettera di Presentazione</i>	17
---	--	----

1 Introduzione

1.1 Scopo del documento

Norme di Progetto è il documento che definisce le regole e gli standard che devono essere seguiti durante il ciclo di vita_g del prodotto_g.

1.2 Scopo del prodotto_g

Il prodotto_g in questione nasce dalla necessità dell'azienda SanMarco Informatica di fornire una soluzione agli sprechi derivati dall'adozione di uno ShowRoom tradizionale proponendo uno ShowRoom3D che sia ugualmente o ancora più immersivo.

1.3 Glossario

In questo documento sono state segnate con il pedice "g" tutte le parole che, secondo noi, necessitano di una loro definizione più accurata nel documento di *Glossario v1.0.0*.

1.4 Riferimenti

1.4.1 Riferimenti normativi

- Regolamento del Progetto Didattico: <https://www.math.unipd.it/~tullio/IS-1/2022/Dispense/PD02.pdf>
- Capitolato di appalto C6 - ShowRoom3D: <https://www.math.unipd.it/~tullio/IS-1/2022/Progetto/C6.pdf>

1.4.2 Riferimenti informativi

- Materiale didattico Ingegneria del Software - T02 Processi di ciclo di vita_g: <https://www.math.unipd.it/~tullio/IS-1/2022/Dispense/T02.pdf>
- Materiale didattico Ingegneria del Software - T03 Il ciclo di vita del SW: <https://www.math.unipd.it/~tullio/IS-1/2022/Dispense/T03.pdf>
- Materiale didattico Ingegneria del Software - T04 Gestione di progetto: <https://www.math.unipd.it/~tullio/IS-1/2022/Dispense/T04.pdf>
- Materiale didattico Ingegneria del Software - T06 Analisi dei requisiti: <https://www.math.unipd.it/~tullio/IS-1/2022/Dispense/T06.pdf>
- Materiale didattico Ingegneria del Software - T010 Verifica e validazione: introduzione <https://www.math.unipd.it/~tullio/IS-1/2022/Dispense/T010.pdf>
- Lo standard ISO/IEC 12207:1995 : https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf

2 Processi Primari

I processi primari devono essere seguiti durante il ciclo di vita_g del software. Nel nostro il ciclo di vita_g del software non comprende l'installazione e la manutenzione ma termina con lo sviluppo. Il progetto in questione è da considerarsi un progetto didattico.

2.1 Fornitura

2.1.1 Scopo

Lo scopo del processo_g di fornitura è quello di stabilire quali sono le attività, le risorse e i compiti necessari allo svolgimento del progetto.

2.1.2 Descrizione

Nella seguente sezione sono riportate le norme che il gruppo si impegna a rispettare al fine di creare e mantenere un rapporto attivo e vantaggioso con il proponente_g SanMarco Informatica.

2.1.3 Rapporti con il proponente_g

Il gruppo si manterrà in contatto con il proponente_g regolarmente, organizzando incontri su Google Meet o comunicando in modo asincrono con email o Google Chat. Di seguito un elenco dei principali temi che possono essere motivo di confronto con il proponente_g:

- Chiarimento di eventuali dubbi o incomprensioni;
- Riscontro sul lavoro svolto;
- Definizione dei requisiti obbligatori e opzionali;
- Opinioni riguardanti tecnologie da utilizzare;
- Proposte da parte del gruppo di soluzioni alternative.

2.1.4 Metriche

Per perseguire la qualità nel processo_g di Fornitura si è deciso di adottare le seguenti metriche:

- **MPC01: Planned Value (PV);**
- **MPC02: Actual Cost (AC);**
- **MPC03: Estimated at Completion (EAC);**
- **MPC04: Earned Value (EV);**
- **MPC05: Estimated to Complete (ETC);**
- **MPC06: Cost Variance (CV);**
- **MPC07: Schedule Variance (SV);**
- **MPC08: Budget Variance (BV).**

2.2 Sviluppo

2.2.1 Scopo

Lo scopo del processo_g di sviluppo è quello di stabilire le attività necessarie allo sviluppo del prodotto_g software.

2.2.2 Descrizione

Nelle seguenti sezioni verranno descritte più nel dettaglio le principali attività, relative al processo_g di sviluppo:

- *Analisi dei Requisiti*;
- Progettazione;

2.2.3 *Analisi dei Requisiti*

L'*Analisi dei Requisiti* è un'attività fondamentale che precede la progettazione. Si concretizza nel documento *Analisi dei Requisiti*. Lo scopo di quest'attività è:

- Definire le funzionalità che il prodotto_g andrà ad offrire;
- Porre le basi per la fase di progettazione del software;
- Fare una stima della mole di lavoro;
- Facilitare la verifica_g.

2.2.3.1 Struttura del documento

Il documento è strutturato nel modo seguente:

- **Introduzione:** contiene una breve descrizione del documento e specifica gli attori dei casi d'uso;
- **Casi d'uso:** vengono specificati i casi d'uso individuati;
- **Requisiti:** sono classificati i requisiti che il prodotto_g finale dovrà soddisfare.

2.2.3.2 Casi d'uso

I casi d'uso sono identificati seguendo la seguente struttura:

UC [Numero caso d'uso].[Numero sottocaso d'uso]-[Titolo caso d'uso]

Vengono poi indicati:

- **Diagramma UML_g:** viene mostrato il diagramma solo per i casi più complessi, in cui è ritenuto necessario per una maggiore comprensione. Per i casi d'uso facoltativi il diagramma è colorato di azzurro (non standard UML_g) per una maggiore differenziazione;
- **Attore_g primario:** utente esterno al sistema che svolge il caso d'uso;
- **Descrizione:** breve descrizione del caso d'uso;
- **Precondizioni:** indica la condizione del sistema prima del verificarsi del caso d'uso;
- **Postcondizioni:** indica la condizione del sistema dopo che si è verificato il caso d'uso;
- **Scenario principale:** descrive l'interazione tra l'attore_g primario e il sistema;
- **Estensioni:** casi d'uso alternativi che possono verificarsi al posto di quello a cui sono collegati.

2.2.3.3 Requisiti

I requisiti sono classificati secondo la seguente struttura:

R.[Tipologia][Numero seriale]

con:

- **Tipologia:**
 - **F:** funzionale;
 - **Q:** qualitativo;
 - **D:** di dominio;
 - **P:** prestazionale.
- **Descrizione:** breve descrizione del requisito;
- **Classificazione:** un requisito può essere facoltativo o obbligatorio;
- **Fonti:** possono essere:
 - Casi d'uso;
 - Capitolato;
 - Decisione interna.

2.2.4 Progettazione

L'attività di progettazione è assegnata ai progettisti. Seguendo quanto indicato nell'*Analisi dei Requisiti*, l'obiettivo è quello di progettare l'architettura del sistema, prima con la creazione di un Proof of Concept_g per la Requirements and Tecnology Baseline e poi andando nel dettaglio per la Product Baseline.

2.2.4.1 Requirements and Tecnology Baseline

In questa fase vengono fissati i requisiti che il gruppo si impegna a soddisfare, in accordo con il proponente_g; si studiano le tecnologie, i framework_g e le librerie utili alla realizzazione del prodotto_g finale e si crea di conseguenza un Proof of Concept_g. I materiali da mostrare sono:

- Tecnologie, framework_g, librerie utilizzate, motivando la scelta;
- Proof of Concept_g;
- *Analisi dei Requisiti*.

La documentazione da mostrare:

- *Piano di Progetto*;
- *Piano di Qualifica*;
- *Norme di Progetto*;
- *Verbali* interni ed esterni.

2.2.5 Codifica

L'attività di codifica è assegnata ai programmatori. Seguendo quanto indicato nel diagramma delle classi, l'obiettivo è quello di implementare ciò che è stato precedentemente progettato. La codifica dovrà rispettare delle norme per motivi legati alla leggibilità, manutenzione, verifica e validazione del codice.

2.2.5.1 Stile di codifica

- **Indentazione:** I blocchi di codice innestati dovranno avere un'indentazione di quattro spazi;
- **Parentesi graffe:** La parentesi aperta dovrà essere inserita nella stessa riga di dichiarazione del costrutto, separata da uno spazio, mentre la parentesi chiusa dovrà essere inserita con la giusta indentazione alla riga immediatamente successiva all'ultima riga di codice del costrutto;
- **Metodi:** il nome dei metodi dovrà iniziare con lettera minuscola e, se composto da più parole, le successive dovranno iniziare con lettera maiuscola. E preferibile mantenere metodi brevi, con poche righe di codice;
- **Classi:** il nome delle classi dovrà sempre iniziare con la lettera maiuscola e, come per i metodi, se composto da più parole, le successive dovranno iniziare con la lettera maiuscola;
- **Variabili:** il nome delle variabili deve sempre essere scritto in minuscolo e in inglese e la loro dichiarazione possibilmente all'inizio della funzione o script. Se il nome è composto da più parole, la seconda dovrà iniziare con la lettera maiuscola;
- **Costanti:** il nome deve essere sempre scritto in maiuscolo e in inglese. Se il nome è composto da più parole, queste dovranno essere separate dal carattere ' _';
- **Univocità dei nomi:** tutti i costrutti dovranno avere nomi univoci e significativi;
- **Commenti:** i commenti dovranno essere inseriti prima dell'inizio del costrutto e presentati in lingua italiana, inoltre sono obbligatori in parti di codice che non sono immediatamente comprensibili. Ogni volta che viene aggiornato un metodo va verificata la validità del commento;
- **File:** dovranno avere un nome che inizia per lettera maiuscola che ne specifichi il contenuto.

2.2.5.2 Norme linguaggi adottati

In questo paragrafo andremo a definire le norme che il gruppo ha deciso di adottare per ogni specifico linguaggio utilizzato nel progetto.

HTML è oggi utilizzato per creazione della struttura logica di una pagina web, abbiamo deciso di adottare le seguenti semplici normative:

- Utilizzare tag lowercase;
- Chiudere tutti i tag;
- Utilizzare attributi dei tag in lowercase;
- Utilizzare le virgolette per i valori degli attributi.

CSS linguaggio che gestisce il design e la presentazione delle pagine web. Il gruppo ha concordato le seguenti normative:

- Un solo selettore per linea in set di regole con più selettori;
- Un singolo spazio prima della parentesi graffa di apertura di un set di regole;
- Una dichiarazione per linea di un blocco dichiarazioni;
- Un livello di rientro per ogni dichiarazione;
- Un singolo spazio dopo i due-punti di una dichiarazione;
- Includete sempre un "punto e virgola" alla fine dell'ultima dichiarazione in un blocco dichiarazioni;
- Mettete la parentesi graffa di chiusura di un set di regole, nella stessa colonna del primo carattere del set di regole;
- Separate ogni set di regole con una linea vuota.

JavaScript è un linguaggio di programmazione multi paradigma orientato agli eventi, utilizzato sia nella programmazione lato client web che lato server.

Il gruppo ha deciso di adottare lo stile di codifica presente al seguente link:

[https://github.com/airbnb/javascript\(21-04-2023\)](https://github.com/airbnb/javascript(21-04-2023))

2.2.5.3 Norme React.js

Il gruppo ha concordato le seguenti normative:

- Il nome del componente deve essere uguale al nome del modulo con la prima lettera maiuscola;
- Scrittura di informazioni dentro ad un tag;
 - Se l'informazione può essere contenuta in una linea di codice allora si mantiene sulla stessa ; linea
 - Se l'informazione è più lunga di una linea di codice allora bisogna indentare le informazioni.
- Utilizzare sempre camelCase per i nomi delle prop;
- Utilizzare i doppi apici tranne che per i valori nelle informazioni di stile;
- Utilizzare una self-close se i tag non hanno figli;
- Utilizzare className per assegnare classi CSS agli elementi;
- Non comprimere troppo il codice ma utilizzare degli whitespace adeguati.

2.2.5.4 Organizzazione Codice

Il codice deve essere organizzato secondo la struttura chiama *Feature Folders* raccomandata dalle linee guida di Redux.

In seguito riportiamo la struttura che il gruppo ha deciso di utilizzare per il progetto:

- **\public**
 - **index.html**
- **\src**
 - **index.jsx** Punto di entrata che effettua il render dei vari componenti React
 - **\app**
 - * **store.js** store setup
 - * **App.jsx** root dei componenti React
 - **\common** contiene componenti generici e utility
 - **\features** contiene tutte le Feature folders
 - * **\feature** Singola feature
 - **featureSlice.js** contiene la logica del reducer e le azioni associate
 - **Feature.jsx** componente React
 - **Feature.css**
 - **\assets**
 - * **\models** cartella contenete i vari modelli da utilizzare nel applicativo
 - * **\img** cartella contenete icone ed immagini

2.2.5.5 Strumenti di sviluppo

Di seguito riportiamo gli strumenti utilizzati durante il processo di sviluppo:

- **React.js:** Libreria open-source per la creazione di interfacce utente in JavaScript;
- **Three.js:** Libreria JavaScript utilizzata per la realizzazione di contenuti 3D per il web, utilizza le API WebGL;
- **Node.js:** è un runtime system open source multiplatforma orientato agli eventi per l'esecuzione di codice JavaScript;
- **Redux:** è una architettura per scrivere applicazioni web basate su React, fornisce un modo semplice per centralizzare lo stato e la logica di un'applicazione web;
- **React three fiber:** è un React renderer per three.js , permette di costruire componenti riutilizzabili e indipendenti;

2.2.5.6 Ambiente di lavoro

Di seguito riportiamo gli strumenti che hanno definito il nostro ambiente di lavoro durante il processo di sviluppo.

- **Visual Studio Code:** Code editor scelto per sviluppare tutto il codice presente nel nostro progetto. Compatibile con Windows, MacOS e Linux;

2.2.5.7 Configurazione Ambiente di lavoro

La configurazione dell'ambiente di lavoro viene fatta da un componente del gruppo, che carica la cartella di lavoro configurata nel repo. Successivamente gli altri componenti devono eseguire le seguenti istruzioni per mettersi in condizioni di lavorare. Per prima cosa ogni componente del gruppo deve installare Node.js e NPM attraverso package manager oppure al seguente link:

[DownloadNode.js](#)

Facendo riferimento alla *v18.16.0* o successive. Per verificare che l'installazione sia andata a buon fine aprire il terminale ed utilizzare il comando *node -v* e *npm -v*.

Successivamente si deve entrare nella cartella di lavoro ottenuta dalla repo, aprire il terminale e utilizzare il comando **npm install** che installa tutte le dipendenze di progetto.

2.2.6 Metriche

Per mantenere la qualità nel processo di Sviluppo abbiamo deciso di adottare la seguente metrica.

- **MPC09: Requirements Stability Index.**

3 Processi di Supporto

I processi di supporto supportano quelli primari in modo da renderli più efficienti ed efficaci.

3.1 Documentazione

3.1.1 Scopo

Il processo_g di documentazione ha come scopo quello di fornire un insieme di documenti che descrivono in dettaglio il progetto software, comprese le sue funzionalità, i requisiti, l'architettura, il design, i processi di sviluppo e i risultati. Questa documentazione serve come riferimento per lo sviluppo del software e aiuta a garantire la coerenza e la completezza del progetto. La documentazione del progetto software è un elemento essenziale e deve essere costantemente aggiornata.

3.1.2 Descrizione

Nella seguente sezione sono raggruppate tutte le norme necessarie alla stesura, alla verifica_g e all'approvazione_g della documentazione, in modo tale da mantenere una coerenza nella forma e nella struttura dei documenti prodotti dal gruppo.

3.1.3 Ciclo di vita_g di un documento

- **Creazione:** il documento viene creato su un nuovo branch, utilizzando un template uguale per tutti i documenti. Viene aggiunta l'intestazione e il registro delle modifiche;
- **Stesura:** si procede con la stesura delle varie sezioni, tracciando i cambiamenti nel registro delle modifiche;
- **Verifica_g:** ogni sezione viene verificata dai verificatori (per maggiori specifiche guardare la sezione [3.4.3 Verifica_g della documentazione](#));
- **Approvazione_g:** una volta che tutte le sezioni sono state verificate si procede con l'approvazione_g del documento, che viene effettuata dal responsabile nel seguente modo:
 - Viene aperta una Pull Request di approvazione_g;
 - Il responsabile rilegge il documento: se riscontra ulteriori problematiche, segnala ai verificatori le eventuali modifiche da apportare;
 - Se sono richieste delle modifiche i verificatori si occupano di apportarle;
 - Il responsabile crea una nuova riga e compila i campi nelle colonne corrispondenti del registro delle modifiche inserendo: l'ultima versione secondo le norme di versionamento, la data, il proprio nome, il suo ruolo e la voce "Approvazione_g" nell'ultima colonna.

Per i *Verbali* si effettua questa pratica non appena il file prodotto_g viene verificato, mentre per tutti gli altri documenti prima di una consegna.

3.1.4 Struttura generale

Ogni documento deve presentare le seguenti sezioni nell'ordine in cui vengono presentate:

- **Intestazione:** contiene:
 - Logo compreso di motto;
 - Indirizzo email di gruppo;
 - Titolo;
 - Tabella contenente le informazioni generali:
 - * Versione;

- * Stato;
 - * Uso;
 - * Approvazione_g: indica il responsabile di progetto che ha approvato il documento;
 - * Redazione: elenco dei collaboratori che hanno partecipato alla stesura del documento;
 - * Verifica_g: elenco dei verificatori che hanno verificato il documento;
 - * Distribuzione: elenco delle persone o organizzazioni a cui è destinato il documento.
- Breve descrizione del documento .

- **Registro delle modifiche:** tabella che identifica ogni versione del documento indicandone:
 - Versione;
 - Data;
 - Autore;
 - Ruolo;
 - Descrizione: la descrizione deve essere breve. Nel caso di aggiunte o modifiche si deve indicare il nome della sezione che è stata aggiunta o modificata. I nomi delle sezioni vanno riportati uguali a come sono scritti nell'indice, senza virgolette.
- **Indice:** elenco ordinato dei titoli dei capitoli, ovvero delle varie parti di cui si compone il documento;
- **Elenco delle figure:** sezione che fornisce l'elenco delle immagini presenti nel documento. Se il documento non presenta immagini, questa sezione sarà omessa di conseguenza;
- **Elenco delle tabelle:** sezione che fornisce l'elenco delle tabelle presenti nel documento. Se il documento non presenta tabelle, questa sezione sarà omessa di conseguenza;
- **Contenuto:** varia a seconda del tipo di documento.

3.1.5 Convenzioni

Le convenzioni di seguito riportate vengono applicate a tutti i documenti. Esse rendono i documenti stilati omogenei tra loro contribuendo a rendere il progetto professionale.

3.1.5.1 Date

Le date devono rispettare il seguente formato: **yyyy-mm-dd** All'interno delle tabelle il formato deve essere il seguente: **dd-mm-yy**

3.1.5.2 Nomi di persona

All'interno dei documenti i nomi di persona rispetteranno l'ordine nome seguito dal cognome della persona menzionata.

3.1.5.3 Elenchi puntati

Gli elenchi puntati devono rispettare le seguenti regole:

- Ogni elemento dell'elenco deve iniziare con la lettera maiuscola;
- Ogni elemento dell'elenco deve terminare con ";" ad eccezione dell'ultimo elemento che deve terminare con " . ";
- Dopo i due punti la frase deve iniziare con la lettera minuscola.

3.1.5.4 Stile del testo

- **Grassetto:** stile utilizzato per i titoli delle sezioni e per i primi termini degli elenchi puntati;
- **Corsivo:** viene utilizzato per citare il nome dei documenti, ad esempio *Piano di Progetto*.

3.1.5.5 Immagini

Le immagini sono raccolte nella cartella “images”, e sono inserite sempre con una didascalia descrittiva posizionata sotto l’immagine.

3.1.5.6 Tabelle

Le tabelle sono provviste di didascalia descrittiva posizionata sotto alla tabella. La tabella contenente il registro delle modifiche è l’unica che fa da eccezione a questa regola.

3.1.5.7 Glossario

All’interno della documentazione si possono trovare dei termini che possono risultare ambigui a seconda del contesto, o non conosciuti dagli utilizzatori.

Per ovviare ad incomprensioni si è deciso di stilare un elenco di termini di interesse accompagnati da una descrizione del loro significato.

I termini presenti all’interno dei documenti che necessitano di una descrizione vengono indicati con il pedice ‘g’ come nell’esempio seguente: termine. È quindi possibile consultare il *Glossario* per reperire tale descrizione.

Ogni componente del gruppo all’inserimento di un termine ritenuto ambiguo deve preoccuparsi di aggiornare il *Glossario*.

Per aggiornare il *Glossario* si devono inserire i nuovi termini nel file .tex nella cartella corrispondente all’iniziale del termine situata al percorso *latex_g/esterni/doc_esterna/Glossario/res/sections/alphabet/*. È stato creato uno script che scansiona un documento di interesse per inserire in automatico il pedice sui termini contenuti nel *Glossario*.

Il componente del gruppo che inserisce all’interno del *Glossario* un nuovo termine deve aggiungere nel file .tex il segnaposto %parola% dopo la subsection (senza spazi) che racchiude il termine per permetterne il riconoscimento da parte dello script.

Il *Glossario* ordina i termini in ordine alfabetico in modo da permetterne una facile e veloce ricerca.

3.1.6 Strumenti per la stesura

- **LaTeX_g**: è un linguaggio di marcatura per la preparazione di testi, basato sul programma di composizione tipografica TEX.

Nel branch documentation si possono trovare i file .pdf prodotti e la cartella “latex_g”. La cartella latex_g contiene tre cartelle interne:

- **esterni e interni**: contengono file .tex di documentazione esterna ed interna come ad esempio i *Verbali* o altra documentazione esterna/interna:
 - * La cartella config contiene i file .tex con le parti fisse dei documenti (intestazione, registro delle modifiche, tracciamento dei temi affrontati) che vengono modificati con i dati del documento specifico;
 - * La cartella res/sections contiene i file .tex con il contenuto vero e proprio (sezioni del documento) che viene redatto in maniera libera dal redattore;
 - * Un file col nome del documento pdf con estensione .tex che viene compilato per produrre il file pdf.
- **template**: contiene file .tex di base utilizzati secondo necessità per comporre i documenti:
 - * changelox.tex è il file di template che serve per scrivere il registro delle modifiche;
 - * package.tex è il file che contiene tutti gli usepackage;
 - * titlepage.tex è il file di template che contiene la configurazione della pagina iniziale di ogni documento;
 - * tracking.tex è il file di template che contiene il tracciamento dei temi affrontati nel documento.

3.1.7 Documentazione interna

La documentazione interna comprende tutti i documenti che contengono informazioni utili principalmente per il gruppo, e che vengono quindi consultati di conseguenza.

3.1.7.1 Verbale

Il *Verbale* ha lo scopo di rendicontare ciò che viene detto durante una riunione. Rispetta la struttura generale. In aggiunta presenta:

- **Informazioni Generali** contiene:
 - Luogo;
 - Data;
 - Ora;
 - Partecipanti.
- **Tabella tracciamento temi affrontati:** tabella che riassume i punti salienti della riunione indicandone:
 - Codice: ha il formato **VX Y.Z** dove X indica la tipologia di *Verbale*, Y indica il numero di *Verbale* (incrementale rispetto agli altri verbali); e Z indica il numero dell'argomento trattato (incrementale rispetto agli altri argomenti del *Verbale*);
 - Descrizione: breve descrizione di uno specifico argomento trattato.

3.1.7.2 Studio di Fattibilità

Lo *Studio di Fattibilità* ha lo scopo di valutare i pro ed i contro dei capitolati proposti in modo da scegliere il più vantaggioso al quale candidarsi. La scelta del capitolato viene fatta sulla base di diverse considerazioni. Vengono valutati:

- **Risorse:** in termini di budget, tempo e personale;
- **Previe Conoscenze:** un capitolato potrebbe rivelarsi più o meno difficile da realizzare a seconda delle conoscenze del contesto applicativo;
- **Guadagno:** un capitolato potrebbe venire scartato per scarso guadagno netto al termine del lavoro commissionato (non è il nostro caso ma andrebbe tenuto in considerazione in ambito lavorativo);
- **Aspetti psicologici:** avendo la possibilità di scegliere tra capitolati (fatto che capita raramente in un reale ambito lavorativo) di pari interesse siamo portati a scegliere il contesto applicativo che ci entusiasma maggiormente. Questo aspetto potrebbe avere un impatto positivo sulla produttività del gruppo.

Il documento tiene in considerazione tutti i capitolati proposti per non scartare un capitolato per le sensazioni soggettive dei componenti del gruppo. Non tenendo in considerazione ogni capitolato, infatti, si potrebbe scartare un capitolato di forte interesse senza rendersene conto.

3.1.7.3 Norme di Progetto

Struttura Norme di Progetto

La struttura di *Norme di Progetto* segue la struttura generale. In aggiunta il contenuto del documento si compone di:

- Introduzione;
- Processi primari;

- Processi di supporto;
- Processi organizzativi;
- Standard di qualità ISO/IEC 9126;
- Standard di qualità ISO/IEC 12207:1995.

Descrizione dei campi che compongono il documento:

- **Introduzione:** comprende una breve descrizione dello scopo del documento e del prodotto_g;
- **Processi primari:** comprende la definizione di processi primari. I processi primari inclusi nel nostro progetto sono:
 - Fornitura: comprende la definizione e le regole di fornitura;
 - Sviluppo: comprende la definizione e le regole di sviluppo.
- **Processi di supporto:** comprende la definizione di processi di supporto. I processi di supporto inclusi nel nostro progetto sono:
 - Documentazione: comprende una descrizione dell'attività di documentazione seguito dalle sezioni che descrivono le norme necessarie alla stesura, alla verifica_g e alla validazione_g della documentazione. Si riportano anche le sezioni dedicate ai documenti interni ed esterni nelle quali si descrive la struttura che deve avere ogni documento;
 - Gestione della configurazione: comprende la definizione e le regole di configurazione;
 - Verifica_g: comprende la definizione e le regole di verifica_g;
 - Validazione_g: comprende la definizione e le regole di validazione_g.
- **Processi organizzativi:** comprende la definizione di processi organizzativi. I processi organizzativi inclusi nel nostro progetto sono:
 - Gestione organizzativa: riguarda l'organizzazione all'interno del gruppo ovvero la ripartizione dei compiti e le attività che ogni figura professionale è tenuta a svolgere fino al completamento del progetto;
 - Gestione infrastrutture: cruciale per garantire la disponibilità, l'affidabilità e la sicurezza delle risorse necessarie per lo sviluppo e l'esecuzione del software.
- **Standard di qualità ISO/IEC 9126:** comprende la descrizione dello standard e le sue metriche utilizzate;
- **Standard di qualità ISO/IEC 12207:1995:** comprende la descrizione dello standard e le sue metriche utilizzate.

3.1.8 Documentazione esterna

La documentazione esterna comprende tutti i documenti che interessano anche al proponente_g e al committente_g.

3.1.8.1 Lettera di Presentazione

Una *Lettera di Presentazione* serve a manifestare la volontà da parte del gruppo di prendere un impegno con il committente_g. L'impegno può essere la candidatura per un capitolato di interesse o per una revisione di avanzamento. Dopo aver ricevuto una *Lettera di Presentazione* sta al committente_g la decisione di accettare o rifiutare l'impegno che ne consegue.

Struttura Lettera di Presentazione

Una *Lettera di Presentazione* è formata dai seguenti campi:

- Header;
- Mittente e data;
- Destinatari;
- Contenuto;
- Riferimenti a documenti;
- Conclusioni e saluti;
- Nome, cognome, firma responsabile.

Descrizione dei vari campi:

- **Header:** contiene il logo del gruppo (con slogan) a sinistra ed il logo dell'Università di Padova a destra;
- **Mittente e data:** contiene il nome,email del gruppo seguito dal nome del corso e dalla data di invio della lettera al destinatario;
- **Destinatari:** contiene i destinatari della lettera compresi di titoli e luogo della loro sede;
- **Contenuto:** contiene il contenuto della lettera scritto in modo formale dichiarando lo scopo della lettera;
- **Riferimenti a documenti:** contiene un elenco puntato di tutti i documenti (e loro versioni) a cui si vuole sottoporre l'attenzione dei destinatari.
- **Conclusioni:** contiene le considerazioni finali seguite dai saluti rivolti al destinatario;
- **Nome, cognome, firma responsabile:** contiene il nome, cognome, titolo e firma digitale del responsabile.

Di seguito viene riportata un'immagine che illustra visivamente i campi appena descritti:











HEADER	
<div><div>SEVEN CLICKERS  You Wish We Click</div><div></div></div>	
<div>Gruppo <i>Seven Clickers</i> E-mail: 7clickersgroup@gmail.com Corso di Ingegneria del Software AA 2022/2023 28 Ottobre 2022</div>	MITTENTE E DATA
	<div>DESTINATARI</div> <div>Prof.  Prof.  Università degli Studi di Padova Dipartimento di Matematica Via Trieste, 63 35121 Padova</div>
<div>CONTENUTO</div> <div>Egregio Prof.  Egregio Prof.  Con la presente, il gruppo <i>Seven Clickers</i> intende comunicarVi ufficialmente l'impegno alla realizzazione del prodotto da Voi commissionato, denominato: ShowRoom 3D proposto dall'azienda </div>	
<div>Si allegano i documenti redatti fino ad ora:<ul style="list-style-type: none">• Piano di progetto v1.1.0• Studio di fattibilità v1.0.0• Verbale interno 19-10-2022• Verbale interno 25-10-2022• Verbale interno 26-10-2022• Verbale esterno 25-10-2022• Verbale esterno 26-10-2022</div>	RIFERIMENTI A DOCUMENTI
<div>Come descritto nel <i>Piano di progetto v1.1.0</i>, il gruppo <i>Seven Clickers</i> si impegna a consegnare il prodotto entro 14 Aprile 2022 preventivando un costo complessivo di  Cordiali saluti,</div>	
CONCLUSIONI E SALUTI	<div> <i>Responsabile di Progetto</i> </div> <div>NOME, COGNOME, FIRMA RESPONSABILE</div>

Figura 1: Esempio di Lettera di Presentazione

3.1.8.2 *Diario di Bordo*

Il *Diario di Bordo* è un documento informale ad uso esterno e permette di interagire con il committente_g in modo da aggiornarlo sullo stato di avanzamento del progetto settimanalmente ed è usato anche per chiedere chiarimenti sui temi in cui riscontriamo dubbi o domande. Fino al termine delle lezioni il *Diario di Bordo* veniva redatto settimanalmente dal responsabile che presentava in classe prima dello svolgimento della lezione. Dal termine delle lezioni il *Diario di Bordo* viene redatto settimanalmente dal responsabile e caricato in una cartella denominata diari_di_bordo presente nel drive di gruppo.

La cartella diari_di_bordo si trova al link:

https://drive.google.com/drive/folders/1a0kAZuOSsDEp_AaQUb6OQJ4XQyZRCN

È stato fornito l'accesso in lettura alla cartella al Professore che viene notificato tramite mail (deve contenere il link al *Diario di Bordo* di interesse) al caricamento di un *Diario di Bordo*. Il formato dei diari di bordo condivisi nella cartella è .pdf. Inoltre tutti i diari di bordo una volta redatti vengono inviati anche al canale discord_g "diari-di-bordo-file" in modo da avere una duplice copia a fronte di qualsiasi imprevisto dato che i diari di bordo non vengono inseriti all'interno del repository_g di gruppo.

Struttura *Diario di Bordo*

La struttura del *Diario di Bordo* non rispetta la struttura generale della documentazione. Sono state fissate delle regole per la stesura dei diari di bordo per fornire delle presentazioni il più possibile simili tra loro senza troppi vincoli superflui per tali documenti.

Il **font** utilizzato per la stesura del documento è Helvetica Neue con dimensione 22pt. Il nome del file per ogni *Diario di Bordo* deve rispettare la codifica **diario_di_bordo_AAAA-MM-GG** in modo da permettere l'ordinamento cronologico utilizzando i filtri di ricerca di Google Drive.

Il *Diario di Bordo* si divide in 4 slides che possono essere raggruppate in 3 se l'elenco degli obiettivi raggiunti e quelli futuri ci stanno in una singola slide. Le slides che compongono il *Diario di Bordo* sono:

- Intestazione;
- Obiettivi raggiunti;
- Obiettivi futuri;
- Domande.

Descrizione slides:

- **Intestazione:** presenta nella parte centrale il logo del gruppo compreso di slogan, in basso centrale la data relativa al *Diario di Bordo* e l'anno accademico;
- **Obiettivi raggiunti:** presenta in alto centrale il titolo "Obiettivi raggiunti", a sinistra un elenco puntato con gli obiettivi raggiunti della settimana, in alto a destra il logo del gruppo compreso di slogan ed in basso a destra la data del *Diario di Bordo*;
- **Obiettivi futuri:** presenta in alto centrale il titolo "Obiettivi futuri", a sinistra un elenco puntato con gli obiettivi raggiunti della settimana ed in alto a destra il logo del gruppo compreso di slogan;
- **Domande:** presenta in alto centrale il titolo "Domande", a sinistra un elenco puntato con le domande da porre al Professore, in alto a destra il logo del gruppo compreso di slogan ed in basso a destra la data del *Diario di Bordo*.

3.1.8.3 *Piano di Progetto*

Il documento *Piano di Progetto* ha lo scopo di aiutare il gruppo nella gestione delle risorse a disposizione per portare a termine il progetto entro la data decisa. Il documento ha anche la funzione di monitorare l'avanzamento del progetto in modo da poter applicare miglioramenti continui e azioni correttive basandosi sull'esperienza ottenuta da pianificazioni precedenti.

Struttura *Piano di Progetto*

La struttura del *Piano di Progetto* segue la struttura generale. In aggiunta il contenuto del documento si compone di:

- Analisi dei rischi;
- Pianificazione;
- Preventivo.

Descrizione delle sezioni:

- **Analisi dei rischi:** vengono riportati in forma tabellare i rischi a cui si va incontro aggiudicandosi il capitolato. Ogni rischio fa riferimento ad una categoria di rischi precisa e viene indicato con:
 - Nome;
 - Descrizione;
 - Probabilità che si verifichi;
 - Grado di pericolosità;
 - Precauzioni da prendere;
 - Piano di contingenza.
- **Pianificazione:** la sezione dedicata alle pianificazioni ha lo scopo di indicare l'inizio e la fine di un periodo di pianificazione e di fornire la suddivisione e l'organizzazione delle attività all'interno di tale periodo. Per ogni periodo vengono forniti:
 - Attività che si andranno a svolgere nel periodo;
 - Sottoperiodi in cui viene diviso il periodo principale, ciascuno con un inizio, una fine ed indicando cosa verrà fatto;
 - Diagramma di Gantt_g che illustra graficamente l'ordine temporale delle attività da svolgere tenendo conto di eventuali margini temporali dovuti ad imprevisti vari.
- **Preventivo:** include:
 - Sezione iniziale in cui si vanno ad indicare in forma tabellare le risorse umane disponibili all'inizio del progetto e come queste risorse andranno suddivise;
 - Costo totale calcolato in base alle ore di impegno dei componenti del gruppo ed al ruolo che dovranno ricoprire (ogni ruolo ha un costo orario);
 - Dettaglio periodi, sezione che prevede i preventivi di ciascun periodo di pianificazione. In questa sezione sono fornite le tabelle con le ore che ciascun componente del gruppo dovrà svolgere con relativi ruoli associati ed una tabella con i costi derivati da tali ruoli.

Nel caso del nostro progetto viene prevista una rotazione dei ruoli a scopo didattico, perciò ogni componente userà le sue ore di impegno in modo diverso a seconda della rotazione dei ruoli.

3.1.8.4 Piano di Qualifica

Il *Piano di Qualifica* è un documento che contiene le descrizione dei processi e i controlli necessari per garantire la qualità del prodotto_g. Il *Piano di Qualifica* include: definizioni di processo_g, standard, metodologie e obiettivi di qualità; requisiti per la qualifica del software; linee guida per la formazione e la documentazione del progetto; test di qualifica e di validazione_g.

Struttura Piano di Qualifica

La struttura del *Piano di Qualifica* segue la struttura generale. In aggiunta il contenuto del documento si compone di:

- Introduzione;
- Qualità del processo_g;
- Qualità del prodotto_g.

Descrizione delle varie sezioni:

- **Introduzione:** contiene una breve descrizione dello scopo del documento e del progetto. Successivamente è presente una sezione in cui vengono riportati i riferimenti informativi con i relativi link alle risorse che abbiamo utilizzato per garantire la qualità del prodotto_g.
- **Qualità del processo_g:** viene indicato lo standard utilizzato per garantire la qualità di processo_g. Per ogni tipo di processo_g (primari, supporto, organizzativi) vengono indicati in forma tabellare il nome del processo_g, la descrizione e le metriche utilizzate indicate con un codice identificativo univoco per ognuna di esse. Successivamente viene riportata una tabella che fornisce per ogni metrica i nomi ed i valori significativi previsti. I dettagli su come vengono applicate tali metriche vengono poi descritti nelle *Norme di Progetto*.
- **Qualità del prodotto_g:** viene indicato lo standard utilizzato per garantire la qualità di prodotto_g. Trattandosi di un prodotto_g software si fa riferimento ad uno standard dedicato alle applicazioni software. Vengono indicati in forma tabellare gli obiettivi, la descrizione e il codice che contraddistingue le metriche utilizzate per soddisfare gli obiettivi. Successivamente viene riportata una tabella che per ogni codice ne descrive il significato ed i valori minimi ed ottimi per cui la metrica è rispettata. I dettagli su come vengono applicate tali metriche vengono poi descritti nelle *Norme di Progetto*.

3.1.8.5 Glossario

Documento che contiene la definizione dei termini specifici utilizzati nel progetto. Il suo scopo è quello di fornire una comprensione comune del linguaggio utilizzato e di evitare confusione o interpretazioni errate. Il *Glossario* può anche essere utile ai membri del team e ai collaboratori esterni per mantenere una coerenza nel linguaggio utilizzato nel corso del progetto.

Struttura Glossario

L'indice del *Glossario* contiene le iniziali dei termini contenuti in esso ordinati in ordine alfabetico. Alla fine di ogni sezione dell'indice è stato inserito un pagebreak in modo che termini di *Glossario* con iniziali differenti non compaiono nella stessa pagina.

3.1.9 Metriche

Per perseguire la qualità sulla documentazione prodotta si è deciso di adottare le seguenti metriche:

- **MPC10: Indice di Gulpease.**

3.2 Gestione della configurazione

3.2.1 Scopo

La gestione della configurazione è un processo_g che mira a gestire e controllare i cambiamenti apportati a un prodotto_g software o a un sistema durante il suo ciclo di vita_g. La gestione della configurazione per la documentazione descrive come vengono identificate, controllate, tracciate e gestite le versioni di un documento.

3.2.2 Descrizione

In questa sezione sono riportate tutte le norme relative alla configurazione degli strumenti utilizzati per il tracciamento e l'organizzazione dei documenti e del codice.

3.2.3 Versionamento

Ogni versione del documento è identificata da un codice di versione nel formato **X.Y.Z** dove:

- **X**: indica una versione approvata dal responsabile, la numerazione parte da 0 e la prima versione approvata è la 1.0.0;
- **Y**: indica una versione che è stata sottoposta ad una revisione completa dopo un numero consistente di modifiche. La numerazione inizia da 0 e si azzerà ad ogni incremento di X. La prima versione è la 0.1.0;
- **Z**: indica una versione modificata da parte dei redattori del documento, con una conseguente verifica. La numerazione parte da 1 e si azzerà ad ogni incremento di X o Y. La prima versione modificata è la 0.0.1.

3.2.3.1 Sistemi software utilizzati

Per il versionamento il gruppo ha deciso di utilizzare un repository_g GitHub_g, un servizio di hosting per progetti software che implementa uno strumento di controllo versione distribuito Git_g. Per informazioni più approfondite sulla struttura e la gestione del repository_g si veda sezione [4.2.3 GitHub_g](#).

3.3 Accertamento della qualità

3.3.1 Scopo

Lo scopo del processo_g di accertamento della qualità è quello di garantire e rispettare i requisiti di qualità preposti.

3.3.2 Descrizione

Questo processo_g è direttamente collegato al documento di *Piano di Qualifica* in cui il gruppo si impegna a mantenere, con l'adozione di metriche concordate e discusse, la qualità nei processi e nei prodotti.

3.3.3 Attività per il controllo di qualità

Queste sono le attività che ogni membro deve rispettare per il mantenimento della qualità. Ogni membro del gruppo deve:

- Comprendere le attività da svolgere e gli obiettivi da raggiungere;
- Aver compreso le metriche inserite nel documento di *Piano di Qualifica* e rispettare le normative e gli standard definiti al suo interno;
- Mantenere le normative inserite nelle *Norme di Progetto*;

- Riconoscere nel documento su cui si sta lavorando valori di cui tenere conto per un'analisi successiva;
- Utilizzare il sistema di tracciamento delle issue_g fornito da Github_g come descritto nel documento di *Norme di Progetto*;
- Attuare miglioramento continuo, ponendosi obiettivi incrementali.

3.3.3.1 Attuazione del controllo di qualità

Il documento deve essere aggiornato ad ogni aggiunta nel numero di versione centrale ,come descritto nella sezione di [3.2.3 Versionamento](#) , in cui si andrà ad aggiornare i grafici e le misure scritte nel documento. L'analisi e il commento di ogni misura inserita nei grafici verrà successivamente fatta in fase di retrospezione dello Sprint.

3.3.4 Denominazione requisiti nel *Piano di Qualifica*

Per la denominazione nel tracciamento dei requisiti nella sezione di *Specifica dei Test* nel documento di *Piano di Qualifica* viene adottata la seguente scrittura:

T[Tipologia di Test]R[Tipologia di Requisito][Numero Identificativo]

[Tipologia di Test] indica a che tipo di Test si sta riferendo:

- Test di Unità corrisponde a "TU";
- Test di Integrità corrisponde a "TI";
- Test di Accettazione "TA";
- Test di Sistema corrisponde a "TS".

[Tipologia di Requisito] indica a che tipo di Requisito si sta riferendo:

- Requisito Funzionale corrisponderà a "RF";
- Requisito Qualitativo corrisponderà a "RQ";
- Requisito di Dominio corrisponderà a "RD";
- Requisito Prestazionale corrisponderà a "RP".

[Numero Indicativo] indica il numero del Requisito descritto nel documento di *Analisi dei Requisiti*.

3.3.5 Denominazione degli obiettivi di qualità nel *Piano di Qualifica*

Per la denominazione delle metriche di prodotto_g è stato adottata la seguente scrittura:

MPD[Numero]

Il Numero indicato è un numero che incrementa in base al numero di metriche per sezione. Per la denominazione delle metriche di processo_g è stato adottata la seguente scrittura:

MPC[Numero]

Il Numero indicato è un numero che incrementa in base al numero di metriche per sezione. Viene inoltre inserito il nome della metrica, il valore minimo e il valore ottimo.

La descrizione dettagliata di ogni metrica la si può trovare alla sezione [7 Metriche di qualità](#).

3.3.6 Metriche

Per tracciare la soddisfazione dei requisiti è stata concordata una semplice metrica.

- **MPC11: Metriche soddisfatte.**

3.4 Verifica_g

3.4.1 Scopo

Lo scopo del processo_g di verifica_g è quello di accertare che non siano stati commessi errori nello svolgimento delle attività prefissate. Questo processo_g viene applicato costantemente sia durante la stesura della documentazione, che durante lo sviluppo del software.

3.4.2 Descrizione

In questa sezione sono descritte le norme necessarie ad applicare il processo_g di verifica_g in modo efficace.

- **Analisi statica:** questo tipo di analisi va a valutare la documentazione o il software senza bisogno di esecuzione. Si controlla che il prodotto_g sia corretto rispetto a determinate regole, che soddisfi determinate caratteristiche, che non abbia errori.
Vengono inoltre adottati dal gruppo i seguenti metodi di lettura:
 - **Walkthrough:** per attuare questo metodo si eseguono letture ad ampio spettro alla ricerca di errori. Non appena il verificatore trova un errore, sarà compito suo e del redattore di discutere a una possibile soluzione;
 - **Inspection:** metodo utile a eseguire una lettura mirata del prodotto in esame, rilevando errori specifici. La ricerca si focalizza quindi su errori noti redigendo una checklist, cioè una lista di controllo che verrà utilizzata dal verificatore.
- **Analisi dinamica:** questo tipo di analisi invece richiede esecuzione, e, per questo, può essere applicata solo al codice. Si tratta di verificare se il comportamento del prodotto_g software durante l'esecuzione è corretto, soddisfatti i requisiti preposti.

3.4.3 Verifica_g della documentazione

La verifica_g viene svolta da due verificatori prima del merge con il branch documentation all'apertura di una pull request. Il primo verificatore sarà quello che esaminerà il documento seguendo una checklist. Il documento verrà esaminato a partire dalle modifiche fatte dopo l'ultima versione verificata, deducibili tramite il registro delle modifiche.

La verifica_g di un documento prevede di seguire la seguente checklist:

- Lettura esplorativa del testo;
- Seconda lettura del testo per capirne meglio i concetti esposti;
- Segnalare i concetti poco chiari o errati;
- Valutare se il testo è organizzato in maniera opportuna ed eventualmente suggerire un'alternativa migliore;
- Valutare se ci sono parti mancanti ed eventualmente segnalarne la mancanza;
- Segnalare errori grammaticali;
- Segnalare le parti che non rispettano le *Norme di Progetto* riguardanti la stesura del documento;
- Segnalare i termini che sono contenuti anche nel *Glossario* a cui non sono stati inseriti i pedici "g";
- Segnalare i nomi dei documenti che non sono stati scritti seguendo la convenzione;
- Segnalare le parti del registro delle modifiche mancanti o che non sono state compilate correttamente seguendo le *Norme di Progetto*;
- Segnalare errori nell'intestazione del documento.

In caso non ci sia nulla da segnalare il verificatore dovrà spuntare come approvata la Pull Request nella sezione dedicata su GitHub_g. A questo punto se un secondo verificatore noterà la necessità di qualche altro cambiamento da apportare lo segnalerà a chi ha fatto la stesura, se invece ritiene che il documento è corretto dovrà:

- Creare la riga nel registro delle modifiche, nel file `changelog_input.tex`, inserendo la nuova versione secondo le regole di versionamento, i nomi dei verificatori e in descrizione "Verifica_g";
- Spuntare come approvata la Pull Request nella sezione dedicata su GitHub_g;
- Confermare il merge secondo le norme descritte nella sezione [4.2.3.4 Pull Requests](#).

3.4.4 Verifica del codice

L'attività di analisi dinamica del codice si concretizza con lo sviluppo e l'utilizzo dei test. I test servono per assicurare un certo comportamento delle componenti oppure dell'intera applicazione. Esistono più tipi di test ed ogni tipologia va a controllare aspetti diversi del software, nello specifico:

- *Test di unità*: è il tipo di test che verifica se i singoli moduli funzionano correttamente. L'obiettivo principale del test unitario è identificare, analizzare e correggere i difetti in ciascuna unità isolandola dal sistema. È compito del programmatore che implementa un modulo di scrivere il test di unità corrispondente. Inoltre i test di unità devono rispettare la struttura AAA che prevede di dividere il test in queste 3 parti:
 - *Arrange*: si prepara l'oggetto e i prerequisiti del test;
 - *Act*: si svolge l'effettiva componente da testare;
 - *Assert*: si verifica che i risultati ottenuti nella fase di Act corrispondano ai risultati attesi.
- *Test di integrazione*: è il tipo di test che verifica che i moduli già testati singolarmente si comportino correttamente anche quando interagiscono tra di loro;
- *Test di regressione*: è il tipo di test che verifica che dopo l'aggiunta di una nuova feature il codice complessivo non perda di qualità, ovvero che le componenti già integrate continuino a comportarsi nel modo atteso;
- *Test di sistema*: è il tipo di test che verifica l'intero comportamento del sistema. In particolare controlla che i requisiti necessari vengano soddisfatti;
- *Test di accettazione*: è il tipo di test che verifica che l'applicazione nel suo complesso soddisfi pienamente i requisiti dal punto di vista strettamente funzionale. Viene effettuato alla fine dello sviluppo dell'applicazione, quando ha già superato tutti gli altri test, e ne precede il rilascio.

Ogni test è caratterizzato dai seguenti parametri:

- **ambiente**: sistema hardware e software sul quale verrà eseguito il test;
- **stato iniziale**: stato iniziale dal quale il test viene eseguito;
- **input**: dati in ingresso immessi;
- **output**: dati in uscita attesi;
- **istruzioni aggiuntive**: ulteriori istruzioni su come deve essere eseguito il test e su come interpretare i risultati ottenuti.

Per ottenere un buon test è necessario :

- che sia ripetibile;
- che specifichi l'ambiente di esecuzione;
- che indichi input e output richiesti;
- che in caso di fallimento fornisca informazioni sul motivo di quest'ultimo tramite una serie di eccezioni che devono essere previste nella scrittura del test.

3.4.4.1 Codice identificativo

I test vengono identificati tramite un codice così definito:

T[tipo][codice identificativo]

Dove il codice identificativo è un numero univoco e il tipo può essere:

- **U**: test di unità;
- **I**: test di integrazione;
- **R**: test di regressione;
- **S**: test di sistema;
- **A**: test di accettazione.

3.4.4.2 Continuous Integration

I test verranno svolti secondo il workflow della CI (Continuous Integration) che prevede che venga effettuata la build del codice ogni volta che viene eseguito un commit nella repository, durante la quale avviene anche l'esecuzione automatica di tutti i test implementati.

Nel caso in cui la build dovesse fallire sarà necessario trovare la causa del fallimento con la massima priorità e in caso non fosse possibile identificarla in poco tempo è necessario ripristinare la repository all'ultimo commit nel quale la build aveva avuto successo.

Per evitare quindi di perdere grosse porzioni di lavoro, è buona prassi svolgere commit frequenti, di portata non eccessiva. La buona implementazione di questa pratica garantisce che il codice all'interno della repository sia sempre funzionante.

3.4.4.3 Procedimento di verifica del software

Qui vengono riportate le varie casistiche che posso scaturire durante la fase di test, e il modo in cui comportarsi a seguito del loro successo o fallimento:

- Il programmatore apporta cambiamenti alla repository effettuando un commit;
- Viene eseguita la build dell'intera applicazione tramite il processo di CI:
 - Se la build ha successo si passa al punto successivo;
 - Se la build fallisce il programmatore cerca di correggere l'errore nel codice e se non ci riesce ritorna all'ultimo commit in cui il codice è verificato;
- Vengono eseguiti i test:
 - Se tutti i test hanno successo si passa al punto successivo;
 - Se almeno un test fallisce il programmatore cerca di correggere l'errore nel codice e se non ci riesce ritorna all'ultimo commit in cui il codice è verificato;
- Vengono calcolate in automatico la copertura del codice e le metriche riguardanti la codifica
- Vengono registrati i valori nel cruscotto di qualità.

3.4.4.4 Strumenti

- Per l'attività di analisi statica si utilizza SonarCloud, un software configurabile ed integrabile con le Action di Github che permette di controllare la qualità del codice;
- Per l'attività di analisi dinamica si utilizza Cypress, un software integrabile con le Action di GitHub che facilita la scrittura dei test;
- Per l'implementazione del workflow CI vengono usate le Action messe a disposizione di GitHub, configurabili tramite un file .yaml oppure .yml.

3.4.5 Metriche

Per mantenere la qualità nella verifica_g è stata scelta la seguente metrica.

- **MPC12: Code coverage.**

3.5 Validazione_g

3.5.1 Scopo

Lo scopo del processo_g di validazione_g è quello di determinare se il prodotto_g finale sia conforme con le aspettative preposte e rispetti i requisiti minimi concordati con il proponente_g. Questo processo_g avviene dopo quello di verifica_g e sarà il responsabile a stabilire se il prodotto_g è accettabile o ha bisogno di ulteriori verifiche.

4 Processi Organizzativi

I processi organizzativi mirano a gestire i processi e il loro miglioramento, l'organizzazione degli strumenti di supporto e la gestione del personale.

4.1 Gestione Organizzativa

4.1.1 Scopo

Questo processo_g ha come scopo quello di gestire al meglio i vari processi. Contiene tutte le attività e i compiti utili a tale scopo.

4.1.2 Descrizione

In questa sezione sono descritte tutte le norme che servono per gestire al meglio l'organizzazione del gruppo, le comunicazioni, la ripartizione dei compiti e dei ruoli e le attività da svolgere.

4.1.3 Pianificazione

4.1.3.1 Ruoli

I componenti del gruppo si suddivideranno nei seguenti ruoli per periodi di circa 2-3 settimane (dipendentemente dalle esigenze del periodo) e al termine del periodo i ruoli verranno risuddivisi. Visto che nelle varie fasi di sviluppo del progetto le attività da svolgere variano, non sempre sarà necessario coprire tutti i ruoli.

Inoltre sarà necessario tenere traccia delle ore che ogni componente dedica al progetto ed il ruolo associato a quelle ore, in modo da andare a rispettare la tabella degli impegni individuali.

Per questo tracciamento verrà utilizzato un foglio Excel_g in cui ogni componente del gruppo segnerà le ore di lavoro settimanalmente. I ruoli e le loro competenze sono i seguenti:

- **Responsabile:** deve avere la visione d'insieme del progetto e coordinare i membri, inoltre si occupa di rappresentare il gruppo con le interazioni esterne (proponente_g, committente_g ecc...). Le sue competenze specifiche sono:
 - Ad ogni iterazione_g c'è un solo responsabile;
 - Presentare il *Diario di Bordo* in aula;
 - Redarre l'ordine del giorno prima di ogni meeting interno del gruppo;
 - Suddivide le attività del gruppo in singole issue_g (ma non le assegna ai membri del gruppo);
 - In fase di release_g si occupa di approvare_g tutti i documenti che necessitano approvazione_g.
- **Analista:** si occupa di trasformare i bisogni del proponente_g nelle aspettative che il gruppo deve soddisfare per sviluppare un prodotto_g professionale. Le sue competenze specifiche sono:
 - Interrogare il proponente_g riguardo allo scopo del prodotto_g e le funzionalità che deve avere;
 - Studiare le risposte del proponente_g per identificare i requisiti e redarre l' *Analisi dei Requisiti*.
- **Amministratore:** si occupa del funzionamento, mantenimento e sviluppo degli strumenti tecnologici usati dal gruppo. Le sue competenze specifiche sono:
 - Ad ogni iterazione_g basta un solo amministratore;
 - Gestione delle segnalazioni e problemi dei membri del gruppo riguardanti problemi e malfunzionamenti con gli strumenti tecnologici;
 - Valuta l'utilizzo di nuove tecnologie e ne fa uno studio preliminare per poter presentare al gruppo i pro e i contro del suo utilizzo;
 - Controllo giornaliero delle project board_g per garantire una buona organizzazione.

- **Progettista:** si occupa di scegliere la modalità migliore per soddisfare le aspettative del committente_g che gli analisti hanno ricavato dall'*Analisi dei Requisiti*. Le sue competenze specifiche sono:
 - Scegliere eventuali pattern architetturali da implementare;
 - Sviluppare lo schema UML_g delle classi.
- **Programmatore:** si occupa di implementare le scelte e i modelli fatti dal progettista. Le sue competenze specifiche sono:
 - Scrivere il codice atto a implementare lo schema delle classi;
 - Scrivere eventuali test;
 - Scrivere la documentazione per la comprensione del codice che scrive.
- **Verificatore:** si occupa di controllare che ogni file che viene caricato in un branch protetto della repository_g sia conforme alle *Norme di Progetto*. Le sue competenze specifiche sono:
 - Controllare i file modificati o aggiunti durante una pull request tra un ramo non protetto e un ramo protetto siano conformi alle *Norme di Progetto* e cercano errori di altra natura (ortografici, sintattici, logici, build ecc...).

4.1.3.2 Sprint

Seguendo il modello Scrum_g, i compiti derivanti dai processi di sviluppo vengono suddivisi in sprint dalla durata di una settimana.

Questo rende l'avanzamento del prodotto più gestibile e permette al gruppo di produrre risultati in modo più rapido e con una maggiore flessibilità.

Ogni sprint viene gestito da un amministratore che si occupa di creare e organizzare le issue associate allo sprint in modo da rendere il lavoro degli altri membri più semplice e trasparente.

È compito dell'amministratore dello sprint quello di assicurarsi che sia avvenuta la verifica del Piano di Progetto e delle Norme di Progetto prima dell'inizio dello sprint successivo, in modo da avere la documentazione adatta ad affrontare il periodo successivo.

4.1.3.3 Metodo di lavoro

A seguito della revisione di avanzamento RTB il gruppo ha deciso di cambiare il metodo di lavoro, per rendere più efficace il lavoro svolto. Si è deciso di passare ad un modello Agile, utilizzando la tecnica degli sprint del metodo Scrum. Ogni sprint viene organizzato nel seguente modo:

- **Sprint planning:**
 - Si stabiliscono in gruppo quali sono le attività da svolgere durante lo sprint;
 - Ogni componente del gruppo segnala le ore che può mettere a disposizione tramite un apposito foglio Google;
 - Il responsabile definisce gli obiettivi dello sprint nel *Piano di Progetto*, aggiornando il backlog, poi assegna i compiti e crea le issue su GitHub in base alla disponibilità dei membri del gruppo.
- **Daily Scrum:** ogni giorno si compila un apposito foglio Google scrivendo quali compiti si andranno a svolgere. Il responsabile controlla l'andamento dello sprint contattando i componenti del gruppo.
- **Sprint review:**
 - Si fa un consuntivo complessivo: ogni membro del gruppo riferisce quello che ha svolto e gli eventuali dubbi che ha riscontrato;
 - Viene fatta una lista degli obiettivi raggiunti e quelli non raggiunti.
- **Sprint retrospective:** si fa una valutazione di quello che è andato bene durante lo sprint e di quello che è da migliorare, per capire cosa continuare o smettere di fare allo sprint successivo.

4.1.4 Gestione delle comunicazioni

Le comunicazioni per tutta la durata del progetto potranno essere interne, ovvero tra i componenti del gruppo; oppure esterne, quindi tra il gruppo e una o più persone esterne al gruppo, in genere il proponente_g o il committente_g.

4.1.4.1 Comunicazioni Interne

Si svolgono tra i componenti del gruppo tramite diversi canali di comunicazione. Ogni canale è dedicato a un diverso tipo di comunicazione e comporta delle regole da seguire, per aumentare la produttività del gruppo.

- **Telegram:** viene utilizzato per delle comunicazioni brevi;
- **Discord_g:** sono stati creati diversi canali, che possono essere:
 - **Testuali:** questi si suddividono in canali per argomento o per ruolo. I canali per argomento vengono utilizzati esclusivamente per trattare dell'argomento in questione, mentre i canali per ruolo sono utili per le comunicazioni tra i membri del gruppo che sono assegnati allo stesso ruolo nello stesso periodo e hanno bisogno di comunicare tra loro;
 - **Vocali:** vengono utilizzati per riunioni brevi, principalmente di aggiornamento.
- **Zoom_g:** viene utilizzato per le riunioni interne del gruppo, al termine delle quali verrà redatto un *Verbale*.

4.1.4.2 Comunicazioni Esterne

Le comunicazioni esterne si svolgono tra il gruppo e una persona esterna, generalmente il proponente_g o il committente_g. Con il committente_g verranno utilizzati i canali suggeriti da quest'ultimo, mentre con il proponente_g si è stabilito di utilizzare:

- **e-mail:** per comunicazioni brevi e immediate, si farà uso dell'indirizzo e-mail del gruppo. Il compito di scrivere le e-mail è assegnato al responsabile, che, dopo una breve verifica_g del messaggio con il gruppo avrà il compito di inviare la e-mail e notificare il gruppo alla risposta;
- **Google Chat:** per messaggi brevi, sempre scritti dal responsabile;
- **Google Meet:** per riunioni esterne in videochiamata. Queste riunioni potranno essere richieste sia dal gruppo, che dal proponente_g e al termine verrà redatto un *Verbale*.

4.1.5 Gestione delle riunioni

Le riunioni si dividono in interne ed esterne. Per ogni riunione verrà redatto un *Verbale*, per poter tenere traccia degli argomenti trattati e delle decisioni prese.

4.1.5.1 Riunioni Interne

Una riunione interna si svolge esclusivamente tra i membri del gruppo utilizzando il canale apposito Zoom_g. Per ogni riunione il responsabile sarà incaricato di

- Preparare una scaletta degli argomenti da trattare, che potranno essere poi integrati da eventuali punti di discussione portati dagli altri membri del gruppo;
- Fare da moderatore e dirigere la riunione per ottimizzare il tempo.

Le riunioni si svolgeranno a cadenza settimanale, cercando di trovare giorni e orari agevoli a tutti i membri del gruppo.

4.1.5.2 Riunioni Esterne

Le riunioni esterne si svolgono tra i membri del gruppo e una persona esterna. Possono essere richieste da entrambe le parti e tramite i canali di comunicazione stabiliti, (Google Meet con il proponente_g). Se necessario verrà preparata una presentazione da mostrare ai soggetti esterni per ottimizzare i tempi e facilitare la comprensione. Al termine della riunione verrà redatto un *Verbale*.

4.1.6 Metriche

Abbiamo considerato questa metrica come rilevante per mantenere la qualità.

- **MPC13: Rischi non previsti.**

4.2 Gestione Infrastrutture

4.2.1 Scopo

Lo scopo del processo_g di gestione delle infrastrutture è quello di garantire la disponibilità, l'affidabilità e la sicurezza delle risorse necessarie ai processi organizzativi.

4.2.2 Descrizione

In questa sezione sono riportate le norme relative alla gestione delle infrastrutture. Vengono stabiliti gli strumenti di cui il gruppo farà uso e le relative regole di utilizzo.

4.2.3 GitHub_g

Servizio di hosting per progetti software che implementa uno strumento di controllo versione distribuito Git_g. Oltre alla copia in remoto del repository_g di progetto ogni componente del gruppo ha una propria copia in locale.

Per ottenere una copia del repository_g ogni componente ha scaricato lo strumento Git_g ed eseguendo il comando 'git_g clone' da git_g bash viene creata una cartella collegata alla repository_g di progetto.

Non sono state imposte modalità specifiche sull'interazione con il repository_g remoto in modo da non sconvolgere le abitudini di lavoro di ciascun componente.

I componenti del gruppo abituati ad interagire con GitHub_g da interfaccia grafica possono continuare a farne uso.

4.2.3.1 Repository_g

Il repository_g si può trovare all'indirizzo <https://github.com/7clickers/ShowRoom3D> ed è pubblico. I collaboratori sono i componenti del gruppo SevenClickers che utilizzano il proprio account GitHub_g personale per collaborare al progetto.

4.2.3.2 Branching

I branch si dividono in:

- **Branches protetti:**
 - **main:** contiene le versioni di release_g del software;
 - **documentation:** contiene i template latex_g e rispettivi pdf della documentazione. I documenti presenti in documentation sono stati approvati dal responsabile o almeno verificati dai verificatori;
 - **dev:** contiene il codice del prodotto verificato dai verificatori.
- **Branches derivanti da documentation:** viene creato un branch libero per ogni documento che si sta modificando. Il nome del branch deve essere:

- Uguale al nome del documento;
- Scritto tutto in minuscolo;
- Gli spazi tra le parole devono essere sostituiti con il trattino basso.

In ogni branch possono lavorare uno o più componenti del gruppo, a condizione che ognuno lavori su parti diverse del documento a cui fa riferimento il branch. Ad ogni componente che lavora sul branch viene assegnata una *issue_g* da risolvere. Una volta che tutte le *issue_g* relative al branch sono state risolte ed è avvenuta la verifica_g, si fa il merge e il branch viene eliminato.

- **Branches derivanti da dev:** viene creato un branch libero per ogni feature da aggiungere. Il nome del branch deve essere:

- Uguale al nome della feature;
- Scritto tutto in minuscolo;
- Gli spazi tra le parole devono essere sostituiti con il trattino basso.

Ad ogni branch è associata una feature, a cui lavora un componente del gruppo. Una volta completata la feature, sempre nello stesso branch verranno svolti i test di unità dal verificatore. Successivamente, prima di fare il merge con il branch protetto dev, si dovranno eseguire con successo i test di integrazione.

4.2.3.3 Commits

È preferibile che ogni commit abbia una singola responsabilità per cambiamento.

I commits non possono essere effettuati direttamente sui branch protetti ma per integrare delle aggiunte o modifiche sarà necessario aprire una Pull Request. All'approvazione_g di una Pull Request tutti i commit relativi al merge verranno raggruppati in un unico commit che rispetti la struttura sintattica descritta in seguito.

I commit dovranno essere accompagnati da una descrizione solo se ritenuta indispensabile alla comprensione del commit stesso.

I messaggi di commit sui **BRANCH PROTETTI** dovranno seguire la seguente struttura sintattica:

`<label><#n_issue><testo>`

dove:

- **label** può assumere i seguenti valori:
 - **feat:** indica che è stata implementata una nuova funzionalità;
 - **fix:** indica che è stato risolto un bug;
 - **update:** indica che è stata apportata una modifica che non sia fix o feat;
 - **test:** qualsiasi cosa legata ai test;
 - **docs:** qualsiasi cosa legata alla documentazione.
- **n_issue:** indica il numero della *issue_g* a cui fa riferimento il commit (se non fa riferimento a nessuna *issue_g* viene omissso);
- **testo:** indica con quale branch è stato effettuato il merge e deve rispettare la forma: merge from <nome branch da integrare> to <nome branch corrente>;
- **descrizione:** se aggiunta ad un commit deve rispondere alle domande WHAT?, WHY?, HOW? ovvero cosa è cambiato, perché sono stati fatti i cambiamenti, in che modo sono stati fatti i cambiamenti.

I messaggi di commit sui **BRANCH LIBERI** dovranno seguire la struttura sintattica dei branch protetti ad eccezione del testo. Il testo dei commit sui branch liberi non è soggetto a restrizioni particolari a patto che indichi in maniera intuitiva i cambiamenti fatti in modo che possano essere compresi anche dagli altri collaboratori.

4.2.3.4 Pull Requests

Per effettuare un merge su un branch protetto si deve aprire da GitHub_g una Pull Request. La Pull Request permette di verificare il lavoro svolto prima di integrarlo con il branch desiderato.

Alla creazione di una Pull Request bisogna associare:

- I verificatori in carica hanno il compito di trovare eventuali errori o mancanze e fornire un feedback riguardante il contenuto direttamente su GitHub_g richiedendo una review con un review comment sulla parte specifica da revisionare o con un commento generico.
Non sarà possibile effettuare il merge finché tutti i commenti di revisione non saranno stati risolti e la Pull Request approvata da due verificatori;
- L'issue_g associata nell'opzione "Development" che verrà chiusa alla risoluzione della Pull Request;
- La Projects Board di cui fa parte;
- Gli assegnatari che hanno il compito di apportare le modifiche necessarie in fase di verifica_g;
- Le labels associate.

Per i commit relativi alle Pull Requests seguire le regole descritte nella sezione [4.2.3.3 Commits](#) per i branch protetti.

4.2.3.5 Milestone_g

Una milestone_g indica un traguardo intermedio significativo per il progetto. Ad essa possono venire assegnate delle issues_g per verificarne il raggiungimento. Ogni milestone_g ha una scadenza che viene discussa e fissata da tutto il gruppo. Oltre alle 2 milestone_g + 1 milestone_g facoltativa fissate dal committente_g il gruppo ne creerà ulteriori per scandire più nel dettaglio i passi che ci porteranno ad ottenere i risultati prefissati. Una delle milestone_g create dal gruppo durante il completamento della technology baseline relativa alla prima milestone_g imposta dal committente_g (Requirements and Technology Baseline) riguarda l'implementazione di un PoC_g (Proof of concept_g).

4.2.3.6 Projects Board

Vengono utilizzate molteplici project board_g per tracciare le issues_g della repo. Una project board_g che funge da backlog del progetto e altre che corrispondono agli sprint.

- Le project board degli sprint sono suddivise in queste sezioni:
 - **Todo:** issue_g che non sono ancora state iniziate o che non sono ancora state assegnate;
 - **In Progress:** issue_g che sono state assegnate e a cui almeno un membro a cui è stata assegnata ha iniziato a lavorarci;
 - **Pull Request:** issue_g che è in fase di integrazione e necessita della verifica_g dei verificatori. Corrisponde all'inizio di una pull request;
 - **Done:** issue_g che sono state chiuse e che sono state verificate (se necessitano di verifica_g);
 - **Approved:** issue_g con label "Da Approvare_g" che hanno ottenuto l'approvazione_g del responsabile subito dopo la verifica_g; Per tutte le issues_g che richiedono un'approvazione_g solo al momento della consegna è stata creata la project board_g dedicata alle approvazioni_g.

Inoltre nelle project board_g degli sprint vengono registrate delle issue_g che non richiedono verifica_g, approvazione_g o neanche integrazione, con lo scopo di monitorare meglio il lavoro di ogni membro del team.

Queste issue_g verranno chiuse e archiviate manualmente una volta che avranno terminato la loro utilità, un esempio può essere la seguente issue_g:

Diario di Bordo 21-11-22; questa issue_g non necessita verifica_g, approvazione_g o integrazione perchè non è di interesse caricare il file nella repo, però è utile tracciare lo svolgimento della issue_g.

Esempio di work flow issues_g:

1. Viene creata l'issue_g ed inserita all'interno della sezione "To Do" della project board_g dello sprint;
2. Quando viene presa in carico l'issue_g viene spostata nella sezione "In Progress" fino al suo completamento;
3. L'incaricato alla risoluzione dell'issue_g apre una pull request a cui assegna l'issue_g in modo che venga chiusa in automatico al momento dell'integrazione con il branch di destinazione;
4. Vengono fatte le verifiche opportune dai verificatori e le conseguenti modifiche prima di accettare la pull request;
5. Dopo l'accettazione la pull request viene postata insieme alle issues_g associate nella colonna "Done";

4.2.3.7 Issue_g Tracking System

Il gruppo utilizza l'issue_g tracking system di Github_g per tenere traccia delle issue_g. Le issues_g verranno determinate dal responsabile, ma la loro assegnazione verrà effettuata dai membri del gruppo, in base alla priorità delle issues_g, i loro ruoli e alle loro disponibilità temporali.

Per marciare le issues_g secondo criteri di interesse (come ambito o priorità) vengono utilizzate le labels.

Labels:

- **Da approvare_g**: indica una issue_g o pull request che necessita di approvazione_g;
- **Da verificare**: indica una issue_g o pull request che necessita di verifica_g;
- **Documentazione**: indica una issue_g o pull request riguardante la documentazione;
- **P=Bassa**: indica una issue_g o pull request a priorità bassa, con una scadenza di massimo 15 giorni;
- **P=Media**: indica una issue_g o pull request a priorità media, con una scadenza di massimo 7 giorni;
- **P=Alta**: indica una issue_g o pull request a priorità alta, con una scadenza di massimo 3 giorni;
- **Presentazione**: indica una issue_g riguardante la redazione di una presentazione in classe o al proponente_g.

Nel caso un membro del gruppo dovesse rendersi conto che l'issue_g che sta svolgendo potrebbe essere suddiviso in ulteriori issue_g, dovrà rivolgersi al responsabile, che è l'unico che può aggiungere, modificare o eliminare le issue_g.

Per raggruppare più issue_g si marca ciascuna con la label di raggruppamento. Il nome di una label di raggruppamento viene preceduto da una G maiuscola (che sta per gruppo) seguito dal nome dell'attività da svolgere. esempio: "G Piano di Progetto". Al completamento di tutte le issues_g di un gruppo si potrà scegliere se eliminare la label o tenerla per raggruppare issues_g future dello stesso gruppo. indica un gruppo di labels relative al documento *Piano di Progetto*.

Utilizzo delle checkbox:

Una issue_g può essere suddivisa in più attività tramite delle checkbox in base alla grandezza o alla necessità di suddividere il lavoro. Al completamento di un'attività si spunta la checkbox corrispondente in modo da avvisare il gruppo sullo stato di avanzamento di quella issue_g.

4.2.4 Discord_g

Strumento utilizzato per la comunicazione tra i componenti del gruppo. Sono creati diversi canali con diverse funzioni:

- **Testuali:** si condividono informazioni in formato testuale;
- **Vocali:** si effettuano videochiamate brevi, che non necessitano di essere verbalizzate;
- **Risorse:** si condivide materiale da consultazione (ed esempio documenti, link, ecc...).

I canali sono creati secondo le necessità del periodo, in accordo con tutti i componenti del gruppo. Su Discord_g è anche possibile controllare in ogni momento quale ruolo è assegnato ad ogni membro del gruppo.

4.2.5 Metriche

Il processo_g di gestione delle infrastrutture non utilizza metriche qualitative particolari.

5 Standard di qualità ISO/IEC 9126

Questa sezione descrive in dettaglio lo standard ISO/IEC 9126 utilizzato dal gruppo. Le norme di questo standard descrivono un modello di qualità del software, definiscono le caratteristiche che lo determinano e propongono metriche per la misurazione. Le norme constano di quattro parti:

- Parte 1: Metriche per la qualità esterna;
- Parte 2: Metriche per la qualità interna;
- Parte 3: Modello della qualità del software;
- Parte 4: Metriche per la qualità in uso.

5.1 Qualità esterne

Le metriche esterne misurano i comportamenti del prodotto_g software rilevabili dai test, dall'operatività, dall'osservazione durante la sua esecuzione. L'esecuzione del prodotto_g software è fatta in base al suo utilizzo in funzione degli obiettivi di business stabiliti ed in un contesto organizzativo e tecnico rilevante.

5.2 Qualità interne

Le metriche interne si applicano alla parte non eseguibile del software (come le specifiche tecniche o il codice sorgente) durante le fasi di progettazione e codifica. Durante le fasi di sviluppo del software, quindi, i prodotti intermedi sono valutati tramite metriche interne che misurano le proprietà intrinseche del prodotto_g. Le metriche interne permettono ad utenti e sviluppatori di individuare eventuali problemi che potrebbero inibire la qualità finale del prodotto_g.

5.3 Modello di qualità

Il modello definisce le caratteristiche di qualità. A ciascuna di esse sono associate sotto-caratteristiche. La tabella successiva descrive le caratteristiche e le sue sotto-caratteristiche del software proposti dal modello. Il modello presenta le seguenti caratteristiche:

5.3.1 Funzionalità

Si intende quella capacità di un prodotto_g software di determinare le esigenze richieste tramite delle funzioni, sotto precise condizioni. Le sue sotto-caratteristiche sono rispettivamente:

- **Appropriatezza:** rappresenta la capacità di fornire un appropriato insieme di funzioni che permettano agli utenti di svolgere determinate task e di raggiungere gli obiettivi prefissati;
- **Accuratezza:** rappresenta la capacità del software di fornire i risultati o gli effetti attesi con il livello di precisione richiesta;
- **Interoperabilità:** rappresenta la capacità del software di interagire con uno o più sistemi specificati;
- **Conformità:** rappresenta la capacità del software di aderire a standard, convenzioni e regolamenti di carattere legale o prescrizioni simili che abbiano attinenza con la funzionalità;
- **Sicurezza:** rappresenta la capacità del software di proteggere le informazioni ed i dati in modo che, persone o sistemi non autorizzati, non possano accedervi e quindi non possano leggerli o modificarli.

5.3.2 Affidabilità

Si intende quella capacità di un prodotto_g software di mantenere il livello di prestazione quando utilizzato sotto certe condizioni specifiche. Le sue sotto-caratteristiche sono rispettivamente:

- Maturità: rappresenta la capacità del software di evitare che si verifichino errori o siano prodotti risultati non corretti in fase di esecuzione;
- Tolleranza agli errori: rappresenta la capacità del software di mantenere il livello di prestazioni in caso di errori nel software o di violazione delle interfacce specificate;
- Recuperabilità: rappresenta la capacità del software di ripristinare il livello di prestazioni e di recuperare i dati direttamente coinvolti in caso di errori o malfunzionamenti;
- Aderenza: rappresenta la capacità del software di aderire a standard, convenzioni e regole relative all'affidabilità.

5.3.3 Efficienza

Si intende quella capacità di un prodotto_g software di realizzare le funzioni richieste nel minor tempo possibile ed utilizzando nel miglior modo le risorse necessarie. Le sue sotto-caratteristiche sono rispettivamente:

- Comportamento rispetto al tempo: rappresenta la capacità del software di fornire appropriati tempi di risposta, tempi di elaborazione e quantità di lavoro eseguendo le funzionalità previste sotto determinate condizioni di utilizzo;
- Utilizzo delle risorse: rappresenta la capacità del software di utilizzare un appropriato numero e tipo di risorse in maniera adeguata;
- Conformità: rappresenta la capacità del software di aderire a standard e convenzioni relative all'efficienza.

5.3.4 Usabilità

Si intende quella capacità di un prodotto_g software di essere comprensibile e appreso dall'utente, quando usato sotto condizioni specificate. Le sue sotto-caratteristiche sono rispettivamente:

- Comprensibilità: rappresenta la capacità del software di permettere all'utente di capire le sue funzionalità e come poterla utilizzare con successo per svolgere particolari task in determinate condizioni di utilizzo;
- Apprendibilità: rappresenta la capacità del software di permettere all'utente di imparare l'applicazione;
- Operabilità: rappresenta la capacità del software di permettere all'utente di utilizzarlo e di controllarlo;
- Attrattività: rappresenta la capacità del software di risultare "attraente" per l'utente;
- Conformità: rappresenta la capacità del software di aderire a standard, convenzioni o regole relative all'usabilità.

5.3.5 Manutenibilità

Si intende quella capacità di un prodotto_g software di essere modificato, includendo correzioni, miglioramenti o adattamenti. Le sue sotto-caratteristiche sono rispettivamente:

- Analizzabilità: rappresenta la capacità del software di poter effettuare la diagnosi sul software ed individuare le cause di errori o malfunzionamenti;

- Modificabilità: rappresenta la capacità del software di consentire lo sviluppo di modifiche al software originale. Si tratta quindi di modifiche al codice o alla progettazione ed alla sua documentazione;
- Stabilità: rappresenta la capacità del software di evitare effetti non desiderati a seguito di modifiche al software;
- Testabilità: rappresenta la capacità del software di consentire la verifica_g e validazione_g del software modificato, cioè di eseguire i test.

5.3.6 Portabilità

Si intende quella capacità di un prodotto_g software di poter essere trasportato da un ambiente di lavoro ad un altro. Le sue sotto-caratteristiche sono rispettivamente:

- Adattabilità: rappresenta la capacità del software di essere adattato a differenti ambienti senza richiedere azioni specifiche diverse da quelle previste dal software per tali attività;
- Installabilità: rappresenta la capacità del software di essere installato in un determinato ambiente;
- Conformità: rappresenta la capacità del software di aderire a standard, convenzioni o regole relative alla portabilità;
- Sostituibilità: rappresenta la capacità del software di sostituire un altro software specifico indipendente, per lo stesso scopo e nello stesso ambiente.

5.4 Qualità in uso

Le metriche delle qualità in uso misurano il grado con cui il prodotto_g software permette agli utenti di svolgere le proprie attività con efficacia, produttività, sicurezza e soddisfazione nel contesto operativo previsto.

- Efficacia: la capacità del software di permettere all'utente di svolgere le funzioni specificate con accuratezza e completezza;
- Produttività: la capacità di far utilizzare all'utente un quantitativo adeguato di risorse in relazione al caso d'uso;
- Soddisfazione: la capacità del software di soddisfare l'utente;
- Sicurezza: la capacità del prodotto_g software di avere livelli accettabili di rischio per dati e persone.

6 Standard di qualità ISO/IEC 12207:1995

Questa sezione descrive in dettaglio lo standard ISO/IEC 12207:1995 scelto dal gruppo per garantire la qualità dei processi del ciclo di vita_g di un software. I processi dello standard contengono attività e compiti che devono essere applicati durante l'acquisizione di un sistema software. Esso contiene tre tipologie di processi che sono:

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

6.1 Processi primari

Questi processi sono essenziali in quanto un progetto è detto tale se e solo se è attivo almeno un processo_g primario. I processi di supporto definiti dallo standard sono i seguenti.

6.1.1 Acquisizione

Il processo_g di acquisizione contiene le attività ed i compiti dell'acquirente. Le attività del processo_g sono le seguenti:

- Iniziazione
- Preparazione della richiesta di proposta;
- Preparazione e aggiornamento del contratto;
- Monitoraggio dei fornitori;
- Accettazione e completamento.

6.1.2 Fornitura

Il processo_g di fornitura contiene le attività e le mansioni del fornitore. Il processo_g può essere. Le attività del processo_g sono le seguenti:

- Iniziazione;
- Preparazione alla risposta;
- Contratto;
- Pianificazione;
- Esecuzione e controllo;
- Revisione e valutazione;
- Rilascio e completamento.

6.1.3 Sviluppo

Il processo_g di sviluppo contiene le attività e i compiti dello sviluppatore. Il processo_g contiene le attività per l'*Analisi dei Requisiti*, la progettazione, la codifica, l'integrazione, il test, l'installazione e l'accettazione relative al sistema se previsto dal contratto. Le attività del processo_g sono le seguenti:

- Implementazione dei processi;
- *Analisi dei Requisiti* di sistema;
- Progettazione architettura di sistema;
- Analisi requisiti software;
- Progettazione architettura software;
- Progettazione dettagliata del software;
- Codifica e test del software;
- Integrazione software;
- Test di qualificazione del software;
- Integrazione del sistema;
- Test di qualificazione del sistema;
- Installazione software;
- Supporto per l'accettazione del software.

6.1.4 Gestione operativa

Il processo_g di gestione operativa contiene le attività e i compiti dell'operatore. Il processo_g riguarda il funzionamento del prodotto_g software e il supporto operativo agli utenti. Le attività del processo_g sono le seguenti:

- Implementazione dei processi;
- Collaudo operativo;
- Operazione di sistema;
- Supporto all'utente.

6.1.5 Manutenzione

Il processo_g di manutenzione contiene le attività e i compiti del manutentore. Questo processo_g viene attivato quando il prodotto_g software subisce modifiche al codice e alla documentazione associata a causa di un problema o necessità di miglioramento o adattamento. Le attività del processo_g sono le seguenti:

- Implementazione di processi;
- Analisi dei problemi e delle modifiche;
- Implementazione della modifica;
- Revisione/Acettazione della manutenzione;
- Migrazione;
- Ritiro del software.

6.2 Processi di supporto

I processi di supporto aiutano le attività di tutti gli altri processi dell'organizzazione a garantire il successo e la qualità del progetto. Questi processi possono essere attivati da un processo_g primario o da un altro processo_g di supporto. Le attività e i compiti in un processo_g di supporto sono di responsabilità dell'organizzazione che esegue tale processo_g. Questa organizzazione garantisce che il processo_g sia in atto e funzionante. I processi di supporto definiti dallo standard sono i seguenti.

6.2.1 Documentazione

Il processo_g di documentazione è un processo_g per la registrazione delle informazioni prodotte da un processo_g o attività del ciclo di vita_g. Il processo_g contiene l'insieme delle attività che pianificano, progettano, sviluppano, producono, modificano, distribuiscono e mantengono quei documenti necessari a tutti gli interessati. Le attività del processo_g sono le seguenti:

- Implementazione di processo_g;
- Produzione;
- Progettazione e sviluppo;
- Manutenzione.

6.2.2 Gestione della configurazione

Il processo_g di gestione della configurazione è un processo_g di applicazione di procedure amministrative e tecniche durante tutto il ciclo di vita_g del software per mantenere l'integrità di tutti i componenti della configurazione e di renderli accessibili a chi ne ha diritto. Le attività del processo_g sono le seguenti:

- Implementazione del processo_g;
- Identificazione della configurazione;
- Controllo della configurazione;
- Valutazione dello stato di configurazione;
- Gestione del rilascio e distribuzione.

6.2.3 Accertamento della qualità

Il processo_g di accertamento della qualità è un processo_g per fornire un'adeguata garanzia che i prodotti ed i processi software nel ciclo di vita_g del progetto siano conformi ai requisiti specificati e aderiscano ai piani stabiliti. Per essere imparziale, la garanzia della qualità deve avere libertà organizzativa e autorità da parte delle persone direttamente responsabili dello sviluppo del prodotto_g software o dell'esecuzione del processo_g nel progetto. Le attività del processo_g sono le seguenti:

- Implementazione del processo_g;
- Accertamento di prodotto_g;
- Accertamento di processo_g;
- Accertamento della qualità di sistema.

6.2.4 Verifica_g

Il processo_g di verifica_g è un processo_g per determinare se i prodotti software di un'attività soddisfano i requisiti o le condizioni loro imposti nelle attività precedenti. Le attività del processo_g sono le seguenti:

- Implementazione del processo_g;
- Verifica_g.

6.2.5 Validazione_g

Il processo_g di validazione_g è un processo_g per determinare se i requisiti e il sistema finale soddisfano l'uso specifico previsto. Le attività del processo_g sono le seguenti:

- Implementazione del processo_g;
- Validazione_g.

6.2.6 Revisione congiunta

Il processo_g di revisione congiunta è un processo_g per valutare lo stato ed i prodotti di un'attività di un progetto in modo appropriato. Le revisioni congiunte avvengono sia a livello di gestione del progetto che a livello tecnico.

- Implementazione del processo_g;
- Revisioni della gestione del progetto;
- Revisioni tecniche.

6.2.7 Audit

Il processo_g di Audit ha lo scopo di determinare in maniera indipendente la conformità di prodotti e processi selezionati ai requisiti, piani e accordi. Le attività del processo_g sono le seguenti:

- Implementazione del processo_g;
- Audit.

6.2.8 Risoluzione dei problemi

Il Processo_g di risoluzione dei problemi è un processo_g per l'analisi e la risoluzione dei problemi, indipendentemente dalla loro natura o origine, che vengono scoperti durante l'esecuzione di processi di sviluppo, funzionamento, manutenzione o altri. L'obiettivo è fornire un mezzo tempestivo, responsabile e documentato per garantire che tutti i problemi scoperti siano analizzati e risolti e che le tendenze siano riconosciute.

- Implementazione del processo_g;
- Risoluzione del problema.

6.3 Processi organizzativi

Le attività e i compiti in un processo_g organizzativo sono di responsabilità dell'organizzazione che utilizza il processo_g. L'organizzazione garantisce che il processo_g sia attivo e funzionante. I processi organizzativi definiti nello standard sono i seguenti.

6.3.1 Gestione organizzativa

Il processo_g di gestione organizzativa ha lo scopo di organizzare e monitorare i processi per il raggiungimento dei loro obiettivi. Il processo_g è stabilito da una organizzazione per assicurare la consistente applicazione di pratiche per l'uso dall'organizzazione e nei progetti. Le attività del processo_g sono le seguenti:

- Inizio e definizione dello scopo;
- Pianificazione;
- Esecuzione e controllo;

- Revisione e valutazione;
- Chiusura.

6.3.2 Gestione delle infrastrutture

Il processo_g di gestione delle infrastrutture è un processo_g per stabilire e mantenere l'infrastruttura necessaria per qualsiasi altro processo_g. L'infrastruttura può includere hardware, software, strumenti, tecniche, standard e strutture per lo sviluppo, il funzionamento o la manutenzione. Le attività del processo_g sono le seguenti:

- Implementazione del processo_g;
- Realizzazione dell'infrastruttura;
- Manutenzione dell'infrastruttura.

6.3.3 Miglioramento del processo_g

Il processo_g di miglioramento è un processo_g per stabilire, valutare, misurare, controllare e migliorare un processo_g del ciclo di vita_g del software. Le attività del processo_g sono le seguenti:

- Stabilimento del processo_g;
- Valutazione del processo_g;
- Miglioramento del processo_g.

6.3.4 Formazione del personale

Il processo_g di formazione del personale è un processo_g per fornire e mantenere personale qualificato. L'acquisizione, la fornitura, lo sviluppo, il funzionamento o la manutenzione di prodotti software dipende in gran parte da personale esperto e qualificato. Le attività del processo_g sono le seguenti:

- Implementazione del processo_g;
- Sviluppo materiale didattico;
- Realizzazione piano formativo.

7 Metriche di qualità

7.1 Metriche per la qualità di processo_g

In questa sezione sono descritte le metriche di qualità di processo_g che il gruppo intende adottare.

7.1.1 MPC01: Planned Value (PV)

Costo (in €) pianificato per realizzare le attività di progetto pianificate al momento del calcolo. Per calcolare questa metrica si utilizzerà la formula:

$$PV = B_{\text{tot}} * \% \text{ lavoro pianificato}$$

Per eseguire il calcolo:

- la % di lavoro pianificato viene calcolata dividendo le ore pianificate per le ore pianificate totali;
- B_{tot} indica il budget totale preventivato.

Valore minimo: $\geq 0 \text{ €}$

Valore ottimo: $\leq \text{Budget at Completion}_g$

7.1.2 MPC02: Actual Cost (AC)

Costo (in €) sostenuto al momento del calcolo. Per calcolare questa metrica si tiene conto di:

$$AC = \sum (\text{Totale ore lavorative per incremento} * \text{costo orario})$$

Per eseguire il calcolo:

- il costo orario è un costo fisso per ruolo ad ora;
- Totale ore lavorative per incremento va a contare per incremento le ore effettive di lavoro per ruolo.

Valore minimo: $\geq 0 \text{ €}$

Valore ottimo: $\leq EAC$

7.1.3 MPC03: Estimated at Completion (EAC)

Un valore (in €) di revisione di costo stimato per la realizzazione del progetto alla data corrente.

$$EAC = AC + ETC$$

Valore minimo: $EAC \leq \text{preventivo} - 8\% \text{ oppure } EAC \geq \text{preventivo} + 5\%$

Valore ottimo: Costo preventivato

7.1.4 MPC04: Earned Value (EV)

Indica il valore (in €) delle attività realizzate nel progetto fino al momento della misurazione.

$$EV = B_{\text{tot}} * \% \text{ lavoro effettivo}$$

Per eseguire il calcolo:

- la % lavoro effettivo è calcolata dividendo le ore effettive per le ore totali preventivate.
- B_{tot} indica il budget totale preventivato.

Valore minimo: $\geq 0 \text{ €}$

Valore ottimo: $\leq EAC$

7.1.5 MPC05: Estimated to Complete (ETC)

Metrica che produce un valore (in €) per indicare il numero di attività necessarie al completamento del progetto.

$$ETC = B_{\text{tot}} - EV$$

Per eseguire il calcolo:

- B_{tot} indica il budget totale preventivato.

Valore minimo: ≥ 0 €

Valore ottimo: $\leq EAC$

7.1.6 MPC06: Cost Variance (CV)

Valore (in €) che indica se il valore del costo attuale del progetto è maggiore, uguale o minore rispetto al costo effettivo. Se $CV \geq 0$ significa che si sta producendo con maggior efficienza risparmiando sul costo rispetto a quanto pianificato, viceversa se negativo.

$$CV = EV - AC$$

Valore minimo: $CV = 0$ €

Valore ottimo: $CV \geq 0$ €

7.1.7 MPC07: Schedule Variance (SV)

Variazione percentuale utilizzata per analizzare se il progetto è in linea, in anticipo o in ritardo rispetto alla schedulazione delle attività di progetto pianificate nella baseline.

$$SV = \frac{EV - PV}{EV} * 100$$

Valore minimo: $\geq -15\%$

Valore ottimo: 0%

7.1.8 MPC08: Budget Variance (BV)

Valore (in €) che indica se alla data corrente si è speso di più o di meno del budget iniziale. Un valore di $BV \geq 0$ significa che il progetto sta spendendo il proprio budget con minor velocità di quanto pianificato, viceversa se negativo.

$$BV = PV - AC$$

Valore minimo: $BV = 0$ €

Valore ottimo: $BV \geq 0$ €

7.1.9 MPC09: Requirements Stability Index (RSI)

Indice che traccia la variazione dei requisiti nell'arco del progetto.

Si calcola tramite la seguente formula:

$$RSI = 1 - \frac{\text{numero requisiti cambiati} + \text{numero requisiti eliminati} + \text{numero requisiti aggiunti}}{\text{numero totale dei requisiti iniziali}} * 100$$

Valore minimo: 70%

Valore ottimo: 100%

7.1.10 MPC10: Indice di Gulpease

Si tratta dell'indice di leggibilità di un testo tarato sulla lingua italiana. I risultati sono compresi tra 0 e 100, dove il valore "100" indica la leggibilità più alta e "0" la leggibilità più bassa. Ai seguenti valori si associano i seguenti significati:

- inferiore a 80 sono difficili da leggere per chi ha la licenza elementare
- inferiore a 60 sono difficili da leggere per chi ha la licenza media
- inferiore a 40 sono difficili da leggere per chi ha un diploma superiore

Viene adottata la seguente formula per calcolarlo:

$$89 + \frac{300 * (\text{numero delle frasi}) - 10 * (\text{numero delle lettere})}{\text{numero delle parole}}$$

Questi sono i valori da noi ritenuti opportuni:

Valore minimo: ≥ 50

Valore ottimo: ≥ 80

7.1.11 MPC11: Metriche soddisfatte

Viene calcolato questo indice tramite una percentuale per tenere conto delle metriche soddisfatte. Una metrica si dice "soddisfatta" se raggiunge almeno il valore minimo imposto sul *Piano di Qualifica*.

Questi sono i valori da noi ritenuti opportuni:

Valore minimo: $\geq 80\%$

Valore ottimo: 100%

7.1.12 MPC12: Code coverage

Viene definita come la percentuale di linee di codice del progetto che sono state eseguite dai test dopo un'esecuzione.

Questi sono i valori da noi ritenuti opportuni:

Valore minimo: $\geq 70\%$

Valore ottimo: $\geq 90\% - 100\%$

7.1.13 MPC13: Rischi non previsti

Il valore è identificato con un numero intero e indica il numero di rischi non previsti durante il corso del progetto.

Questi sono i valori da noi ritenuti opportuni:

Valore minimo: ≥ 0

Valore ottimo: 0

7.2 Metriche per la qualità di prodotto_g

In questa sezione sono descritte le metriche di qualità di prodotto_g che il gruppo intende adottare.

7.2.1 MPD01: Percentuale requisiti soddisfatti

Valore percentuale che serve per indicare i requisiti soddisfatti.

Valore minimo: 100% dei requisiti obbligatori

Valore ottimo: 100% di tutti i requisiti

7.2.2 MPD02: Densità di fallimenti durante l'esecuzione

Si intende la percentuale di failure_g o di esecuzioni non andate a buon fine di determinate azioni. Le eventuali esecuzioni fallite o failure_g sono state segnate dai Programmatori con cui hanno poi calcolato il valore della metrica.

$$\text{Densità di fallimenti} = \frac{\text{numero test eseguiti sul programma falliti}}{\text{numero totale di test eseguiti}} * 100$$

Valore minimo: 20%

Valore ottimo: 10%

7.2.3 MPD03: Tempo medio di risposta

Tempo medio impiegato dal software per rispondere a una richiesta utente o svolgere un'attività di sistema. *Valore minimo:* 4 secondi

Valore ottimo: 2 secondi

7.2.4 MPD04: Tempo di caricamento

Tempo medio di attesa per il caricamento del sito.

Valore minimo: 15 secondi

Valore ottimo: 10 secondi

7.2.5 MPD05: Facilità di apprendimento

Misura l'intuibilità e la facilità di utilizzo del programma.

Valore minimo: 5 minuti

Valore ottimo: 2 minuti

7.2.6 MPD06: Complessità ciclomatica

Metrica utilizzata per misurare la complessità di un programma. Calcolata sul grafo dei cammini linearmente indipendenti percorsi dal software ed i nodi presenti, cioè i punti decisionali del programma.

$$v(G) = e - n + 2p$$

dove:

- $v(G)$ = complessità ciclomatica del grafo G
- e = il numero di archi nel grafo
- n = il numero di nodi nel grafo
- p = il numero di componenti connesse

Valore minimo: ≤ 10

Valore ottimo: ≤ 4

7.2.7 MPD07: Densità dei commenti

Misura la percentuale delle righe di commento sul totale delle righe di codice presenti in un modulo.

$$\text{Densità dei commenti} = \frac{\text{numero righe di commento}}{\text{numero righe di codice}} * 100$$

Valore minimo: 20%

Valore ottimo: 10%

7.2.8 MPD08: Browser supportati

Valore percentuale dei browser supportati dal prodotto_g software.

Da associare a questa metrica è doveroso inserire la versione del browser da cui è stato testato il prodotto_g.

$$\text{Browser supportati} = \frac{\text{browser in cui il prodotto}_g \text{ funziona}}{\text{browser testati}} * 100$$

Valore minimo: 80%

Valore ottimo: 100%