Progetto di Tecnologie Web

TecWeb&Pastorizia - Progetto di Tecnologie Web

giovanni.cavallin.1@studenti.unipd.it

Informazioni sul documento

Redazione	Manuel Vianello - 1102467 Cavallin Giovanni - 1148957 Stefano Panozzo - 1097068
Referente	Cavallin Giovanni - 1148957 giovanni.cavallin.1@studenti.unipd.it

Descrizione

Documento riportante le informazioni relative al progetto di tecnologie web.

Indice

1 Introduzione

Il sito web sviluppato per il progetto di tecnologie web si occupa di fornire delle informazioni relative ad un'azienda agricola di fantasia. Questa azienda produce grano biologico, che si occupa di vendere come business to business, e fornisce alcune attrezzature come terzista per il lavoro nei campi. Lo scopo principale di questo sito è quello di intercettare un bacino d'utenza che in genere fatica a reperire le informazioni qui presentate, perché trasmesse principalmente come passaparola nell'ambiente contadino. Inoltre si occupa di dare visibilità all'azienda agricola, che altrimenti non avrebbe modo di presentarsi in veste web.

2 Processi primari

2.1 Processo di fornitura

2.1.1 Studio di Fattibilità

Si tratta del documento in cui vengono analizzati tutti i capitolati, valutandone pregi e difetti, allo scopo di scegliere quello di maggiore affinità per il gruppo. Dopo che il *Responsabile di Progetto* (il cui ruolo è descritto nel paragrafo 4.2.2.1) avrà riunito il team e discusso con esso di tutti i capitolati, gli *Analisti* avranno il compito di stilare lo *StudioDiFattibilità_v1.0.0* seguendo le considerazioni emerse.
Lo *StudioDiFattibilità_v1.0.0* sarà organizzato come segue:

- **Informazioni sul capitolato**: Vengono ricordati il nome del $progetto_{\mathrm{G}}$, il $proponente_{\mathrm{G}}$ e i $committenti_{\mathrm{G}}$;
- **Descrizione**: Si riassume lo scopo del capitolato;
- **Dominio applicativo**: Si specifica il settore di utilizzo del prodotto finale;
- **Dominio tecnologico**: Si elencano le tecnologie che dovranno essere utilizzate nello sviluppo del capitolato;
- **Aspetti positivi**: Si elencano i motivi che il gruppo potrebbe considerare vantaggiosi;
- **Aspetti negativi**: Si elencano le potenziali criticità che il gruppo dovrà tenere in considerazione nella scelta finale;
- **Valutazione finale**: Si analizzano gli aspetti positivi e quelli negativi riscontrati e si motiva l'eventuale approvazione o esclusione.

2.1.2 Rapporti con il proponente

Una volta scelto il capitolato, si intende instaurare un rapporto quanto più costante e profittevole con la Red Babel allo scopo di:

- Stabilire un accordo in merito allo sviluppo, al mantenimento, al funzionamento e alla consegna del prodotto;
- Realizzare un prodotto che soddisfi totalmente i requisiti obbligatori concordati e quanto più possibile quelli desiderabili;
- Stimare i costi;
- Concordare la qualifica del prodotto.

2.1.3 Collaudo e consegna del prodotto

Una volta terminate le fasi di sviluppo, verifica e validazione si effettuerà il collaudo al fine di dimostrare che tutti i requisiti obbligatori e, possibilmente, anche alcuni dei requisiti opzionali siano stati soddisfatti.
In questa fase inoltre si dovrà dimostrare che l'esecuzione di tutti i test di validazione abbia dato un'esito positivo.
Il team consegnerà in ultima il prodotto finale su un supporto fisico ai committenti Tullio Vardanega e Riccardo Cardin.

2.2 Processo di sviluppo

2.2.1 Attività

2.2.1.1 Analisi dei Requisiti

Gli *Analisti*, una volta terminato lo *StudioDiFattibilità_v1.0.0*, dovranno stilare l'*AnalisiDeiRequisiti_v4.0.0*, che si dovrà attenere alle seguenti regole:

Classificazione dei Requisiti: i requisiti saranno classificati secondo la seguente codifica:

R[Importanza][Tipo][Codice]

dove:

- **Importanza** può assumere questi valori:
  - F: indica un requisito funzionale;
  - Q: indica un requisito di qualità;
  - P: indica un requisito prestazionale;
  - V: indica un requisito di vincolo.
- **Tipo** può assumere questi valori:
  - O: indica un requisito obbligatorio;
  - D: indica un requisito desiderabile;
  - F: indica un requisito facoltativo.
- **Codice** indica il codice identificativo del requisito, è univoco e deve essere indicato in forma gerarchica.

Per ogni requisito si dovrà inoltre indicare una breve descrizione e la fonte, che può essere una tra le seguenti:

- Capitolato: deriva direttamente dal testo del capitolato;
- Verbale: deriva da un incontro verbalizzato;
- Interno: deriva da discussioni interne al team.

Classificazione dei casi d'uso: i casi d'uso saranno classificati secondo la seguente codifica:

UC[Codice padre].[Codice identificativo]

dove:

- Codice padre: indica il codice del caso d'uso padre di quello in esame, se non è identificabile è da omettere;

- Codice identificativo: codice univoco e progressivo del caso d'uso in esame.

Per ogni caso d'uso saranno inoltre identificate le seguenti informazioni:

- **Attori**: indica gli attori coinvolti nel caso d'uso;
- **Descrizione**: chiara, precisa e concisa descrizione del caso d'uso;
- **Precondizione**: indica la situazione che deve essere vera prima dell'esecuzione del caso d'uso;
- **Flusso principale degli eventi**: descrizione composta dal flusso dei casi d'uso figli;
- **Postcondizione**: indica la situazione che deve essere vera dopo l'esecuzione del caso d'uso;
- **Estensioni**: indica quali sono tutte le estensioni, se presenti;
- **Generalizzazioni**: indica quali sono tutte le generalizzazioni, se presenti.

2.2.1.2 Progettazione

I *Progettisti* (il cui ruolo è descritto nel paragrafo 4.2.2.4) dovranno delinerare i requisiti utili alla documentazione specifica e determinare le linee guida da seguire.
La progettazione ha come scopo quello di soddisfare le peculiarità identificate durante l'*AnalisiDeiRequisiti_v4.0.0*. Un altro obiettivo è quello di realizzare un prodotto *manutenibile*$_{\text{G}}$ ovvero, un che abbia una struttura in grado di facilitare i cambiamenti futuri.
Infine deve realizzare al meglio i requisiti di qualità imposti dal committente.

2.2.1.2.1 Linee guida per la progettazione

Dopo aver completato l'*AnalisiDeiRequisiti_v4.0.0* i *Progettisti* dovranno sottostare alle seguenti linee guida per lo sviluppo dell'architettura logica del sistema:

- Si dovrà puntare ad una progettazione chiara e di immediata comprensione;
- Le componenti progettate dovranno essere quanto più riutilizzabili e manutenibili;
- La complessità non dovrà mai essere intrattabile;
- I *Progettisti* dovranno rientrare nei costi e nelle risorse disponibili;
- I *Progettisti* dovranno descrivere i *design pattern*$_{\text{G}}$ che intendono utilizzare per la realizzazione dell'architettura, fornendone una breve descrizione e un diagramma.

2.2.1.2.2 Formattazione

- Il rientro predefinito dovrà essere di un *Tab* per allineare le sezioni di codice;
- La parentesi graffa di apertura sarà alla fine della riga, mentre quella di chiusura a capo;
- Prima e dopo ogni operatore dovrà esserci uno spazio);
- I blocchi saranno tra loro spaziati per una maggiore comprensione;

2.2.2 Strumenti

Gli strumenti utilizzati durante la fase dei processi primari sono:

- **TexStudio**
  Il gruppo ha scelto *TexStudio* come editor multipiattaforma per comporre i documenti in LaTeX$_{\text{G}}$;

- **SWEgo**$_{\text{G}}$ [1]
  Il gruppo ha scelto *SWEgo* come database per la gestione dei casi d'uso, dei requisiti e del loro tracciamento per il documento *AnalisiDeiRequisiti_v4.0.0*. Dopo un'attenta analisi preliminare dello strumento il team ha deciso di utilizzarlo nonostante alcuni problemi rilevati nella generazione del codice LaTeX perché, permette una stesura standardizzata e sempre aggiornata di alcuni capitoli importanti del documento, a fronte di correzioni minori e applicabili in maniera seriale e regolamentata;

- **Astah**$_{\text{G}}$
  Il gruppo ha scelto l'utilizzo di *Astah* per la generazione dei diagrammi dei casi d'uso dell'*AnalisiDeiRequisiti_v4.0.0*;

- **Microsoft Excel 365**
  Il gruppo ha scelto di utilizzare *Microsoft Excel 365*$_{\text{G}}$ per la generazione di grafici e di tabelle da poter inserire all'interno del *PianoDiQualifica_v4.0.0* e *PianoDiProgetto_v4.0.0* perché di immediata realizzazione e visualizzazione;

- **Instagantt** [2]
  Instagantt è un servizio web fortemente legato ad Asana (vedi descrizione negli strumenti dei *processi organizzativi*), strumento con il quale si integra perfettamente, nonostante sia disponibile anche una versione *standalone*$_{\text{G}}$. Esso permette di creare diagrammi di Gantt e gestire in maniera semplificata la timeline e la struttura di un progetto.

- **Visual Studio Code** [3]
  Visual Studio Code è un editor di testo gratuito e multipiattaforma che supporta operazioni di debugging e di versionamento. Permette inoltre di installare vari pacchetti che consentono di personalizzarne l'usabilità e di supportare nuove tecnologie o linguaggi di programmazione. Il gruppo lo ha scelto come editor di testo per scrivere il codice in React e Redux.

- **Truffle** [4]
  Truffle è un ambiente di sviluppo e testing per Ethereum. Il gruppo ha deciso di utilizzarlo perché fornisce un aiuto nello sviluppare, pubblicare e testare gli smart contract.

- **Ganache** [5]
  Ganache è uno strumento legato a Truffle che consente di avere una blockchain virtuale locale utilizzabile per sviluppare contratti, applicazioni ed effettuare dei test. Permette anche di generare degli account di testing utilizzabili durante lo sviluppo.

- **Npm** [6]
  npm è un gestore di pacchetti per il linguaggio Javascript.

- **Remix**
  Remix è l'ambiente di sviluppo integrato ufficiale di Ethereum per il linguaggio di programmazione di smart contract *solidity*$_{\text{G}}$. Viene eseguito all'interno di un browser web, online all'indirizzo `https://remix.ethereum.org/` oppure in locale scaricando i relativi file sulla propria macchina.

[1] https://www.swego.it/
[2] https://instagantt.com/
[3] https://code.visualstudio.com/
[4] http://truffleframework.com/
[5] http://truffleframework.com/ganache/
[6] https://www.npmjs.com/

3 Processi di Supporto

3.1 Processo di documentazione

Durante lo svolgimento del capitolato si dovrà rendere conto, tramite una documentazione dettagliata, di tutti i processi che saranno coinvolti. Per questo motivo, il team suddividerà i documenti in:

- **Documenti interni**
  Tutti quei documenti che saranno visionati da fornitori e committenti;
- **Documenti esterni**
  Tutti quei documenti che saranno visionati anche dai proponenti.

3.1.1 Ciclo di vita di un documento

Un documento passerà attraverso tre stati:

- **In lavorazione**: si tratta della fase di stesura del documento e non è consultabile;
- **Da verificare**: dopo che il documento è stato ultimato, passerà nelle mani del *Verificatore* (il cui ruolo è descritto nel paragrafo 4.2.2.6), che dovrà esaminarlo;
- **Approvato**: dopo la verifica, il documento dovrà essere approvato definitivamente dal *Responsabile di Progetto*.

Ogni documento sarà identificato con un flag alla fine del nome, distanziato con un underscore, in base allo stato in cui si trova. Per il primo si userà $\_L$, per il secondo $\_V$, per il terzo $\_A$.

3.1.2 Documenti finali ad uso interno

3.1.2.1 Studio di fattibilità (SdF)

Lo *StudioDiFattibilità_v1.0.0* ha lo scopo di raccogliere le informazioni salienti dei capitolati proposti, esprimendone gli aspetti positivi e le potenziali criticità che sono emerse durante il confronto col gruppo.

3.1.2.2 Norme di progetto (NdP)

Le *NormeDiProgetto_v4.0.0* contengono le regole che il team utilizzerà durante lo sviluppo del progetto.

3.1.2.3 Verbale interno (VI)

Il *VerbaleInterno_v1.0.0* servirà al gruppo per documentare le discussioni e le decisioni prese durante le riunioni. La denominazione dovrà essere come segue:

verbale_Tipo del verbale_Numero del verbale_Data del verbale

dove:

- **Tipo del verbale**: specifica se Interno (I) o Esterno (E);
- **Numero del verbale**: numero univoco identificativo del verbale;
- **Data del verbale**: identifica la data in cui la riunione si è svolta. Si utilizzerà il formato:

  YYYY-MM-DD

Nella parte introduttiva del verbale verranno specificati:

- Data riunione;
- Ora inizio riunione;
- Ora fine riunione;
- Durata riunione;
- Luogo d'incontro;
- Oggetto di discussione;
- Moderatore;
- Segretario;
- Partecipanti.

3.1.3 Documenti finali ad uso esterno

3.1.3.1 Piano di Progetto (PdP)

Il *PianoDiProgetto_v4.0.0* contiene le indicazioni sulle scadenze temporali e fornisce un preventivo dei costi da presentare al proponente.
Vengono inoltre individuati i rischi ed analizzate le loro ricorrenze.
In questo documento vengono fatte emergere le *milestone*$_G$ legate ai punti critici e viene effettuata una *pianificazione*$_{\mathrm{G}}$ con l'uso di *diagrammi di Gantt*$_{\mathrm{G}}$.
In questo documento vengono fatte emergere le *milestone*$_G$ legate ai punti critici e viene effettuata una *pianificazione*$_{\mathrm{G}}$ con l'uso di diagrammi di *Gantt*$_{\mathrm{G}}$.

3.1.3.2 Piano di Qualifica (PdQ)

Il *PianoDiQualifica_v4.0.0* deve fornire ai membri del gruppo tutte le informazioni con cui poter soddisfare gli obiettivi di qualità.

3.1.3.3 Analisi dei Requisiti (AdR)

L'*AnalisiDeiRequisiti_v4.0.0* descrive gli *attori*$_{\mathrm{G}}$ del sistema, individua i *casi d'uso*$_{\mathrm{G}}$ a partire dai *requisiti*$_{\mathrm{G}}$ e fornisce una visione chiara ai *Progettisti* sul problema da trattare.

3.1.3.4 Glossario (G)

Nel documento *Glossario_v3.0.0* i termini tecnici, gli acronimi e le abbreviazioni sono definiti in modo chiaro e conciso, in modo tale da evitare ambiguità e massimizzare la comprensione dei documenti.

3.1.3.5 Manuale Utente (MU)

Il *ManualeUtente_v1.0.0* è un manuale pensato per aiutare l'utente ad utilizzare il prodotto, viene incrementato durante lo sviluppo di quest'ultimo. Deve avere un approccio incentrato sulle funzionalità che il prodotto offre.

3.1.3.6 Manuale Sviluppatore (MS)

Il *ManualeSviluppatore_v1.0.0* è un manuale per aiutare lo sviluppatore nella manutenzione e nell'incremento delle funzionalità del prodotto.

3.1.3.7 Verbale Esterno (VE)

Il *VerbaleEsterno_v1.0.0* è un documento in cui si tiene traccia delle discussioni del team con i committenti ed i proponenti. Come struttura ricalca quella del *VerbaleInterno_v1.0.0*.

3.1.4 Struttura del documento

3.1.4.1 Prima pagina La prima pagina di ogni documento sarà così strutturata:

- Logo;
- Nome del documento;
- Nome del gruppo - Nome del progetto;
- Email del gruppo;
- Informazioni sul documento:
  - Versione del documento;
  - Redazione;
  - *Verifica*$_{\text{G}}$;
  - Approvazione;
  - *Uso*$_{\text{G}}$;
  - Distribuzione.
- Descrizione del documento.

3.1.4.2 Registro delle modifiche

In seconda pagina il documento conterrà il registro delle modifiche che traccerà le modifiche apportate al documento. Sarà organizzato in una tabella che conterrà le seguenti colonne:

- Versione: indica la versione del documento;
- Data: indica la data in cui il documento è stato modificato;
- Descrizione: descrive la modifica effettuata nella relativa versione;
- Autore: indica il nome della persona che ha effettuato la modifica;
- Ruolo: indica il ruolo dell'autore.

3.1.4.3 Indice

Dopo il registro delle modifiche, il documento sarà correlato da un indice di tutte le sezioni. In alcuni documenti, se necessario, sarà aggiunto anche l'indice delle immagini, delle tabelle e dei riferimenti.

3.1.4.4 Formattazione generale della pagina

Ogni pagina del documento, fatta eccezione per la prima, conterrà una intestazione ed un pié di pagina.
L'intestazione presenterà a sinistra il logo e a destra l'email del gruppo.
Nel pié di pagina saranno presenti a sinistra il nome del documento susseguito dal nome del gruppo e, a destra il numero della pagina.

3.1.5 Norme tipografiche

Tutti i documenti dovranno sottostare alle seguenti norme tipografiche ed ortografiche.

3.1.5.1 Formati

- **Data**: il formato della data seguirà quello esplicato nell'*ISO*$_{G}$ *8601:2004*$_{G}$, quindi sarà:

  YYYY-MM-DD

  dove i simboli stanno per:

  – YYYY: anno;

  – MM: mese;

  – DD: giorno.

- **Orario**: ci si atterrà allo standard europeo delle 24 ore:

  hh:mm

  dove i simboli stanno per:

  – hh: ore;

  – mm: minuti.

3.1.5.2 Composizione del testo

- **Elenchi puntati**: ogni punto dell'elenco deve terminare con "*;*" tranne l'ultimo che deve terminare con ".". La prima parola deve iniziare con unla lettera maiuscola;
- **Glossario**: il pedice "$_{G}$" verrà utilizzato in corrispondenza di vocaboli presenti nel Glossario_v3.0.0.

3.1.5.3 Stili di testo

- **Grassetto**: il grassetto deve essere utilizzato per evidenziare parole particolarmente importanti negli elenchi puntati o nelle frasi;
- **Corsivo**: il corsivo deve essere utilizzato con i seguenti termini: situazioni:
  – Ruoli;
  – Documenti;
  – Stati del documento;
  – Citazioni;
  – Glossario;
  – Nomi di file.
- **Maiuscolo**: il maiuscolo deve essere utilizzato solamente per gli acronimi.

3.1.5.4 Sintassi

Per la stesura dei documenti i membri del team adotteranno la terza persona singolare.

3.1.5.5 Sigle

- **AdR**: Analisi dei Requisiti;
- **PdP**: Piano di Progetto;
- **NdP**: Norme di Progetto;
- **SdF**: Studio di Fattibilità;

- **PdQ**: Piano di Qualifica;
- **LdP**: Lettera di Presentazione;
- **G**: Glossario;
- **TB**: Technology Baseline;
- **PB**: Product Baseline;
- **MU**: Manuale Utente;
- **MS**: Manuale Sviluppatore;
- **RR**: Revisione dei Requisiti;
- **RP**: Revisione di Progettazione;
- **RQ**: Revisione di Qualifica;
- **RA**: Revisione di Accettazione.

3.1.6 Nome del file *.pdf*$_G$

Ogni documento sarà generato come file con estensione .pdf ed avrà un nome che rispetti la seguente convenzione:

NomeFile_versione.pdf

Il *NomeFile* seguirà la notazione CamelCase già descritta, invece la *versione* seguirà lo standard descritto qui di seguito.

3.1.7 Struttura dei file in LaTeX

Lo scheletro dei file necessari per generare un file .pdf di un documento è così strutturato:

- *sos.sty*: contiene tutti i package e le macro che possono trovare utilizzo in tutti i documenti. Contiene anche anche la macro che genera la copertina della prima pagina e che viene invocata dal file *main.tex*;
- *main.tex*: Include il template e tutte le sezioni che compongono il documento. Il file non è monolitico perché include le sezioni (che quindi sono dei file *.tex* che hanno vita propria), in questo modo si cerca di puntare all'assenza di conflitti durante modifiche contemporanee a più parti dello stesso documento e, di facilitare la revisione di singole parti del documento;
- *comandi.tex*: contiene delle macro specifiche per il documento che si sta creando;
- *"img" folder*: contiene il logo del gruppo e le altre immagini che sono incluse nel documento.

3.1.8 Versionamento

Il *versionamento*$_G$ permette a ciascun membro del team di condividere il lavoro nello spazio comune e di lavorare su vecchi e nuovi *Configuration Item*$_G$ senza rischio di sovrascritture accidentali.
Avrà questa forma:

vX.Y.Z

dove:

- **X**:

  - Inizia da 0;
  - Viene incrementato dal *Responsabile di Progetto* una volta approvato il documento.
- **Y**:
  - Inizia da 0;
  - Viene incrementato dal *Verificatore* una volta verificato il documento;
  - Quando viene incrementato, *Z* viene riportato a 0.
- **Z**:
  - Alla creazione parte da 1;
  - Viene incrementato dal redattore del documento ogni volta che questo è viene modificato.
  - Quando vengono incrementati X o Y viene riportato a 0.

3.1.9 Strumenti

3.1.9.1 LaTeX

Il team ha scelto di usufruire del linguaggio LaTeX per questi motivi:

- Permette un versionamento più semplice e compatibile con *Git*$_{\text{G}}$;
- Evita i conflitti che si possono creare usando software differenti (per esempio OpenOffice e Microsoft Word).

3.1.10 Gestione del *repository*$_{\text{G}}$

Per tenere traccia di tutte le modifiche apportate ai documenti ed al codice che sono stati prodotti, il team ha deciso di utilizzare *GitHub*, creando una *repository*$_{\text{G}}$ dedicata. Si seguiranno le norme descritte qui di seguito.

3.1.10.1 Struttura

- Il *master*$_{\text{G}}$ è annidato nella repository *origin*$_{\text{G}}$ e deve contenere solo il *template*$_{\text{G}}$ per la scrittura di tutti i documenti;
- Ogni *branch*$_{\text{G}}$ annidato sul master deve corrispondere solo alle *revisioni*$_{\text{G}}$;
- Ogni branch annidato in branch relativi alle revisioni contiene dei documenti che le riguardano;
- Ogni membro del gruppo potrà modificare i file di ogni documento o codice, o direttamente sul branch corrispondente oppure creandone uno parallelo a sua discrezione a meno di direttive diverse da parte dei *Progettisti*.

3.1.10.2 Tipi di file Tutti i file pertinenti verranno caricati nel repository e saranno sottoposti a versionamento. Verranno tuttavia ignorati, quindi inseriti all'interno del file *.gitignore*$_{\text{G}}$, tutti quelle quei file generati automaticamente durante le varie *build*$_{\text{G}}$ e che hanno estensioni indesiderate (ad esempio *.log*, *.out*).

3.1.10.3 Norme sui *commit*$_G$

I membri del team dovranno registrare le modifiche apportate alla repository tramite commit. Questo genererà una *pull request*$_G$ che il responsabile dovrà approvare prima di unire le modifiche al branch di riferimento.

3.2 Processo di verifica

Il processo di verifica deve essere continuo, questo serve ad evitare che eventuali errori arrivino fino alla fase di validazione. Questa attività, che viene svolta in corso d'opera, deve essere:

- Tempestiva, cioè il dato deve esserci quando serve;
- Accurata, devono essere evitate scorrettezze;
- Non intrusiva, non deve interrompere alcuna attività durante la sua esecuzione.

3.2.1 Qualità

Nel documento *PianoDiQualifica_v4.0.0* si devono descrivere le tecniche usate dai membri del gruppo con lo scopo di garantire la qualità del prodotto, in particolare:

- tutti gli standard adottati e che si occupano della qualità dei processi devono essere citati;
- devono essere presentati tutti i processi rilevanti di tali standard.

Inoltre per ogni processo si devono descrivere:

- gli obbiettivi di qualità che derivano da esso;
- le metodologie e le strategie considerate di utilità per raggiungerne gli obbiettivi;
- le metriche da utilizzare per poter effettuare delle misurazioni oggettive e qualitative.

3.2.1.1 Metriche di qualità

3.2.1.1.1 Metriche per il processo Per la valutazione di qualità dei processi si è deciso di utilizzare le seguenti metriche, che verranno utilizzate all'interno del *PianoDiQualifica_v4.0.0* per rendicontare le attività di verifica:

- **Schedule Variance - SV**: verrà utilizzata per valutare la pianificazione temporale prevista dal *PianoDiProgetto_v4.0.0* e per la valutazione dei rischi all'interno del Risk Management Process;
- **Struttura a 6 livelli definita da SPICE**: verrà utilizzata per valutare e migliorare la qualità del lavoro svolto;
- **Cost Variance - CV**: verrà utilizzata per verificare se i costi sono stati rispettati rispetto a quanto concordato nel *PianoDiProgetto_v4.0.0*.

3.2.1.1.2 Metriche per i documenti Per la valutazione di qualità dei documenti si è deciso di utilizzare le seguenti metriche, che verranno utilizzate all'interno del *PianoDiQualifica_v4.0.0* per rendicontare le attività di verifica:

- **Numero di errori ortografici**: verrà utilizzata per valutare la presenza di errori ortografici all'interno del documento verificato;

- **Indice Gulpease**: verrà utilizzata per valutare l'indice di leggibilità del testo prodotto;
- **Errori contenutistici:** verrà utilizzata per esprimere la correttezza di contenuto all'interno del documento;
- **Errori di forma:** verrà utilizzata per valutare quanto il prodotto sia conforme alle regole strutturali definite.

3.2.1.1.3 Metriche per il software Per la valutazione di qualità del software prodotto si è deciso di utilizzare le seguenti metriche, che verranno utilizzate all'interno del *PianoDiQualifica_v4.0.0* per rendicontare le attività di verifica:

- **Requisiti soddisfatti**: verrà utilizzata per valutare la funzionalità software attraverso una misurazione quantitativa dei requisiti soddisfatti;
- **Successo dei test**: verrà utilizzata per valutare il livello di affidabilità del prodotto software tramite il calcolo dei test aventi successo durante la fase di verifica;
- **Tempo di risposta**: verrà utilizzata per valutare l'efficienza del prodotto basandosi sul tempo medio che intercorrerà tra la richiesta di una funzionalità da parte dell'utente e la risposta del software;
- **Validazione di pagine web**: verrà utilizzata per valutare l'usabilità del prodotto finale.
- **Metriche per il codice**: verranno utilizzate per valutare la manutenibilità del codice, e la sua proprietà di essere evolvibile nel tempo attraverso correzioni, miglioramenti ed aggiunte;
- **Numero di browser supportati**: verrà utilizzata per valutare la portabilità del software prodotto, ossia la sua capacità di operare in ambienti$_G$ diversi, limitando la necessità di apportare cambiamenti.

3.2.1.2 Verifica dei documenti Il *Responsabile di Progetto* Ha il compito di dare inizio al processo di verifica della documentazione prodotta, assegnando i corrispettivi compiti ai *Verificatori*. Questi dovranno verificare che siano state rispettate in maniera rigida le seguenti regole, e in caso contrario segnalarlo per una successiva correzione:

- Utilizzo di una sintassi corretta e semplice;
- Utilizzo di brevi periodi, per evitare frasi troppo complesse;
- Un corretto annidamento della struttura del documento, evitando gerarchie troppo profonde o estese;
- Rispetto di tutte le norme precedentemente elencate in questo documento.

Per tutta la durata del progetto il team avrà inoltre l'obbligo di utilizzare il registro delle modifiche ogni qual volta deciderà di alterare il contenuto di un documento, al fine di tenere traccia dei cambiamenti effettuati e degli errori commessi. Un *Verificatore*, per un corretto svolgimento della sua attività, ha il compito di svolgere i task descritti nei paragrafi successivi secondo il seguente ordine:

- Attività manuale di verifica;
- Attività automatica di verifica;
- Resoconto delle attività di verifica.

3.2.1.2.1 Attività manuale di verifica I *Verificatori* avranno il compito di eseguire l'attività manuale di verifica sui documenti in maniera attenta e minuziosa, accompagnando l'attività stessa da un elenco contenente il resoconto degli errori riscontrati secondo tutte le metriche presentate nelle *NormeDiProgetto_v4.0.0* e discusse nel *PianoDiQualifica_v4.0.0*.

3.2.1.2.2 Attività automatica di verifica Per ottenere una seconda verifica sui documenti prodotti, andando a consolidare già la prima verifica manuale effettuata, si è deciso di svolgere una verifica automatizzata tramite l'utilizzo del correttore automatico dell'editor di testo *TeXstudio*, al fine di individuare eventuali errori tralasciati per motivi di distrazione.

3.2.1.2.3 Resoconto dell'attività di verifica Al termine dello svolgimento dell'attività di verifica, il *Verificatore* è tenuto a consegnare al *Responsabile di Progetto*, il resoconto finale contenente tutti gli errori riscontrati durante lo svolgimento dell'attività, in modo che possano venire effettuate le opportune correzioni e discussi eventuali cambiamenti da entrambi i membri. Al termine della correzione dovrà quindi essere aggiornato il registro delle modifiche per tenere traccia dei cambiamenti effettuati e degli errori precedentemente commessi. Tutti gli errori riscontrati durante l'ultima attività di verifica, verranno inoltre impiegati in seguito per calcolare le metriche riguardanti gli obiettivi di qualità prefissati per i documenti redatti, e pubblicati nelle appendici del *PianoDiQualifica_v4.0.0*.

3.2.2 Analisi

3.2.2.1 Analisi statica

Studia il codice sorgente e la documentazione e controlla che essi seguano le norme. Non richiede l'esecuzione del prodotto software in nessuna sua parte, ma si limita all'osservazione. Questo processo può essere visto come una verifica dinamica del comportamento del programma su un insieme finito di casi selezionati nel dominio di tutte le esecuzioni possibili. Ciascun caso di prova, specifica i valori in ingresso e lo stato iniziale del sistema e deve produrre un esito decidibile, verificato rispetto ad un comportamento atteso.
Si può declinare in due maniere:

- ***Walkthrough***$_{\text{G}}$
  Questa tecnica di analisi deve essere effettuata nella fase iniziale dello sviluppo del prodotto per trovare eventuali errori che possano essere riscontrati. Non sapendo che tipo di errori cercare, si effettua una lettura a largo spettro. Quando i *Verificatori* avranno stilato una *checklist*$_{\text{G}}$ di tutti gli errori più comuni, si passerà alla fase di *Inspection*$_{\text{G}}$. Essendo un'analisi continua, la checklist tenderà ad aumentare costantemente di dimensione, rendendo più efficace ed efficiente l'Inspection.
- **Inspection**
  Basandosi sulla checklist redatta durante la Walkthrough, i *Verificatori* analizzeranno tutto il prodotto cercando, di volta in volta e in modo mirato, tutti gli errori ricorrenti in essa contenuti.

3.2.2.2 Analisi dinamica

L'analisi dinamica, al contrario di quella statica, richiede l'esecuzione del codice. Viene effettuata tramite test ed è coinvolta sia nel processo di verifica che in quello di validazione. I test verranno descritti in un capitolo a parte.

3.2.3 Test

Il testing è una parte essenziale del processo di verifica: produce una misura della qualità del sistema aumentandone il valore, identificandone e rimuovendone i difetti. Il suo inizio

non va differito al termine delle attività di codifica e le sue esigenze devono essere tenute in conto nella progettazione del sistema. Tutti i test devono essere rieseguibili e devono sempre produrre lo stesso esito. Devono essere eseguiti in condizioni controllate, diventando così deterministici.
Di seguito vengono descritte le tipologie di test, assieme alla sintassi relativa al modo in cui sono identificati all'interno del *PianoDiQualifica_v4.0.0*.

3.2.3.1 Test di unità (TU)
Il loro obiettivo è quello di verificare la correttezza del codice *as implemented*, ovvero puntando a controllare il codice in maniera microscopica.
La responsabilità della loro realizzazione è del *Programmatore* per le unità più semplici, e di un *Verificatore* indipendente per le altre.
Le risorse consumate da questo tipo di test sono molto poche perché, sono coinvolte piccole parti di codice che hanno basso accoppiamento le une con le altre. I test di unità sono classificati come segue:

TU[codice identificativo]

dove il codice identificativo è numerico, incrementale (iniziando da 1) ed univoco per ogni test.

3.2.3.2 Test di integrazione (TI)
Questi test verificano non solo il corretto comportamento di ogni singolo oggetto, ma anche le relazioni con gli altri componenti dell'applicazione.
Possono rilevare i seguenti problemi:

- Errori residui nella realizzazione dei componenti;
- Modifica delle interfacce o cambiamenti nei requisiti;
- Riuso di componenti dal comportamento oscuro o inadatto;
- Integrazione con altre applicazioni non bene conosciute.

Un test di questo tipo consuma molte risorse dato che, la grandezza del codice da controllare è significativa ed il grado di accoppiamento tra le varie parti da testare è massimo. L'utilizzo di un test di questo tipo deve quindi essere ponderato e, in generale, da non preferire a quello di unità. I test di integrazione sono classificati come segue:

TI[codice identificativo]

dove il codice identificativo è numerico, incrementale (iniziando da 1) ed univoco per ogni test.

3.2.3.3 Test di sistema (TS)
I test di sistema possono essere visti come un'attività interna del fornitore per accertare la copertura dei requisiti software.
Questo tipo di test richiede molte risorse e, in generale, non va utilizzato per testare unità singole. I test di sistema sono classificati come segue:

TS[codice requisito]

dove il codice requisito identifica il codice univoco associato ad ogni requisito descritto nel documento *AnalisiDeiRequisiti_v4.0.0*.

3.2.3.4 Test di regressione (TR)
I test di regressione sono l'insieme dei TU e TI necessari ad accertare che la modifica di una parte del prodotto non causi errori nelle altre parti che hanno relazioni con essa.

Un test di questo tipo consuma tante più risorse quanto la parte modificata è accoppiata con le altre, anche perché comporta la ripetizione di test già previsti ed effettuati per ogni parte che non è stata modificata.

3.2.3.5 Test di validazione (TV)

Il test di validazione è un'attività supervisionata dal committente e dal proponente come dimostrazione di conformità del prodotto sulla base di casi di prova specificati o implicati dal contratto. Alla validazione segue il rilascio del prodotto con eventuale garanzia e la fine della *commessa*$_{\text{G}}$, con eventuale manutenzione. I test di validazione sono classificati come segue:

TV[codice requisito]

dove il codice requisito identifica il codice univoco associato ad ogni requisito descritto nel documento *AnalisiDeiRequisiti_v4.0.0*.

3.2.4 Strumenti

3.2.4.1 Strumenti per l'analisi statica

- **TexStudio**$_{\text{G}}$[7]
  Questo editor include un correttore ortografico automatico che sarà utilizzato per verificare la corretta sintassi dei manuali in lingua inglese da fornire alla Red Babel. Tuttavia tale strumento non è preciso nell'individuazione degli errori più sottili, quindi sarà comunque necessaria un'analisi più approfondita da parte dei *Verificatori*;
- **Indice *Gulpease***$_{\text{G}}$
  Per i test di leggibilità il team ricorrerà al calcolo dell'indice Gulpease;
- ***W3C Markup Validation Service***$_{\text{G}}$[8]
  È uno strumento per la validazione rispetto agli standard dei documenti *HTML*$_{\text{G}}$ e *xHTML*$_{\text{G}}$;
- **Microsoft Excel**$_{\text{G}}$
  Tale programma verrà utilizzato per calcolare alcune metriche di qualità e fornirne i risultati in un formato grafico nel *PianoDiQualifica_v4.0.0*.

3.3 Validazione

Il processo di validazione ha i seguenti obiettivi:

- verificare che il prodotto finale sia conforme con quanto specificato;
- verificare che il prodotto finale sia in grado di minimizzare gli effetti degli errori commessi.

3.3.1 Ruoli

I ruoli per il processo di validazione saranno divisi nel seguente modo:

- i *Verificatori* avranno il compito di eseguire i test, tracciando i risultati e riportandone la valutazione;

[7] https://www.texstudio.org/
[8] https://validator.w3.org/

- il *Responsabile di Progetto* avrà il compito di revisionare i risultati dei test effettuati, e decidere se convalidarli o ripeterli. Egli sarà inoltre incaricato di fornire al committente i risultati dei test effettuati, accompagnati da una documentazione per poterli svolgere in maniera indipendente.

3.3.2 Procedure di validazione

L'intera procedura di validazione dovrà quindi comprendere i seguenti punti:

- La verifica da parte dei *Verificatori* sul prodotto finale, ed il tracciamento dei risultati ottenuti;
- L'analisi dei risultati da parte del *Responsabile di Progetto*, con verdetto finale di accettazione o ripetizione dei test;
- La consegna da parte del *Responsabile di Progetto* dei risultati accettati al proponente, informandolo dettagliatamente sulle modalità di validazione.

4 Processi organizzativi

4.1 Processo di coordinamento

4.1.1 Comunicazioni

4.1.1.1 Comunicazioni interne Per gestire le comunicazini interne si è riscontrata la necessità di distinguere la modalità di presentazione delle informazioni in:

- **Comunicazione formale**: questa modalità viene usata per le discussioni ufficiali ed inerenti a gestione e sviluppo del progetto;
- **Comunicazione informale**: questa modalità viene usata per scambiare informazioni non ufficiali e discutere riguardo le attività di progetto.

Si è deciso di utilizzare i seguenti strumenti:

- ***Telegram***$_{\mathrm{G}}$;
- ***Slack***$_{\mathrm{G}}$.

4.1.1.2 Comunicazioni esterne

- **Email**: è stato creato l'indirizzo di posta elettronica: **sonsofswe.swe@gmail.com**

  L'utilizzo di tale casella di posta è affidato unicamente al *Responsabile di Progetto*, per la gestione delle relazioni con il committente ed il proponente del progetto.

  Le email dovranno avere la seguente forma:

  - **Oggetto**: l'oggetto dev'essere chiaro e conciso, per riconoscere e distinguere facilmente tra loro le email;
  - **Apertura**: ogni email deve iniziare con un aggettivo di circostanza seguito dal titolo e dal nome del destinatario, oppure con un saluto formale e terminare con una virgola;
  - **Corpo**: nel corpo, che deve essere breve ed esaustivo, verranno spiegate le ragioni per cui si sta scrivendo l'email;
  - **Allegati**: è possibile l'invio di allegati, su richiesta del proponente o del committente;

  - **Chiusura**: la chiusura deve essere separata dal corpo con il doppio ritorno a capo e prevede un congedo formale e la firma del mittente.

- **Slack**: verrà utilizzato, su richiesta del committente, per lo scambio di informazioni in maniera ufficiosa.

4.1.2 Riunioni

4.1.2.1 Obiettivi

Le riunioni sono di fondamentale importanza per la gestione di un progetto. È indispensabile che i membri del team abbiano modo di confrontarsi tra loro al fine di risolvere i problemi esistenti e generare nuove idee. Una riunione è inoltre un ottimo mezzo per creare armonia e consolidare i rapporti all'interno del gruppo. Le riunioni con il proponente/i o committente/i avranno invece lo scopo di condividere gli obiettivi raggiunti e di confrontarsi nel caso in cui si presenti la necessità di colmare lacune.

4.1.2.2 Riunioni interne

Per un produttivo svolgimento delle riunioni interne, verranno rispettati i seguenti ruoli:

- **Moderatore**: il *Responsabile di Progetto* ha il dovere di convocare il team alle riunioni interne quando necessario per un confronto tra i vari membri; durante gli incontri è richiesto che tutti i membri siano presenti, salvo eccezioni precedentemente segnalate. Il *Responsabile di Progetto* avrà quindi l'obbligo di seguire l'ordine del giorno riguardo agli argomenti da affrontare e di trovare un consenso unanime riguardo le decisioni da prendere. Per svolgere il prorio compito in maniera adeguata, il moderatore è tenuto ad avere un atteggiamento autorevole, flessibile e diligente;
- **Segretario**$_{\text{G}}$: il *Segretario* ha il compito di stendere una prima minuta dell'incontro, controllare che venga seguito punto per punto l'ordine del giorno e redigere la versione finale del *verbale*$_{\text{G}}$. Conclusa una riunione, il *Segretario* avrà inoltre il compito di fornire una copia del verbale ad ogni membro del team, comprendente un resoconto delle decisioni prese e degli obiettivi prefissati;
- **Partecipanti**: i partecipanti sono tenuti a presenziare puntualmente qualora venga convocata una riunione da parte del *Responsabile di Progetto*. Nel caso in cui un membro sia impossibilitato a partecipare è invitato a comunicarlo tempestivamente al *Responsabile di Progetto*, al fine di accordarsi sul come procedere. È richiesto ai partecipanti di tenere un comportamento diligente e responsabile per un corretto svolgimento dell'assemblea.

4.1.2.2.1 Descrizione

Le riunioni dovranno avere cadenza settimanale, per fare il punto della situazione riguardo gli obettivi prefissati e le difficoltà incontrate. Il *Responsabile di Progetto*, dovrà informare i vari membri della riunione tramite email e questi saranno tenuti a rispondere confermando la presenza o motivando l'assenza. Una riunione verrà effettivamente considerata valida solo nel caso in cui siano presenti almeno la metà dei membri convocati. Nel caso contrario, sarà proposta una seconda data all'interno della settimana per ripetere e validare l'incontro. Le riunioni dovranno concentrarsi su quattro punti cardine:

- Informare i membri riguardo le novità;
- Valutare il lavoro precedentemente fatto;
- Prendere decisioni collettive riguardo eventuali problematiche;
- Progettare il percorso per raggiungere gli obiettivi prefissati entro le tempistiche stabilite.

Al termine di ogni riunione verrà quindi redatto il corrispondente verbale ed inviato per mail a tutti i membri del team.

4.1.2.3 Riunioni esterne

4.1.2.3.1 Descrizione Le riunioni con il proponente hanno la funzione di supportare il gruppo di progetto e di consolidare il rapporto tra entrambe le parti. Durante le riunioni, il gruppo avrà l'impegno di condividere gli obiettivi raggiunti e le eventuali problematiche incontrate. Il *Responsabile di Progetto* avrà il dovere di convocare le riunioni quando necessario, anche confrontandosi in base alle esigenze dei membri del team. Il segretario scelto invece avrà il compito di verbalizzare quanto emerso dalla discussione, e le possibili richieste di cambiamento e/o correzione emesse dal proponente.

4.2 Processo di pianificazione

4.2.1 Descrizione

Lo sviluppo di un progetto prevede la cooperazione di diversi ruoli. Per una comprensione unanime del progetto, ogni membro del team dovrà esercitare obbligatoriamente tutti i ruoli previsti, durante periodi temporali differenti, che saranno di seguito elencati. Nel caso in cui sorga la necessità, un membro potrà ricoprire più di un ruolo contemporaneamente, ma solo nel caso in cui non si tratti di redattore e verificatore. Questo caso in particolare è considerato un controsenso, in quanto la figura del verificatore risulterebbe corrotta nell'analisi dei propri documenti. Inoltre, ogni ruolo avrà incarichi diversi, i quali verranno gestiti ed assegnati dal *Responsabile di Progetto* con l'ausilio di opportuni strumenti, in modo da garantire monitoraggio, controllo e revisione del progetto per tutta la sua durata.

4.2.2 Ruoli

4.2.2.1 Responsabile Il *Responsabile di Progetto* è colui che detiene la responsabilità sul lavoro svolto dal team, mantiene i contatti esterni e presenta al committente il progetto finale. Egli possiede il potere decisionale e si fa carico dei seguenti oneri:

- Pianificazione, coordinamento e controllo delle attività;
- Gestione e controllo delle risorse;
- Analisi e gestione dei rischi;
- Approvazione della documentazione;
- Contatti con gli enti esterni.

Di conseguenza il *Responsabile di Progetto* ha il compito di assicurarsi che le attività di verifica e validazione vengano svolte in riferimento alle *NormeDiProgetto_v4.0.0*, di redigere l'*Organigramma*$_G$, di garantire che vengano rispettati i ruoli assegnati all'interno del *PianoDiProgetto_v4.0.0*, e di garantire che non vi siano conflitti tra redattori e verificatori.

4.2.2.2 Analista L'*Analista* si occupa di capire il problema da affrontare, ascoltando le richieste del committente; è un individuo esperto, in grado di capire il dominio e la complessità del problema. Le sue mansioni principali sono:

- Analizzare le qualità e i servizi che dovrà offrire il prodotto finale;
- Valutare la fattibilità del progetto, riportando tali valutazioni nel *StudioDiFattibilità_v1.0.0*;

- Redigere l'*AnalisiDeiRequisiti_v4.0.0*, in cui verranno indicati tutti i requisiti del progetto individuati.

4.2.2.3 Amministratore L'*Amministratore* è responsabile della gestione dell'ambiente di lavoro; deve offrire al gruppo più facilitazioni e automazioni possibili al fine di incrementare l'operatività e l'efficienza. I suoi doveri sono:

- Gestione della documentazione di progetto;
- Controllo di versioni e configurazioni;
- Ricerca e gestione di strumenti di supporto che facilitino l'operato del gruppo;
- Redazione delle *NormeDiProgetto_v4.0.0* e supporto alla redazione del *PianoDiProgetto_v4.0.0*.

4.2.2.4 Progettista Il *Progettista* ha il compito di gestire la progettazione vera e propria, sfruttando le sue competenze e conoscenze in ambiti tecnico e tecnologico; il suo ruolo è direttamente collegato a quello dell'*Analista*, in quanto deve capire e spiegare come risolvere i problemi identificati precedentemente dagli *Analisti*. Ha il compito di:

- Influenzare le scelte tecniche e tecnologiche, per:
  - Orientare il gruppo all'utilizzo di strumenti e servizi il più possibile efficienti;
  - Effettuare scelte progettuali atte a garantire la manutenibilità e la modularità del prodotto finale.
- Sviluppare l'architettura del prodotto software e i relativi documenti informativi e redigere la parte programmatica del *PianoDiQualifica_v4.0.0*.

4.2.2.5 Programmatore Il *Programmatore* avrà il compito di codificare e mantenere il codice prodotto, implementando le soluzioni proposte dal *Progettista*: deve perciò avere alte competenze tecniche. I suoi compiti sono:

- Implementare in maniera rigorosa quanto richiesto dai *Progettisti*;
- Implementare componenti aggiuntive allo scopo di creare strumenti di test e verifica del prodotto;
- Redigere eventuali *ManualeUtente_v1.0.0* e *ManualeSviluppatore_v1.0.0*.

4.2.2.6 Verificatore Il *Verificatore* sarà responsabile della verifica, vale a dire del continuo controllo del prodotto e della documentazione: sarà attivo per tutta la durata del progetto ed agirà sfruttando le proprie capacità di giudizio, esperienza e competenza. I suoi compiti sono:

- Accertarsi del rispetto delle *NormeDiProgetto_v4.0.0* e della conformità rispetto al *PianoDiQualifica_v4.0.0*;
- Redigere il *PianoDiQualifica_v4.0.0*.

4.2.3 Ticketing

Al fine di permettere una migliore gestione del lavoro interno al gruppo, verrà utilizzato lo strumento **Asana**, utile a creare ed assegnare $Task_{\mathrm{G}}$; in questo modo il *Responsabile di Progetto* sarà in grado di tenere costantemente sotto controllo l'avanzamento delle attività di progetto. Complementariamente verrà usato **InstaGantt**, strumento che permette di creare diagrammi di Gantt basandosi su ciò che viene dichiarato in Asana.

4.2.3.1 Procedura di assegnazione

L'assegnazione di un Task coinciderà con la sua creazione e si basa sulla seguente sequenza di passi:

- Assegnazione di un titolo al task;
- Assegnazione del task stesso al/ai membro/i designato/i;
- Aggiunta di una breve ma concisa descrizione del compito e dei suoi obiettivi finali;
- Inserimento delle date di inizio e fine;
- Impostazione dello stato del ticket ad "Aperto".

4.2.3.2 Possibile stato di un ticket

Un ticket può essere nei seguenti stati:

- Aperto;
- In elaborazione;
- Sospeso;
- Completato/Risolto.

4.3 Processo dell'infrastruttura

4.3.1 Ambienti di sviluppo

Ogni componente del gruppo avrà la possibilità di utilizzare il sistema operativo che più gli aggrada, a patto che i servizi offerti siano gli stessi. In particolare, verranno utilizzati:

- Windows 10 Pro x64;
- Windows 10 Home x64;
- Windows 10 Education x64;
- Ubuntu 17.04 x64;
- Ubuntu Mate 16.04 x64;
- OSX El Capitan versione 10.11.6.

4.3.2 Strumenti

Gli strumenti utilizzati durante la fase dei processi organizzativi sono:

- **Telegram** [9]
  Telegram è un servizio di messaggistica istantanea multipiattaforma;
- **Slack** [10]
  Slack è un servizio di comunicazione multipiattaforma pensato per facilitare il lavoro di gruppo tramite canali distinti. Fornisce inoltre la possibiltà di integrare servizi esterni, come Github e Google Drive;
- **Google Drive** [11]
  Google Drive è un servizio di memorizzazione ed archiviazione online fornito da Google

[9] https://telegram.org/
[10] https://slack.com/
[11] https://www.google.com/drive/

basato sul *Cloud*$_G$. Esso verrà utilizzato dal gruppo per lo scambio di documenti ed informazioni di supporto allo sviluppo del progetto ma non soggetti al versionamento;

- **Fogli Google** [12]
  Fogli Google è un servizio offerto da Google per la creazione di fogli di lavoro online modificabili in contemporanea da chiunque ne abbia accesso. Il gruppo utilizzerà questo strumento per il rendiconto delle ore di lavoro;
- **Github** [13]
  Github è un' implementazione dello strumento di controllo di versione Git ed offre un servizio di *hosting*$_G$ per progetti software. Il gruppo utilizzerà un repository comune denominato *Marvin* avente come proprietario il profilo *SOS-SonsOfSwe* le cui credenziali, saranno rese note a tutti i componenti; ognuno potrà inoltre apportare le proprie modifiche direttamente dal proprio account personale;
- **Asana** [14]
  Asana è un servizio web utile a migliorare la collaborazione all'interno di un gruppo di lavoro, dando la possibilità di gestire i task di ogni componente del team online. In esso è possibile:
  - Suddividere il progetto sezioni, in modo da dividere gli ambiti di lavoro;
  - Suddividere le sezioni in task veri e propri;
  - Ipostare per ogni task scadenze diverse;
  - Impostare dipendenze tra le parti, in modo che un task non possa essere completato se uno o più task, da cui *dipende*, non sono stati completati in precedenza;
  - Suddividere i task in sotto-task, qualora il compito risulti troppo complesso e necessiti dunque un'ulteriore modularizzazione;
  - Avere un calendario delle scadenze sempre aggiornato;
  - Avere una completa visione della distribuzione del carico di lavoro all'interno del team.

  Per gruppi di studenti, Asana offre la propria versione premium gratuitamente.

Registro delle modifiche

Versione	Data	Descrizione	Autore	Ruolo
3.0.0	2018-05-07	Approvazione	Eleonora Thiella	*Responsabile di Progetto*
2.1.0	2018-05-06	Verifica	Stefano Panozzo	*Verificatore*
2.0.3	2018-05-06	Aggiornata la lista degli strumenti utilizzati	Andrea Favero	*Analista*
2.0.2	2018-05-05	Aggiunta del *processo di validazione*	Giovanni Dalla Riva	*Amministratore*
2.0.1	2018-05-04	Incremento dei *processi di supporto*	Andrea Favero	*Amministratore*
2.0.0	2018-04-28	Approvazione	Giovanni Dalla Riva	*Responsabile di Progetto*
1.1.0	2018-04-28	Verifica	Giovanni Cavallin	*Verificatore*

[12] https://www.google.it/intl/it/sheets/about/
[13] https://github.com/
[14] https://asana.com/

Versione	Data	Descrizione	Autore	Ruolo
1.0.1	2018-04-27	Revisione correttiva dei contenuti di seguito alle segnalazioni del committente	Federico Caldart	*Amministratore*
1.0.0	2018-03-12	Approvazione	Stefano Panozzo	*Responsabile di Progetto*
0.1.0	2018-03-12	Verifica	Andrea Favero	*Verificatore*
0.0.10	2018-03-09	Completata la stesura dei *processi organizzativi*	Lorenzo Menegon	*Amministratore*
0.0.9	2018-03-09	Completata la stesura dei *processi di supporto*	Giovanni Cavallin	*Amministratore*
0.0.8	2018-03-08	Completata le stesura della sezione sui *processi primari*	Eleonora Thiella	*Amministratore*
0.0.7	2018-03-20	Correzione per una formattazione standard per tutte le sezioni	Eleonora Thiella	*Amministratore*
0.0.6	2018-03-08	Correzione punteggiatura e ortografia per tutte le sezioni	Giovanni Cavallin	*Amministratore*
0.0.5	2018-03-07	Aggiornati strumenti per la sezione *processi organizzativi*	Federico Caldart	*Amministratore*
0.0.4	2018-03-05	Creata la sezione riguardante i *processi organizzativi*	Federico Caldart	*Amministratore*
0.0.3	2018-03-04	Creata la sezione riguardante i *processi di supporto*	Giovanni Cavallin	*Amministratore*
0.0.2	2018-03-04	Creata sezione riguardante i *processi primari*	Eleonora Thiella	*Amministratore*
0.0.1	2018-03-03	Creato lo scheletro del documento e la sezione *Introduzione*	Giovanni Cavallin	*Amministratore*