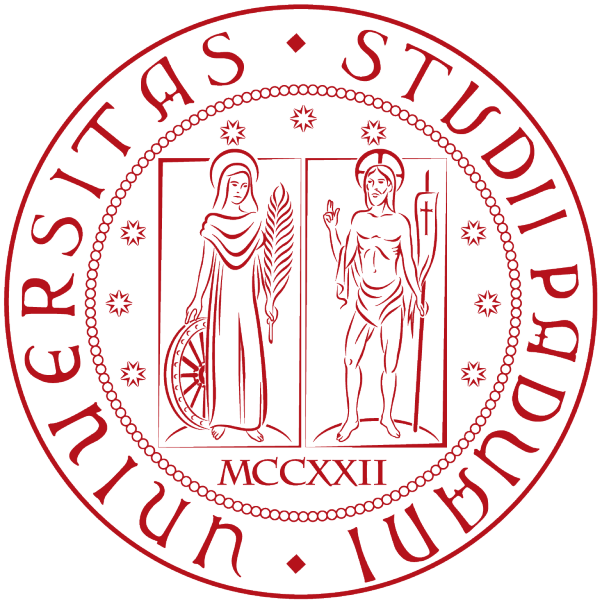

Norme di Progetto

SonsOfSwe - Progetto Speect

sonsofswe.swe@gmail.com

Informazioni sul documento

Versione	1.0
Redazione	Caldart Federico Cavallin Giovanni Dalla Riva Giovanni Favero Andrea Menegon Lorenzo Panozzo Stefano Thiella Eleonora
Verifica	Caldart Federico
Approvazione	Cavallin Giovanni
Uso	interno
Distribuzione	Vardanega Tullio

Descrizione

Questo documento descrive le regole, gli strumenti e le convenzioni adottate dal gruppo SonsOfSwe durante la realizzazione del progetto Marvin.

Indice

1 Introduzione **4**
1.1 Scopo del documento 4
1.2 Scopo del prodotto 4
1.3 Glossario 4
1.4 Riferimenti 4
1.4.1 Normativi 4
1.4.2 Informativi 5

2 Processi primari **5**
2.1 Processo di fornitura 5
2.1.1 Studio di Fattibilità 5
2.1.2 Rapporti con il proponente 5
2.1.3 Documentazione fornita 6
2.1.4 Collaudo e consegna del prodotto 6
2.2 Processo di sviluppo 6
2.2.1 Attività 6
2.2.1.1 Analisi dei Requisiti 6
2.2.1.2 Progettazione 6
2.2.1.2.1 Linee guida per la progettazione 6
2.2.1.2.2 UML 7
2.2.1.3 Codifica 7
2.2.1.3.1 Nomi 7
2.2.1.3.2 Commenti 7
2.2.1.3.3 Formattazione 7
2.2.2 Strumenti 8

3 Processi di Supporto **9**
3.1 Processo di documentazione 9
3.1.1 Processo di garanzia della qualità 9
3.1.2 Ciclo di vita di un documento 9
3.1.3 Documenti finali ad uso interno 9
3.1.3.1 Studio di fattibilità (SdF) 9
3.1.3.2 Norme di progetto (NdP) 9
3.1.3.3 Verbale interno (VI) 9
3.1.4 Documenti finali ad uso esterno 10
3.1.4.1 Piano di Progetto (PdP) 10
3.1.4.2 Piano di Qualifica (PdQ) 10
3.1.4.3 Analisi dei Requisiti (AdR) 10
3.1.4.4 Specifica Tecnica (ST) 10
3.1.4.5 Definizione di Prodotto (DdP) 10
3.1.4.6 Glossario (G) 10
3.1.4.7 Manuale Utente (MU) 10
3.1.4.8 Manuale Manutentore (MM) 11
3.1.4.9 Verbale Esterno 11
3.1.5 Struttura del documento 11
3.1.5.1 Prima pagina 11
3.1.5.2 Diario delle modifiche 11
3.1.5.3 Indice 11
3.1.5.4 Formattazione generale della pagina 11
3.1.6 Norme tipografiche 12
3.1.6.1 Formati 12
3.1.6.2 Composizione del testo 12
3.1.6.3 Stili di testo 12
3.1.6.4 Sintassi 13

3.1.6.5 Sigle . . . 13
3.1.7 Composizione email . . . 13
3.1.8 Componenti grafiche . . . 14
3.1.9 Nome del file *.pdf*$_G$ . . . 14
3.1.10 Struttura dei file in LaTeX . . . 14
3.1.11 Versionamento . . . 14
3.1.12 Strumenti . . . 15
3.1.12.1 ***LaTeX*** . . . 15
3.1.12.2 Da aggiungere qui eventuali altri strumenti che utilizzeremo per altro . . . 15
3.1.13 Gestione del *repository*$_G$ . . . 15
3.1.13.1 Struttura . . . 15
3.1.13.2 Tipi di file . . . 15
3.1.13.3 Norme sui *commit*$_G$ . . . 15
3.2 Processo di verifica . . . 15
3.2.1 Analisi . . . 16
3.2.1.1 Analisi statica . . . 16
3.2.1.2 Analisi dinamica . . . 16
3.2.2 Test . . . 16
3.2.2.1 **Test di *unità*$_G$ (TU)** . . . 16
3.2.2.2 **Test di integrazione (TI)** . . . 16
3.2.2.3 **Test di sistema (TS)** . . . 17
3.2.2.4 **Test di *regressione*$_G$ (TR)** . . . 17
3.2.2.5 **Test di validazione** . . . 17
3.2.3 Strumenti . . . 17
3.2.3.1 Strumenti per l'analisi statica . . . 17
3.2.3.2 Strumenti per l'analisi dinamica . . . 18

4 Processi organizzativi . . . 19
4.1 Processo di coordinamento . . . 19
4.1.1 Comunicazioni . . . 19
4.1.1.1 Comunicazioni interne . . . 19
4.1.1.2 Comunicazioni esterne . . . 19
4.1.2 Riunioni . . . 19
4.1.2.1 Obiettivi . . . 19
4.1.2.2 Riunioni interne . . . 19
4.1.2.2.1 Descrizione . . . 20
4.1.2.3 Riunioni esterne . . . 20
4.1.2.3.1 Descrizione . . . 20
4.2 Processo di pianificazione . . . 20
4.2.1 Descrizione . . . 20
4.2.2 Ruoli . . . 21
4.2.2.1 Responsabile . . . 21
4.2.2.2 Analista . . . 21
4.2.2.3 Amministratore . . . 21
4.2.2.4 Progettista . . . 21
4.2.2.5 Programmatore . . . 21
4.2.2.6 Verificatore . . . 22
4.2.3 Ticketing . . . 22
4.2.3.1 Procedura di assegnazione . . . 22
4.2.3.1.1 Possibile stato di un ticket . . . 22
4.2.3.1.2 aggiungere diagramma uml??? . . . 22
4.3 Processo dell'infrastruttura . . . 22
4.3.1 Ambienti di sviluppo . . . 22
4.3.1.1 Sistemi operativi . . . 22
4.3.2 Strumenti . . . 23
4.3.2.1 Telegram . . . 23

4.3.2.2 Slack 23
4.3.2.3 Google Drive 23
4.3.2.4 Fogli Google 23
4.3.2.5 Github 23
4.3.2.6 Asana 23
4.3.2.7 Instagantt 24

1 Introduzione

1.1 Scopo del documento

In questo documento verranno trattate le norme interne alle quali i membri di *SonsOfSwe* dovranno obbligatoriamente sottostare. Ogni membro dovrà visionare il documento e seguirne le regole in esso contenute, per ottenere la massima *efficienza*$_G$ ed *efficacia*$_G$, mantenendo un certo grado di uniformità.
In questo documento verranno esposte le norme riguardanti:

- L'identificazione dei ruoli e delle relative mansioni che essi dovranno svolgere;
- Le modalità di lavoro durante le fasi di *progetto*$_G$;
- Le interazioni tra i membri del gruppo e con le entità esterne;
- L'organizzazione e la cooperazione all'interno del *team*$_G$;
- La modalità e le regole adottate per la stesura dei documenti;
- La definizione degli ambienti di sviluppo.

1.2 Scopo del prodotto

Lo scopo del *capitolato*$_G$ *Marvin* è di realizzare un *prototipo*$_G$ di Uniweb come *ĐApp*$_G$ che giri su *Ethereum*$_G$.
I tre attori che si rapportano con Marvin sono:

1. Università;
2. Professori;
3. Studenti.

Il portale deve quindi permettere agli studenti di accedere alle informazioni riguardanti le loro carriere universitarie, di iscriversi a esami, accettare o rifiutare voti e devono poter vedere il loro libretto universitario.
Ai professori deve invece essere permesso di registrare i voti degli studenti.
L'università ogni anno crea una serie di corsi di laurea rivolti a studenti, dove ognuno di essi comprende un elenco di esami disponibili per anno accademico. Ogni esame ha un argomento, un numero di crediti e un professore associato.
Gli studenti si iscrivono ad un corso di laurea e tramite il libretto mantengono traccia ufficiale del progresso.

1.3 Glossario

Nel documento Glossario 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.
I vocaboli presenti in esso saranno posti in corsivo e presenteranno una "G" maiuscola a pedice.

1.4 Riferimenti

1.4.1 Normativi

- ***ISO***$_G$ **31-0**: http://en.wikipedia.org/wiki/ISOwiki/ISO_31;
- ***ISO***$_G$ **8601**: http://it.wikipedia.org/wiki/ISO_8601;
- ***ISO***$_G$ **12207-1995**: http://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf.

1.4.2 Informativi

- *Piano di Progetto v0.0.1*;
- *Piano di Qualifica v0.0.1*;
- Amministrazione di progetto: http://www.math.unipd.it/~tullio/IS-1/2014/Dispense/P06.pdf;
- *Specifiche UTF-8*$_G$: http://www.unicode.org/versions/Unicode6.1.0/ch03.pdf;
- INSERIRE QUI TUTTI I RIFERIMENTI CHE USEREMO DURANTE IL PROGETTO

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* avrà riunito il team e discusso con esso di tutti i capitolati, gli *Analisti* avranno il compito di stilare lo *Studio di Fattibilità* seguendo le considerazioni emerse.
Lo *Studio di Fattibilità* sarà organizzato come segue:

- **Informazioni sul capitolato**: Vengono ricordati il nome del 'emphprogetto$_G$, il *proponente*$_G$ e i *committenti*$_G$;
- **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 il *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 Documentazione fornita

Al *Red Babel* e ai Prof. Tullio Vardanega e Prof. Riccardo Cardin verranno forniti i seguenti documenti:

- ***Analisi dei Requisiti***: contiene l'analisi dei casi d'uso e dei requisiti;
- ***Piano di Qualifica***: contiene l'attività di verifica, di validazione e la garanzia di qualità di processi e prodotto;
- ***Specifica Tecnica***: contiene l'attività di progettazione, che misura il conseguimento di una solida base tecnologica, e verrà consegnata nella fase di *Revisione di Progettazione*;
- ***Definizione di Prodotto***: contiene l'attività di progettazione, che misura il conseguimento di una solida base architetturale, e verrà consegnata nella fase di *Revisione di Qualifica.*

2.1.4 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 a Prof. Tullio Vardanega e Prof. Riccardo Cardin.

2.2 Processo di sviluppo

2.2.1 Attività

2.2.1.1 Analisi dei Requisiti

Gli *Analisti*, una volta terminato lo *Studio di Fattibilità*, dovranno stilare l'*Analisi dei Requisiti*, che si dovrà attenere alle seguenti regole:

- **Classificazione dei Requisiti**: *Scrivere qui come poi saranno classificati*;
- **Classificazione dei casi d'uso**: *Scrivere qui come poi saranno classificati.*

2.2.1.2 Progettazione

I *Progettisti* 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'*Analisi dei Requisiti.*
Un altro obiettivo è quello di realizzare un prodotto *manutenibile*$_{\text{G}}$, ovvero che abbia una struttura che faciliti 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'*Analisi dei Requisiti* 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 UML

Le tipologie di diagrammi *UML*$_G$ che verranno adoperate per analizzare, descrivere e specificare le scelte progettuali adottate saranno:

- **Diagrammi di classe**:
- **Diagrammi di *package***$_G$: documentano le dipendenze tra le classi ed è utile per controllare la complessità strutturale in sistemi medio-grandi;
- **Diagrammi di attività**: modellano un processo e organizzano più entità in un sistema di azioni secondo un determinato flusso. I diagrammi delle attività sono un tipo particolare di *diagramma di stato*$_G$ che identifica la variazione di stato al verificarsi di alcune condizioni legate ad una o più entità;
- **Diagrammi di sequenza**: descrivono la collaborazione di un gruppo di oggetti che devono implementare collettivamente un comportamento. Sono diagrammi molto semplici, ma che permettono di capire se l'architettura creata viene eseguita.

2.2.1.3 Codifica

In questa fase i *Programmatori*, seguendo le norme delineate nella progettazione, devono realizzare il passaggio dalla fase di pianificazione all'effettiva realizzazione del prodotto. Le norme qui presenti serviranno come strumento per realizzare un codice uniforme e di alta qualità. Inoltre, per mantenerne la manutenibilità, dovrà essere realizzato in inglese. I *Programmatori* si dovranno attenere agli standard di codifica qui di seguito elencati.

2.2.1.3.1 Nomi

- Ogni elemento deve avere un nome rappresentativo e pertinente alla funzione da esso svolta;
- Si dovrà utilizzare la notazione *CamelCase*, ovvero la concatenazione di più parole, ognuna delle quali con lettera iniziale maiuscola. In caso di metodi e variabili la prima lettera dovrà essere minuscola, mentre per le classi maiuscola;
- Si potranno utilizzare singole lettere esclusivamente per identificare gli indici dei cicli;
- Saranno da evitare notazioni troppo simili tra di loro in significato e denominazione;
- Si dovranno evitare errori di ortografia.

2.2.1.3.2 Commenti

Sarà necessario che il codice contenga dei commenti esplicativi per facilitarne la comprensione.
In particolare, si dovranno seguire queste linee guida:

- lista di linee guida per i commenti

2.2.1.3.3 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. Da decidere: lasciamo spazio anche tra i blocchi dentro al metodo?
- Il codice sorgente dovrà essere quanto più suddiviso in file e dovrà essere raggruppato in sottocartelle che dovranno rispecchiare il pattern utilizzato.
- Se ce ne sono altre le inseriamo qui. Da pensare al pattern,

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. Serve una descrizione migliore?

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 Processo di garanzia della qualità

Cosa scriviamo qui? Nell'ISO 1995 è al paragrafo 6.3, da verificare nell'ISO aggiornato.

3.1.2 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*, 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 $\_A$.

3.1.3 Documenti finali ad uso interno

3.1.3.1 Studio di fattibilità (SdF)

Lo *Studio di Fattibilità* 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.

3.1.3.2 Norme di progetto (NdP)

Le *Norme di Progetto* contengono le regole che il team utilizzerà durante lo sviluppo del progetto.

3.1.3.3 Verbale interno (VI)

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

verbale_Numero del verbale_Data del verbale_Tipo del verbale

dove:

- **Numero del verbale**: numero univoco identificativo del verbale;
- **Tipo del verbale**: specifica se *Interno* o *Esterno*;
- **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.

3.1.4 Documenti finali ad uso esterno

3.1.4.1 Piano di Progetto (PdP)

Il *Piano di Progetto* 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 $pianificazione_G$ con l'uso di diagrammi di $Gant_G$.

3.1.4.2 Piano di Qualifica (PdQ)

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

3.1.4.3 Analisi dei Requisiti (AdR)

L'*Analisi dei Requisiti* fornisce la lista dei casi d'uso, i diagrammi delle utilità tra l'utente e il sistema sviluppato e tutti i servizi offerti dal prodotto. Lo scopo è quindi quello di dare una visione generale dei requisiti e dei casi d'uso. Da ampliare quando avremo fatto il documento

3.1.4.4 Specifica Tecnica (ST)

La *Specifica Tecnica* si occupa di dare una descrizione ad alto livello del prodotto, descrivendo pregi e difetti delle sue tecnologie. Da ampliare quando avremo fatto il documento

3.1.4.5 Definizione di Prodotto (DdP)

La *Definizione di Prodotto* descrive i dettagli implementativi del prodotto, andando a definire anche le funzioni delle componenti terminali del sistema tramite diagrammi *UML*.

3.1.4.6 Glossario (G)

Nel documento *Glossario* 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.

3.1.4.7 Manuale Utente (MU)

Il *Manuale Utente* è un manuale per aiutare l'utente nell'utilizzo del prodotto e cresce durante il suo sviluppo. Deve avere un approccio incentrato sulle funzionalità che esso offre.

3.1.4.8 Manuale Manutentore (MM)
Dobbiamo vedere se inserirlo, in quanto non dovremmo occuparci di manutenzione.

3.1.4.9 Verbale Esterno
Il *Verbale Esterno* è un documento in cui si tiene traccia delle discussioni del team con i committenti e i proponenti. Come struttura ricalca quella del *Verbale Interno*.

3.1.5 Struttura del documento

3.1.5.1 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.

3.1.5.2 Diario delle modifiche
In seconda pagina il documento conterrà il diario delle modifiche, che traccerà le modifiche del documento. Sarà organizzato in una tabella così strutturata: Capire se serve la descrizione. In caso negativo togliere il grassetto

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

3.1.5.3 Indice
Dopo il diario 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.

3.1.5.4 Formattazione generale della pagina
Ogni pagina del documento, fatta eccezione per prima, conterrà un'intestazione e un pié di pagina.
L'intestazione presenterà a sinistra il logo e a destra l'email del gruppo.
Nel pié di pagina ci sarà il nome del documento e il nome del gruppo e a destra il numero della pagina.

3.1.6 Norme tipografiche

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

3.1.6.1 Formati

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

- **Nome del documento**: Dovremmo elaborare un comando che, una volta invocato, genera automaticamente il nome del documento e la sua versione
- **Nome del gruppo**: per riferirsi al nome del gruppo si dovrà utilizzare il comando garantendo in questo modo la corretta sintassi;
- **Nome del progetto**: per riferirsi al nome del progetto si dovrà utilizzare il comando garantendo in questo modo la corretta sintassi;
- **Link sito del gruppo**: per riferirsi al link del sito del gruppo si dovrà utilizzare il comando garantendo in questo modo la corretta sintassi;
- **Email del gruppo**: per riferirsi all'indirizzo email del gruppo si dovrà utilizzare il comando garantendo in questo modo la corretta sintassi;
- **Nome del proponente**: per riferirsi al nome del proponente, ovvero del proponente, si dovrà utilizzare il comando garantendo in questo modo la corretta sintassi.

3.1.6.2 Composizione del testo

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

3.1.6.3 Stili di testo

- **Grassetto**: il grassetto deve essere utilizzato per evidenziare parole particolarmente importanti, negli elenchi puntati o nelle frasi; controllare dovunque!
- **Corsivo**: il corsivo deve essere utilizzato nelle seguenti situazioni: da controllare questi punti
  - Ruoli;
  - Documenti;
  - Stati del documento;

  - Citazioni;
  - Glossario;
  - Nomi di file.
- **Maiuscolo**: il maiuscolo deve essere utilizzato solamente per gli acronimi.

3.1.6.4 Sintassi
Nei documenti i membri del team adotteranno la terza persona singolare per la stesura dei documenti.

3.1.6.5 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;
- **ST**: Specifiche Tecniche;
- **MU**: Manuale Utente;
- **MS**: Manuale Sviluppatore;
- **RR**: Revisione dei Requisiti;
- **RP**: Revisione di Progettazione;
- **RQ**: Revisione di Qualifica;
- **RA**: Revisione di Accettazione.

3.1.7 Composizione email

Le email dovranno essere utilizzate principalmente per le comunicazioni esterne ed ufficiali. Il mittente dovrà essere obbligatoriamente *sonsofswe.swe@gmail.com*, mentre i destinatari potranno essere il Prof. Tullio Vardanega, il Prof. Riccardo Cardin o i proponenti del progetto. Le email si articoleranno inoltre in:

- **Oggetto**:
  dovrà essere conciso e preciso, e rimandare immediatamente all'argomento trattato nel corpo;
- **Corpo**
  si dovrà puntare in tutto e per tutto alla sintesi, senza dilungarsi in dettagli inutili;
- **Allegati**
  È altamente sconsigliato l'invio di allegati direttamente tramite email. Se possibile, e in accordo con il destinatario, si preferirà l'invio tramite link da *Google Drive*$_{\text{G}}$.

3.1.8 Componenti grafiche

Da controllare, quando inizieremo ad usarle

- Tabelle
- Immagini

3.1.9 Nome del file *.pdf*$_G$

Ogni documento sarà stampato su .pdf utilizzando, per il nome, questa regola:

NomeFile_ versione.pdf

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

3.1.10 Struttura dei file in LaTeX

Ogni file .pdf è la stampa di un insieme di file LaTeXcosì strutturato:

- *sos.sty*: contiene tutti i package e le variabili globali da utilizzarsi in tutti i documenti. Contiene uno script che genera la prima pagina e che viene invocato dal main. Si occupa inoltre di formattare il documento. Verrà usato in tutti i main.
- *main.tex* contiene l'inclusione del template e si occupa di generare la copertina, l'indice delle sezioni e di includere i file relativi alle sezioni;
- *sezione.tex*: ogni sezione, in maniera più atomica possibile, sarà inserita all'interno di un singolo file .tex. Questa scelta sarà adottata dal team per puntare all'assenza di conflitti durante modifiche contemporanee a più parti dello stesso documento, e per facilitare la revisione durante l'assemblaggio;
- *comandi.tex*: contiene tutti i comandi relativi alla creazione della copertina. Questa modularità permette di usare lo stesso package per costruire copertine differenti, cambiando di volta in volta i valori al suo interno.
- *"img" folder*: conterrà il logo e le altre immagini utilizzate nel documento.

3.1.11 Versionamento

Il *versionamento*$_G$ permette a ciascun membro del team di condividere il lavoro nello spazio comune e di lavorare su vecchi e nuovi *CI*$_G$ *Configuration Item* 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 *X* viene riportato a 0.
- **Z**:

- Alla creazione parte da 1;
- Viene incrementato dal redattore del documento ogni volta che viene modificato.
- Quando vengono incrementati X o Y viene riportato a 0.

3.1.12 Strumenti

3.1.12.1 *LaTeX* G

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

- Permette di rendere ogni sezione molto autonoma, mantenendo comunque lo stile coerente;
- Permette un versionamento più semplice e compatibile con *Git*G;
- Evita conflitti usando software ed envornment differenti.

3.1.12.2 Da aggiungere qui eventuali altri strumenti che utilizzeremo per altro

3.1.13 Gestione del *repository*G

Per avere traccia di tutte le modifiche realizzate sui documenti e sul codice prodotti, il team ha deciso di utilizzare *GitHub*, creando una *repository*G. Si seguiranno le norme dscritte qui di seguito.

3.1.13.1 Struttura

- Il *master*G è annidato sulla repository *origin*G e deve contenere solo il *template*G per la scrittura di tutti i documenti;
- Ogni *branch*G annidato sul master deve corrispondere solo alle *revisioni*G;
- Ogni branch annidato sui branch relativi alle revisioni corrispondono ai documenti ad esse corrispondenti;
- Ogni membro del gruppo potrà modificare i files per 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*.
- Capire come fare la repo quando si dovrà mettere dentro il codice

3.1.13.2 Tipi di file Tutti i file pertinenti verranno caricati nel repository e sottoposti a versionamento. Verranno tuttavia ignorati, quindi inseriti all'interno del file *.gitignore*G, tutte quelle estensioni dei file generati automaticamente durante le varie *build*G.

3.1.13.3 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. Dovrà essere eseguito nella maniera più *atomica*G possibile; qualora non si riuscisse a concludere un'attività entro la giornata, si dovrà eseguire un commit alla fine di essa, per garantirne una copia offline.

3.2 Processo di verifica

Il processo di verifica deve essere continuo e 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, cioè che vengano evitate scorrettezze;
- Non intrusiva, cioè che non interrompa alcuna attività durante la sua esecuzione.

Le modalità sfruttate saranno elencate qui di seguito.

3.2.1 Analisi

3.2.1.1 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**$_G$
  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*$_G$ di tutti gli errori più comuni, si passerà alla fase di *Inspection*$_G$. 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.

3.2.1.2 Analisi dinamica

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

3.2.2 Test

Il test 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.
I test si possono dividere nelle categorie qui descritte.

3.2.2.1 Test di *unità*$_G$ (TU)

Il suo obiettivo è quello di verificare la correttezza del codice *as implemented*, ovvero puntando a controllare il codice in maniera microscopica.
La responsabilità della sua realizzazione è del *Programmatore* per le unità più semplici, altrimenti di un *Verificatore* indipendente.
Le risorse consumate da questo tipo di test sono molto basse, poiché coinvolge piccole parti di codice con basso accoppiamento le une con le altre. Serve una sintassi identificativa?

3.2.2.2 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.
Può rilevare questi 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.

Questo tipo di test consuma molte risorse, poiché la grandezza del codice da controllare è significativa e il grado di accoppiamento tra le varie parti da testare è massimo. Quindi il suo utilizzo deve essere ponderato e, in generale, da non preferire a quello di unità. Serve una sintassi identificativa?

3.2.2.3 Test di sistema (TS)

Il test di sistema può essere visto 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. Serve una sintassi identificativa?

3.2.2.4 Test di *regressione*$_G$ (TR)

Il test di regressione è 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.
Questo tipo di test 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. Serve una sintassi identificativa?

3.2.2.5 Test di validazione Il test di validazione è un'attività supervisionata dal committente e/o 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*$_G$, con eventuale manutenzione.

3.2.3 Strumenti

3.2.3.1 Strumenti per l'analisi statica

- **TexStudio**$_G$[1]
  Questo editor include il correttore ortografico automatico. Tuttavia non è preciso nell'individuazione di errori più sottili, quindi servirà comunque un'analisi più approfondita da parte dei *Verificatori*;
- **Indice *Gulpease***$_G$
  Per i test di leggibilità il team ricorrà al calcolo dell'indice Gulpease;
- **JSHint**$_G$[2]
  È uno strumento *OpenSource*$_G$ funzionale alla rilevazione degli errori e possibili problemi nel codice *JavaScript*$_G$;
- **W3C Markup Validator Service**$_G$[3]
  È uno strumento per la validazione dei fogli di stile *CSS*$_G$;
- **Draw.io**$_G$[4]
  È un software per la creazione di diagrammi di flusso, di processo, organigrammi, UML, ER e diagrammi di rete.

[1] https://www.texstudio.org/
[2] http://jshint.com/
[3] https://jigsaw.w3.org/css-validator/
[4] https://www.draw.io/

3.2.3.2 Strumenti per l'analisi dinamica

- **Karma**$_G$[5]
  È uno strumento per effettuare test di unità sugli script realizzati, installabile come modulo per *Node.js*$_G$;
- **Mocha**$_G$[6]
  È un framework per l'esecuzione dei test asincroni e in serie di Javascript, scritto Node.js.

[5] https://karma-runner.github.io/2.0/index.html
[6] https://mochajs.org/

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 committente e 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 nome preceduto dal titolo del destinatario o 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 gli argomenti da affrontare, da fungere da moderatore QUESTO IL MODERATORE LO FA?, 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$_G$**: 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$_G$*. 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 su 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 *Norme di progetto*, di redigere l'*Organigramma*$_G$, di garantire che vengano rispettati i ruoli assegnati all'interno del *Piano di progetto*, 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 *Studio di Fattibilità*.
- Redigere l'*Analisi dei Requisiti*, 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 *Norme di Progetto* e supporto alla redazione del *Piano di Progetto*.

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.
- Redigere *Specifica Tecnica*, *Definizione di Prodotto* e la parte programmatica del *Piano di Qualifica*.

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 *Manuale Utente* ed *Manuale Manutentore*.

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 *Norme di Progetto* e della conformità al *Piano di Qualifica*.
- Redigere il *Piano di Qualifica*.

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*$_G$; 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*$_G$ 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".

4.2.3.1.1 Possibile stato di un ticket Un ticket può essere nei seguenti stati:

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

4.2.3.1.2 aggiungere diagramma uml???

4.3 Processo dell'infrastruttura

4.3.1 Ambienti di sviluppo

4.3.1.1 Sistemi operativi

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.
- OS X El Capitan versione 10.11.6;

4.3.2 Strumenti

4.3.2.1 Telegram

Telegram è un servizio di messaggistica istantanea multipiattaforma. [7]

4.3.2.2 Slack

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. [8]

4.3.2.3 Google Drive

Google Drive è un servizio di memorizzazione e archiviazione online fornito da Google basato sul *Cloud*$_{\text{G}}$. Esso verrà utilizzato dal gruppo per lo scambio di documenti ed informazioni di supporto allo sviluppo del progetto ma non soggetti al versionamento. [9]

4.3.2.4 Fogli Google

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. [10]

4.3.2.5 Github

Github è un' implementazione dello strumento di controllo di versione Git ed offre un servizio di *hosting*$_{\text{G}}$ 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 sue modifiche direttamente dal proprio account personale. [11]

4.3.2.6 Asana

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 altri, da cui *dipende*, non sono stati completati in precedenza;

[7] https://telegram.org/
[8] https://slack.com/
[9] https://www.google.com/drive/
[10] https://www.google.it/intl/it/sheets/about/
[11] https://github.com/

- 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. [12]

4.3.2.7 Instagantt

Instagantt è un servizio web fortemente legato ad Asana, 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. [13]

[12] https://asana.com/
[13] https://instagantt.com/