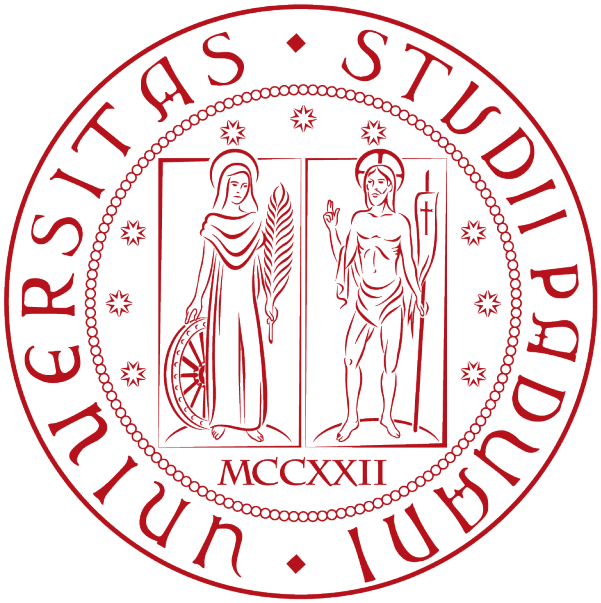

Norme di Progetto

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

1.1 Scopo del documento

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

-

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 gruppo 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 progetto, il proponente e i committenti.
- **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 considerabili vantaggiosi per il gruppo.
- **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 le criticità riscontrate e viene motivata l'eventuale approvazione o esclusione.

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.

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

- **Diagrammi di classe**
- **Diagrammi di package$_G$**
- **Diagrammi di attività**
- **Diagrammi di sequenza**

2.2.1.2.2 Obiettivi della progettazione 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, ovvero che abbia una struttura che faciliti i cambiamenti futuri. Infine deve realizzare al meglio i requisiti di qualità imposti dal committente.

2.2.1.3 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 ai seguenti standard di codifica:

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 maiuscola, 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 ed 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 chiususa 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**
- **Documenti esterni** Da specificare la distribuzione a committenti e 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

Tra i documenti ad uso interno saranno presenti:

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) I *Verbale Interno* serviranno 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:**

Tra i documenti ad uso esterno saranno presenti:

3.1.3.4 Piano di Progetto (PdP) Il *Piano di Progetto* contiene indicazioni sulle scadenze temporali e fornisce un preventivo dei costi da presentare al proponente. Vengono inoltre individuati i rischi e analizzate le loro occorrenze. In questo documento vengono fatte emergere le *milestone* legate ai punti critici e viene effettuata una pianificazione con l'uso di diagrammi di Gant.

3.1.3.5 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.

3.1.3.6 Analisi dei Requisiti (AdR) L'*Analisi dei Requisiti* fornisce la lista dei casi d'uso, i diagrammi delle utilità tra utente e 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

3.1.3.7 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

3.1.3.8 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.

3.1.3.9 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.

3.1.3.10 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.

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

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

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
  - Approvazione
  - Uso
  - Distribuzione
- Descrizione del documento.

3.1.4.2 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**
- **Descrizione**
- **Autore**
- **Ruolo**
- **Data**

3.1.4.3 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.

3.1.4.4 Formattazione generale della pagina Ogni pagina del documento, fatta eccezione della prima, conterrà una intestazione e un pié di pagina.
L'intestazione conterrà a sinistra il logo e a destra la mail del gruppo.
Il pié di pagina ci sarà il nome del documento e il nome del gruppo e a destra il nome della pagina sul numero totale.

3.1.5 Norme tipografiche

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

3.1.5.1 Formati

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

  $$YYYY\text{-}MM\text{-}DD$$

  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
  - **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.

3.1.5.2 Composizione del testo

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

3.1.5.3 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: ogni riferimento a ruoli di progetto va scritto in corsivo;
  - Documenti: ogni riferimento a un documento va scritto in corsivo;
  - Stati del documento: ogni stato del documento va scritto in corsivo;
  - Citazioni: ogni citazione va scritta in corsivo;
  - Glossario: ogni parola presente nel glossario, oltre ad avere un pedice, deve essere scritta in corsivo.

3.1.5.4 Sigle Servono?

3.1.6 Composizione email

Le email dovranno essere utilizzate principalmente per le comunicazioni esterne e ufficiali. Il mittente dovrà essere obbligatoriamente *prova@seibrutto.it*, mentre il destinatario 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, che rimandi immediatamente all'argomento trattato nel corpo.
- **Corpo** Si dovrà puntare in tutto e per tutto alla sintesi, senza dilungarsi in dettagli inutili.
- **Allegati** E' 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.

3.1.7 Componenti grafiche

Da controllare, quando inizieremo ad usarle

- Tabelle
- Immagini

3.1.8 Versionamento

Il versionamento$_G$ permette a ciascun membro del team di lavorare su vecchi e nuovi CI_G *Configuration Item* senza rischio di riscritture accidentali e di condividere il lavoro nello spazio comune.
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;

3.1.9 Strumenti

3.1.9.1 LaTeX

Il team ha scelto di usufruire del linguaggio LaTeXper questi motivi:

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

3.1.9.2 Da aggiungere qui eventuali altri strumenti che utilizzeremo per altro

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;

Si sfruttano le seguenti modalità:

3.2.1 Analisi

L'analisi si divide in due parti:

3.2.1.1 Analisi statica Studia il codice sorgente e la documentazione e controlla che essi seguino le norme. Non richiede l'esecuzione del prodotto software in nessuna sua parte, ma richiede l'osservazione del codice. Questo processo è 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 di 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 nello sviluppo del prodotto per trovare eventuali errori che possono essere riscontrati. Non sapendo che tipo di errori cercare, si effettua una lettura a largo spettro. Quando i *Verificatori*avranno stilato una *checklist* di tutti gli errori più comuni, si passerà alla fase di *Inspection*. Essendo un'analisi continua, la *checklist* tenderà ad aumentare costantemente di dimensione, rendendo più efficace ed efficiente l'*Insection*.
- **Inspection$_G$** 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 viene 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 in cinque categorie:

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 è molto basso, 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

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**$_\mathbf{G}$[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**$_\mathbf{G}$ Per i test di leggibilità il team ricorrà al calcolo dell'indice Gulpease.
- **JSHint**$_\mathbf{G}$[2] È uno strumento *OpenSource*$_G$ funzionale alla rilevazione degli errori e possibili problemi nel codice Javascript.
- **W3C Markup Validator Service**$_\mathbf{G}$[3] È uno strumento per la validazione dei fogli di stile *CSS*.
- **Draw.io**$_\mathbf{G}$[4] È un software per la creazione di diagrammi di flusso, di processo, organigrammi, UML, ER e diagrammi di rete.

3.2.3.2 Strumenti per l'analisi dinamica

- ***Karma***[5] E' uno strumento per effettuare test di unità sugli script realizzati. Installabile come modulo per *Node.js*.

[1] https://www.texstudio.org/
[2] http://jshint.com/
[3] https://jigsaw.w3.org/css-validator/
[4] https://www.draw.io/
[5] https://karma-runner.github.io/2.0/index.html

- ***Mocha***[6] Framework per l'esecuzione dei test asincroni e in serie di Javascript, scritto in *Node.js*.

4 Processi organizzativi

4.1 Processo di coordinamento

4.1.1 Comunicazioni

4.1.2 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***$_{\mathrm{G}}$, servizio di messaggistica istantanea multipiattaforma.
- ***Slack***$_{\mathrm{G}}$, servizio di comunicazione pensato per facilitare il lavoro di gruppo tramite canali distinti.

4.1.3 Comunicazioni esterne

- **E-mail**: E' stata creato l'indirizzo di posta elettronica: **sonsofswe.swe@gmail.com**

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

  Le e-mail dovranno avere la seguente forma:

  - **Oggetto**: L'oggetto dev'essere chiaro e conciso, per riconoscere e distinguere facilmente tra loro le mail.
  - **Apertura**: Ogni mail deve iniziare con un aggettivo di circostanza seguito dal nome preceduto dal titolo del destinatario, e terminare con una virgola.
  - **Corpo**: Il corpo dev'essere breve ma esaustivo; nella prima parte verrà spiegata la ragione per cui si scrive per garantire la corretta comprensione del messaggio.
  - **Allegati**: E' possibile l'invio di allegati tramite mail, 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**: Su richiesta del committente verrà utilizzato per lo scambio di informazioni in maniera ufficiosa.

[6] https://mochajs.org/