Norme di Progetto

Gruppo MILCTdev – Progetto OpenAPM
milctdev.team@gmail.com

Versione	1.0.0
Redazione	Tommaso Carraro Luca Dal Medico
Verifica	Mattia Bano Carlo Munarini
Approvazione	Dragos Cristian Lizan
Uso	Interno
Distribuzione	Prof. Tullio Vardanega Prof. Riccardo Cardin Gruppo MILCTdev

Descrizione
Questo documento descrive le regole, gli strumenti e le convenzioni che il *teamG* MILCTdev dovrà rispettare per tutta la realizzazione del *progettoG* OpenAPM

Registro delle modifiche

Descrizione	Autore	Ruolo	Data	Versione
Approvazione del documento	Dragos Cristian Lizan	Responsabile	26-11-2017	1.0.0
Verifica del documento	Mattia Bano	Verificatore	22-11-2017	0.4.0
Verifica del documento	Carlo Munarini	Verificatore	18-11-2017	0.3.0
Creazione della parte : Processi organizzativi	Luca Dal Medico	Amministratore	17-11-2017	0.2.3
Stesura della parte supporto : Verifica	Tommaso Carraro	Amministratore	17-11-2017	0.2.2
Aggiunte correzioni dopo verifica	Luca Dal Medico	Amministratore	17-11-2017	0.2.1
Verifica documento	Mattia Bano	Verificatore	16-11-2017	0.2.0
Aggiunta riferimenti per la parte Supporto : Documentazione	Tommaso Carraro	Amministratore	15-11-2017	0.1.2
Aggiunte correzioni dopo verifica	Tommaso Carraro	Amministratore	15-11-2017	0.1.1
Verifica documento	Carlo Munarini	Verificatore	14-11-2017	0.1.0
Stesura della parte : garanzia di qualità	Luca Dal Medico	Amministratore	14-11-2017	0.0.5
Stesura del processo di supporto : configurazione	Tommaso Carraro	Amministratore	14-11-2017	0.0.5
Stesura del processo di supporto : documentazione	Tommaso Carraro	Amministratore	14-11-2017	0.0.4
Stesura della parte : sviluppo	Tommaso Carraro	Amministratore	14-11-2018	0.0.3
Stesura processo primario : fornitura	Luca Dal Medico	Amministratore	13-11-2017	0.0.2
Stesura introduzione	Luca Dal Medico	Amministratore	12-11-2017	0.0.1
Inserimento template nel documento	Tommaso Carraro	Amministratore	10-11-2017	0.0.0

Indice

Immagini

1 Introduzione

1.1 Scopo del documento

Questo documento ha lo scopo di definire le regole, gli strumenti e le convenzioni adottate dal gruppo MILCTdev durante l'intero svolgimento del progetto. In quest'ottica, questo documento deve essere visionato da tutti i componenti del gruppo, i quali sono obbligati ad applicare quanto scritto al fine di mantenere omogeneità e coesione in ogni aspetto del progetto. In caso di modifiche o aggiunte al presente documento è obbligatorio informare ogni membro del gruppo.

1.2 Scopo del prodotto

Lo scopo del *prodottoG* è realizzare un set di funzioni basate su *ElasticsearchG* e *KibanaG* per interpretare i dati raccolti da un *AgentG*. I dati interpretati forniranno a *DevOpsG* statistiche e informazioni utili per comprendere il funzionamento della propria applicazione. In particolare si richiede lo sviluppo di un motore di generazione di *metricheG* da *traceG*, un motore di generazione di *baselineG* basato sulle metriche del punto precedente, e un motore di gestione di *critical eventG*.

1.3 Glossario

All'interno del documento sono presenti termini che possono assumere significati diversi a seconda del contesto. Per evitare ambiguità, i significati dei termini complessi adottati nella stesura della documentazione sono contenuti nel documento *Glossario v1.0.0*. Per segnalare un termine del testo presente all'interno del Glossario verrà aggiunta una *G* a pedice e il testo sarà in corsivo.

1.4 Riferimenti

1.4.1 Riferimenti Normativi

- **Standard ISO/IEC 12207:1995**
  http://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf

1.4.2 Riferimenti Informativi

- **Verbale Interno - 28-11-2017**

- **Documentazione - Slide del corso "Ingegneria del Software"**
  http://www.math.unipd.it/~tullio/IS-1/2017/Dispense/L12.pdf
- **Documentazione su Git**
  http://www.piratpartiet.it/mediawiki/index.php?title=Mini_Guida_a_GIT#Comandi_di_base_a_riga_di_comando
- **Informazioni su TexMaker**
  http://www.xm1math.net/texmaker/
- **Processi Software - Slide del corso "Ingegneria del Software"**
  http://www.math.unipd.it/~tullio/IS-1/2017/Dispense/L03.pdf
- **SWEgo**
  http://www.swego.it
- **Julia**
  https://www.juliasoft.com
- **IntelliJ**
  https://www.jetbrains.com/idea/
- **Gestione di Progetto - Slide del corso "Ingegneria del Software"**
  http://www.math.unipd.it/~tullio/IS-1/2017/Dispense/L06.pdf

2 Processi primari

2.1 Fornitura

2.1.1 Scopo del processo

Il processo di fornitura contiene tutte le attività e i compiti del fornitore. Esso può essere avviato sia dalla decisione di preparare una risposta ad una richiesta del *ProponenteG*, sia con la stipulazione di un contratto con l'acquirente per la consegna del prodotto. Il processo continua con la determinazione delle procedure e delle risorse necessarie per l'organizzazione ed il completamento del progetto, incluso lo sviluppo di un Piano di Progetto e l'esecuzione di questo fino alla consegna del materiale prodotto all'acquirente. Il processo di Fornitura è composto dalle seguenti attività:

- avvio;
- preparazione della risposta;
- contrattazione;
- pianificazione;
- esecuzione e controllo;
- revisione e valutazione;
- consegna e completamento.

2.1.2 Descrizione

Questa sezione tratta le norme che i membri del gruppo MILCTdev sono tenuti a rispettare nelle fasi di progettazione, sviluppo e consegna del prodotto OpenAPM, al fine di proporsi e diventare fornitori nei confronti della Proponente Kirey Group e dei *CommittentiG* Prof. Tullio Vardanega e Prof. Riccardo Cardin.

2.1.3 Studio di Fattibilità

Successivamente alla presentazione ufficiale dei *capitolatiG* d'appalto, avvenuta in data 10-11-2017 presso l'aula 1C150 di Torre Archimede, i membri del gruppo hanno tenuto una discussione che ha portato a proporsi come fornitori di OpenAPM. Tale scelta è motivata nel documento *Studio di Fattibilità v1.0.0*, la cui stesura è compito degli Analisti. Questo documento indica per ogni capitolato:

- **Descrizione generale:** breve presentazione del progetto;

- **Obiettivo finale:** descrizione in linea di massima delle caratteristiche principali richieste nel prodotto completato ed il suo ambito di utilizzo;
- **Tecnologie richieste:** elenco delle tecnologie da impiegare nella realizzazione del prodotto;
- **Valutazione finale:** riassunto degli aspetti principali che hanno portato il gruppo ad accettare o scartare il capitolato in questione.

2.1.4 Rapporti di fornitura con la Proponente

Durante l'intero svolgimento del progetto il gruppo intende instaurare con la Proponente Kirey Group, in particolar modo nelle figure dei referenti Stefano Bertolin e Stefano Lazzaro, un rapporto di collaborazione costante e costruttiva al fine di:

- determinare i bisogni della Proponente;
- specificare come saranno definiti ed eseguiti i processi;
- stimare i costi;
- accordarsi circa la qualifica di prodotto.

2.1.5 Documentazione fornita

Vengono qui specificati i documenti forniti alla Proponente Kirey Group e ai Committenti Prof. Tullio Vardanega e Prof. Riccardo Cardin al fine di assicurare la massima trasparenza circa le attività di:

- **Pianificazione, consegna e completamento:** descritte all'interno del *Piano di Progetto v1.0.0*;
- **Analisi:** tutte le analisi dei requisiti e i casi d'uso sono specificati in *Analisi dei Requisiti v0.0.0*;
- **Verifica e validazione:** tutte le procedure atte a garantire la qualità dei processi e di prodotto sono descritte e definite nel documento *Piano di Qualifica v0.0.0*.

2.1.6 Collaudo e Consegna del Prodotto

Al fine di consegnare il prodotto terminato, il gruppo deve effettuare un collaudo in presenza della Proponente e dei Committenti. Precedentemente a questo test il gruppo deve assicurare correttezza, completezza e affidabilità di ogni parte del materiale realizzato affinché si possa dimostrare che:

- tutti i requisiti obbligatori descritti nel documento *Analisi dei Requisiti v0.0.0* siano completamente soddisfatti;
- l'esecuzione di tutti i test descritti nel documento *Piano di Qualifica v0.0.0* abbia esito positivo.

In seguito al collaudo finale, il Responsabile di Progetto presenta il consuntivo al Committente e consegna il prodotto su supporto fisico.

2.2 Sviluppo

2.2.1 Scopo del processo

Il seguente processo contiene le attività e i compiti dello sviluppatore. Contiene attività per analisi dei requisiti, design, codifica, integrazione, testing e installazione del prodotto *softwareG*. Lo sviluppatore esegue o supporta le attività di questo processo, in conformità con il contratto stipulato con il Committente. Questo processo è composto dalle seguenti attività:

1. implementazione del processo;
2. analisi dei requisiti di sistema;
3. disegno dell'architettura di sistema;
4. analisi dei requisiti software;
5. disegno dell'architettura software;
6. disegno dell'architettura software in dettaglio;
7. codifica e test del software;
8. integrazione del software;
9. test di qualifica del software;
10. integrazione di sistema;
11. test di qualifica del sistema;
12. installazione del software;
13. supporto all'accettazione del software.

2.2.2 Descrizione

Questa sezione tratta le norme che i membri del gruppo hanno stilato in merito alle attività di analisi dei requisiti, progettazione e codifica.

2.2.3 Analisi dei requisiti

In seguito al completamento dello Studio di Fattibilità, è compito degli Analisti redigere il documento di Analisi dei Requisiti. Esso ha l'intento di formalizzare e rendere tracciabili i requisiti e i casi d'uso individuati, presentandoli seguendo le specifiche sottostanti.

2.2.3.1 Requisiti

Ogni requisito è classificato tramite il seguente formalismo:

R{X}{Y}{codice_identificativo}

dove:

- **X** specifica la tipologia di requisito:
  - *F*: requisito funzionale;
  - *P*: requisito prestazionale;
  - *Q*: requisito qualitativo;
  - *V*: vincolo progettuale.
- **Y** indica uno dei seguenti gradi di necessità:
  - *O*: requisito obbligatorio;
  - *D*: requisito desiderabile;
  - *F*: requisito facoltativo.
- **codice_identificativo** è un codice composto da una serie di numeri separati tramite punto che identificano il requisito in maniera univoca e lo esprimono gerarchicamente.

È imperativo che un requisito non possa cambiare denominazione col passare del tempo.

La lista dei requisiti è espressa in forma tabellare e ciascun requisito è esplicato nel seguente modo:

- **ID Requisito**: rappresenta un identificativo del requisito, costruito come sopra;
- **Descrizione**: breve ma chiara e precisa; non deve lasciare spazio ad ambiguità;
- **Fonte**: indica la fonte da cui è stato estrapolato il requisito e può essere:
  - **capitolato**: derivante direttamente dal capitolato;
  - **verbale**: derivante da un incontro verbalizzato;
  - **caso d'uso**: derivante da uno o più casi d'uso.

Esempi di requisiti:

ID Requisito	Descrizione	Fonti
RVO1.2	Esempio 1	Capitolato; UC1
RPF3.1	Esempio 2	Capitolato; UC2

2.2.3.2 Casi d'uso

Ogni caso d'uso è classificato tramite il seguente formalismo:

UC{codice_identificativo}

dove:

- **codice_identificativo** è un codice composto da una serie di numeri separati tramite punto che identificano il caso d'uso in maniera univoca e lo esprimono gerarchicamente.

È imperativo che un caso d'uso non possa cambiare denominazione col passare del tempo.

La lista dei casi d'uso è espressa in forma tabellare e ciascun caso d'uso è esplicato nel seguente modo:

- **ID Caso d'uso**: identificativo del caso d'uso, costruito come scritto sopra;
- **Titolo**: titolo del caso d'uso;
- **Descrizione**: breve descrizione del caso d'uso;
- **Attori**: lista di attori principali e secondari coinvolti nel caso d'uso;
- **Precondizione**: condizione che deve essere verificata prima dell'esecuzione del caso d'uso;
- **Postcondizione**: condizione che deve essere verificata dopo dell'esecuzione del caso d'uso;
- **Scenario principale**: descrizione composta dal flusso dei casi d'uso figli;
- **Scenari alternativi**: descrizione composta dai casi d'uso che non appartengono al flusso principale di esecuzione;
- **Estensioni**: indica quali sono tutte le estensioni, se presenti;
- **Inclusioni**: indica quali sono tutte le inclusioni, se presenti;
- **Generalizzazioni**: indica quali sono tutte le generalizzazioni, se presenti.

Esempi di casi d'uso:

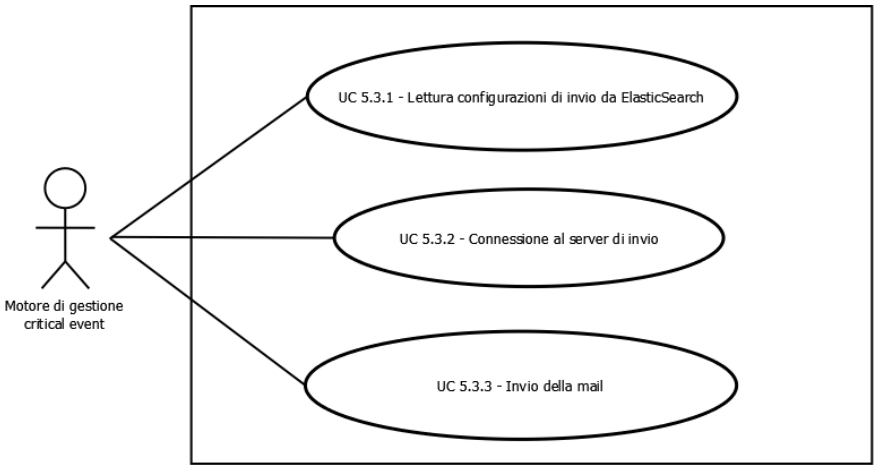

Figure 1: L'immagine raffigura un esempio di caso d'uso

2.2.4 Progettazione

L'attività di progettazione consiste nel descrivere un'architettura che garantisca, in maniera dimostrabile, il rispetto di tutti i requisiti presenti in *Analisi dei Requisiti v0.0.0*. Per far ciò sono necessari:

- la descrizione di test di unità e di integrazione che consentano la verifica dell'architettura;
- il tracciamento di ogni requisito al componente che lo soddisfa per avere garanzia di completezza, oltreché per disporre di una misura di progresso della progettazione stessa.

Tale attività è compito dei Progettisti che devono definire, mediante diagrammi *UML 2.0G*, architettura generale ed architettura in dettaglio tali che:

- siano modulari, rispettino il principio dell'*information hidingG* e minimizzino le dipendenze tra componenti;
- sfruttino i *design patternG*, riconducendosi a problemi già conosciuti e risolvendoli con soluzioni progettuali già consolidate, documentandone anche l'utilizzo;
- siano comprensibili ai Programmatori, per semplificare la successiva attività di codifica, ai Committenti, al Responsabile e ai Verificatori.

2.2.5 Codifica

Questa attività, competenza dei Programmatori, consiste nella scrittura del codice sorgente di quanto progettato nell'attività di progettazione. Il codice deve compilare senza errori né warning. Deve inoltre rispettare le successive specifiche ed essere accompagnato da relativa documentazione per poter assicurare manutenibilità e leggibilità.

2.2.5.1 Intestazione

Ogni file sorgente deve iniziare con la seguente intestazione:

```
/**
 * Name: {nome del file}
 * Package: {package di appartenenza}
 *
 * History:
 * Version          Date                Programmer
 * ================================================
 * {versione}       {AAAA-MM-GG}        {nome e cognome}
 * ------------------------------------------------
 * {descrizione}
 * ================================================
```

```
* {versione}        {AAAA-MM-GG}         {nome e cognome}
* ------------------------------------------------------
* {descrizione}
* ======================================================
* . . .
*/
```

Classi e metodi devono invece essere preceduti da commenti secondo le convenzioni Javadoc:

http://www.oracle.com/technetwork/articles/java/index-137868.html

2.2.5.2 Nomenclatura

- I nomi dei file devono essere tutti in minuscolo;
- i nomi di variabili e metodi devono avere la prima lettera minuscola;
- i nomi delle classi devono avere la prima lettera maiuscola;
- i nomi di costanti devono essere tutti in maiuscolo;
- i nomi di variabili, metodi e classi devono essere in *camel caseG*.

2.2.5.3 Formattazione

- Indentazione mediante tabulazioni lunghe 4 spazi;
- inserire sempre le parentesi graffe negli if *statementG*, nei cicli while, for con una singola istruzione;
- il codice che appartiene ad un blocco deve essere innestato allo stesso livello di quel blocco;
- metodi e linee di codice vanno resi corti ove possibile;
- le parantesi graffe non devono mai stare da sole su una riga, sarebbe uno spreco di righe di codice visibili a schermo.

2.2.5.4 Commenti

Il codice va commentato per semplificarne la comprensione e la manutenzione; questo significa anche evitare di commentare ciò che è ovvio:

- i commenti riguardo uno statement è preferibile che siano posizionati sopra lo statement stesso;
- una modifica del codice deve essere seguita da una modifica ai commenti che vi ci si riferiscono.

2.2.5.5 Ricorsione

La ricorsione va evitata il più possibile. Ogni metodo ricorsivo dovrà essere accompagnato da una prova di terminazione e dall'analisi dell'occupazione della memoria. Risultati non soddisfacenti ne comporteranno il rifiuto.

2.2.6 Strumenti utilizzati

2.2.6.1 SWEgo

Figure 2: SWEgo: applicazione web per il tracciamento dei requisiti

SWEgoG è un potente strumento di tracciamento dei requisiti disponibile online. Esso offre una moltitudine di funzioni che semplificano e velocizzano il modo di produrre software.

2.2.6.2 IntelliJ IDEA

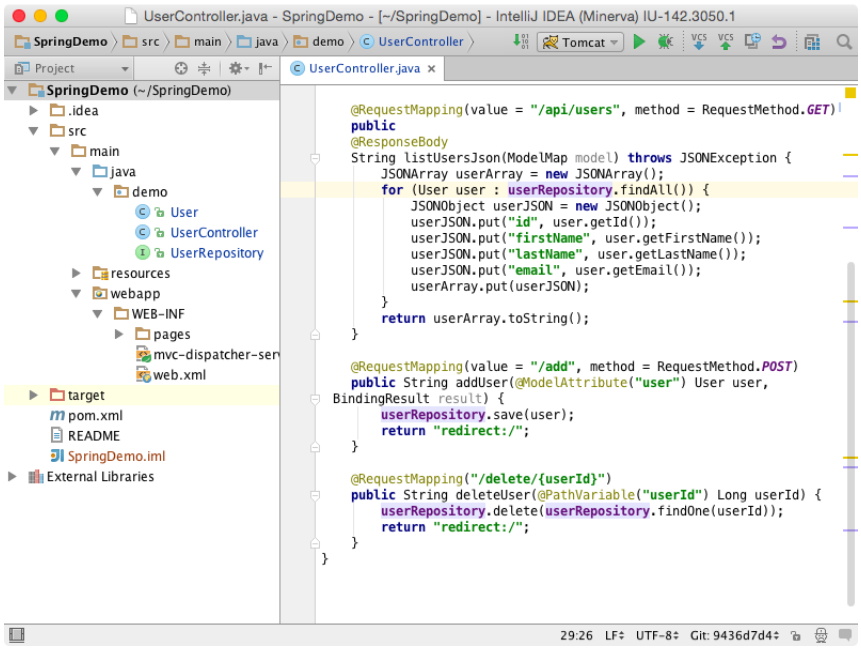

Figure 3: IntelliJ IDEA: IDE utilizzato dal gruppo MILCTdev

È un potente *IDEG* che presenta integrazione con *GitHubG* e strumenti di analisi statica utili per altri processi. L'utilizzo dello stesso IDE, con la stessa configurazione, da parte di tutti i Programmatori preverrà o quantomeno faciliterà la soluzione di problemi riguardanti l'ambiente di sviluppo.

3 Processi di supporto

3.1 Documentazione

3.1.1 Scopo del processo

Il processo di documentazione si prefigge lo scopo di registrare le informazioni prodotte dai processi del ciclo di vita del software. Questo processo contiene l'insieme di attività atte alla pianificazione, progettazione, sviluppo, produzione, modifica, distribuzione e mantenimento dei documenti necessari al team di progetto e agli utenti del sistema o del prodotto software. Questo processo è composto dalle seguenti attività:

- implementazione del processo;
- progettazione e sviluppo;
- produzione;
- manutenzione.

3.1.2 L'importanza della documentazione

Il processo di documentazione è uno dei processi di supporto più importanti, esso aiuta a:

- assicurare che i processi produttivi si svolgano con la qualità attesa;
- garantire una completa, accurata, tempestiva e non-invasiva circolazione di informazione;
- facilitare il controllo di avanzamento secondo le regole del modello di sviluppo adottato;
- segnare il confine tra creatività e disciplina.

3.1.3 Descrizione

In questa sezione vengono illustrate le norme e le decisioni prese dal gruppo di progetto in merito alla stesura, la verifica e l'approvazione della documentazione formale. Queste norme devono essere rispettate per la produzione di documenti validi, comprensibili e coerenti.

3.1.4 Categorie di documenti: Formali ed Informali

Ogni documento può essere informale oppure formale:

1. **Informale**: un documento informale è un documento in fase di stesura che può aver passato o meno le verifiche. Può diventare formale solamente dopo l'approvazione del Responsabile di Progetto;
2. **Formale**: un documento formale è approvato dal Responsabile di Progetto e pronto per essere presentato sotto forma di documento Interno o Esterno.

3.1.5 Categorie di documenti: Interni ed Esterni

Esistono due categorie di documenti formali:

1. **Interni**: sono i documenti utilizzati dal team di progetto;
2. **Esterni**: sono i documenti condivisi con la Proponente in sede di revisione dei requisiti, e con il Committente nelle revisioni successive.

Ogni documento formale può far parte di una di queste categorie. I documenti informali possono rientrare nella *repositoryG* documentale del gruppo ma non devono essere poi presentati alle revisioni.

3.1.6 Struttura repository documentale

Nella repository, la cartella relativa alla documentazione (Documenti) deve contenere quattro cartelle: RR (Revisione dei Requisiti), RP (Revisione di Progettazione), RQ (Revisione di Qualifica), RA (Revisione di Accettazione). Ognuna di queste deve contenere due cartelle: una per i documenti interni (Interni) e una per i documenti esterni (Esterni). I verbali devono essere posti in un'apposita cartella, denominata "Verbali", posta all'interno della documentazione interna (se si tratta di un verbale interno) o esterna (se si tratta di un verbale esterno).

3.1.7 Documenti da presentare

Questa sezione illustra i documenti che dovranno essere presentati nelle varie revisioni di progetto. I documenti sono presentati in ordine alfabetico e tra parentesi viene indicato il loro utilizzo (esterno o interno):

- **Analisi dei Requisiti** (uso esterno): lo scopo del documento è di esporre i requisiti del progetto a *stakeholderG*, Proponente e Committenti. Una descrizione più accurata sarà presente nella sezione "Scopo" del documento stesso;

- **Glossario** (uso esterno): il Glossario è un documento unico per tutti i documenti formali. Lo scopo di tale documento è raggruppare tutti i termini tecnici usati nei vari documenti formali per facilitarne la comprensione da parte di stakeholder, Proponenti e Committenti;
- **Manuale Utente** (uso esterno): lo scopo del documento è di fornire una guida, a supporto dell'utente finale, per l'installazione e l'esecuzione del prodotto collaudato;
- **Norme di Progetto** (uso interno): lo scopo del documento è di illustrare le regole con le quali il progetto viene sviluppato dai membri del team. Sono presenti regole per i processi primari, di supporto e organizzativi;
- **Piano di Progetto** (uso esterno): lo scopo del documento è di mostrare come il team gestisce le proprie risorse, come attribuisce responsabilità ed autorità e come il progetto viene distribuito nel tempo di calendario;
- **Piano di Qualifica** (uso esterno): lo scopo del documento è di illustrare le metodologie adottate per effettuare verifica e validazione in maniera tale da ottenere un prodotto di qualità, conforme ad aspettative e requisiti;
- **Studio di Fattibilità** (uso interno): lo scopo del documento è di mostrare i motivi per i quali il gruppo ha scelto il capitolato d'appalto. Il documento dovrà contenere inoltre le motivazioni per le quali il gruppo ha deciso di escludere gli altri capitolati proposti;
- **Verbali** (uso interno): sono documenti redatti da un segretario in occasione di incontri interni al gruppo o con altre entità esterne. Possono contenere un elenco di decisioni prese, in caso di incontro interno, oppure risposte a domande poste ai Committenti, in caso di incontro esterno.

3.1.8 Nome e formato dei file di documentazione

Ciascun documento deve avere un nome significativo ed un numero di versione (esempio: NomeDocumento_vX.Y.Z), ad eccezione dei verbali che non necessitano di controllo di versione. Le regole per il nome e la versione sono le seguenti:

- **Nome del documento**: il nome non deve contenere spazi e bisogna usare una lettere maiuscola per iniziare ogni parola;
- **Numero di versione del documento**: il numero di versione viene separato dal nome tramite un underscore (_) ed il significato della sua composizione è spiegato nella sezione 3.2.2.3.

I documenti devono essere memorizzati in repository con estensione *.texG* fino a quando non vengono approvati. Quando un documento viene approvato dal Responsabile di progetto, viene generato il file .pdf del documento. È stata presa questa decisione per

non produrre pdf per documenti che potenzialmente verranno modificati in seguito a delle verifiche.

3.1.9 Norme sul ciclo di vita dei documenti

Ogni documento può trovarsi in una delle seguenti tre fasi di ciclo di vita:

1. **Sviluppo**: lo sviluppo di un documento inizia da quando ha inizio la sua stesura e termina quando esso viene approvato. Il documento può passare alla fase di verifica da parte di un Verificatore se il Responsabile di Progetto ritiene che esso sia pronto per la verifica;
2. **Verifica**: la verifica di un documento deve essere eseguita da un Verificatore secondo le modalità espresse dal Piano di Qualifica per la documentazione. Al termine della verifica, il Responsabile, dopo averne esaminato accuratamente l'output, può decidere di approvare il documento oppure segnalare i problemi riscontrati agli sviluppatori dello stesso. In tal caso il documento rientra in fase di sviluppo per la correzione degli errori;
3. **Approvazione**: il documento viene approvato dal Responsabile di Progetto in seguito ad un esito di verifica positivo da parte dei Verificatori. Per segnalare l'importanza dell'approvazione di un documento, il suo numero di versione viene aggiornato dal Responsabile di progetto nelle modalità descritte dalla sezione 3.2.2.3.

3.1.10 Norme sulla struttura della documentazione

In questa sezione sono contenute tutte le regole per la struttura della documentazione.

3.1.10.1 Template

Per la formattazione dei documenti è stato utilizzato un *templateG*, costruito per unificare la struttura dei documenti in modo da renderli il più uniformi possibili. Il template è stato creato mediante il linguaggio *LaTeXG*.

3.1.10.2 Frontespizio

Il frontespizio è la prima pagina di ogni documento ed è composto dalle seguenti parti in ordine di apparizione:

- **Logo**: è il logo del gruppo MILCTdev;
- **Nome del documento**: indica il nome del documento che si sta visionando (es. Norme di Progetto o Analisi dei Requisiti);

- **Informazioni su gruppo e progetto**: sono indicati il nome del gruppo ed affianco il nome del capitolato scelto. Sotto queste informazioni è indicata la mail del gruppo;
- **Informazioni sul documento**: sotto forma di tabella sono indicate le informazioni principali del documento:
  1. **Versione**: indica la versione corrente del documento;
  2. **Redazione**: indica il nome dei membri del gruppo che si sono impegnati nella redazione del documento;
  3. **Verifica**: indica il nome dei membri del gruppo che si sono impegnati nella verifica del documento (Verificatori);
  4. **Approvazione**: indica il nome del componente del gruppo che si è impegnato nell'approvazione del documento (Responsabile di Progetto);
  5. **Uso**: specifica l'uso del documento, in particolare se si tratta di un documento interno oppure esterno;
  6. **Distribuzione**: indica il nome delle persone o dei gruppi alle quali il documento sarà distribuito.
- **Descrizione**: contiene una breve descrizione del documento che si sta visionando.

3.1.10.3 Diario delle modifiche

Il diario delle modifiche contiene, sotto forma di tabella, tutte le informazioni riguardanti le modifiche che sono state effettuate sul documento durante la sua stesura.
Il diario inizia nella seconda pagina di ogni documento. Ogni riga della tabella è composta da 5 colonne:

- **Descrizione**: riporta una piccola descrizione della modifica apportata al documento;
- **Autore**: indica il nome della persona che ha effettuato la modifica del documento;
- **Ruolo**: indica il ruolo della persona che ha effettuato la modifica al documento;
- **Data**: indica la data della modifica;
- **Versione**: indica la versione del documento. Le modalità tramite le quali questo numero deve essere incrementato sono spiegate nella sezione 3.2.2.3.

3.1.10.4 Indice

Ogni documento, ad eccezione dei verbali, deve avere un indice che ne permetta una lettura *ipertestualeG* e non per forza sequenziale. Nell'indice sono riportati capitoli, sottosezioni e sottosezioni di sottosezioni. La numerazione delle sezioni deve rispettare le seguenti regole:

1. **Capitolo**: la numerazione deve partire da uno ed essere del tipo "a." dove "a" è l'indice del capitolo;
2. **Sottosezione**: la numerazione deve partire da uno ed essere del tipo "a.b" dove "a" è l'indice del capitolo e "b" è l'indice della sottosezione del capitolo "a";
3. **Sottosezione di sottosezione**: la numerazione deve partire da uno ed essere del tipo "a.b.c" dove "a" è l'indice del capitolo, "b" è l'indice della sottosezione del capitolo "a" e "c" è l'indice della sottosezione della sottosezione "b" del capitolo "a".

Se nel documento sono presenti delle immagini deve essere stilato un indice contenente il nome delle immagini.

3.1.10.5 Contenuto principale

Tutte le pagine di ogni documento, ad eccezione del frontespizio, devono contenere un'intestazione ed un piè di pagina, costruiti tramite il template. L'intestazione e il piè di pagina sono separati dal contenuto principale da una linea orizzontale.
L'intestazione contiene:

- **Indirizzo e-mail**: il nome e l'indirizzo e-mail del gruppo sono situati a sinistra;
- **Nome documento**: il nome del documento, con la rispettiva versione, è posto a destra;

Il piè di pagina contiene:

- **Logo**: il logo del gruppo è situato a sinistra;
- **Indicazione pagina**: l'indicazione del numero di pagina è posta a destra.

3.1.10.6 Immagini

Se sono presenti immagini nel documento queste devono essere riportare in un indice separato. Le immagini devono essere inoltre centrate rispetto al documento e devono comprendere necessariamente una didascalia che ne descriva il contenuto.

3.1.10.7 Verbali

I verbali hanno una loro struttura, diversa dalle altre tipologie di documenti. Come gli altri documenti contengono il frontespizio, ma non contengono l'indice essendo documenti brevi. La struttura di ogni verbale interno o esterno deve essere la seguente:

- **Informazioni incontro**: questa sezione contiene informazioni riguardanti l'incontro tenuto, in particolare:
  1. **Luogo**: indica il luogo dove si è tenuto l'incontro. Potrebbe essere un'aula, in caso di incontro interno, oppure la sala riunioni di un'azienda, in caso di incontro esterno;
  2. **Data**: indica la data dell'incontro;
  3. **Ora**: indica l'ora di inizio incontro e l'ora di fine incontro, separate da un trattino;
  4. **Componenti interni**: indica i nomi dei componenti del gruppo che hanno preso parte all'incontro;
  5. **Componenti esterni**: indica i nomi delle figure esterne al gruppo che hanno preso parte all'incontro, questa voce è presente solo se si tratta di un incontro esterno.
- **Argomenti**: questa sezione contiene una breve descrizione di quali sono stati gli argomenti discussi durante l'incontro;
- **Riassunto incontro**: questa sezione descrive concretamente le decisioni prese, in caso di incontro interno, oppure le risposte a delle domande fatte, in caso di incontro esterno;
- **Riepilogo tracciamento decisioni**: poiché ogni decisione, così come ogni requisito, deve essere tracciabile (una decisione può portare alla definizione di un nuovo requisito, per cui è importante sapere dove e quando esso ha avuto origine) è importante tenere traccia delle decisioni prese tramite opportuni codici. La struttura del codice deve essere la seguente:
  1. **ID verbale**: è un identificativo del verbale. Può essere VI, se si tratta di un verbale interno, oppure VE, se si tratta di un verbale esterno;
  2. **Data**: la data, nel codice, è separata dal ID tramite un underscore. La data è scritta senza spazi tra giorno, mese ed anno e deve coincidere con la data dell'incontro (es. 28112017 indica il 28 Novembre 2017);
  3. **Numero decisione**: il numero della decisione deve partire sempre da uno ed essere incrementato per ogni decisione diversa presa. È separato dalla data tramite un punto.

Un esempio di codice per una decisione di verbale è il seguente: VI_28112017.1, che sta per decisione uno del verbale interno tenutosi in data 28 Novembre 2017.

3.1.11 Norme sulla stesura della documentazione

Questa sezione contiene le norme per scrivere la documentazione legata al progetto. Le seguenti regole devono essere rispettate.

3.1.11.1 Date

Le date devono essere scritte nel formato **GG-MM-AAAA**, dove GG rappresenta il giorno, MM il mese e AAAA l'anno.

3.1.11.2 Orari

Gli orari devono essere scritti nel formato **HH:MM**, dove HH rappresenta l'ora e MM rappresenta i minuti. I secondi non sono ritenuti importanti per lo scopo del progetto. Gli orari verranno principalmente utilizzati per indicare l'orario degli appuntamenti interni ed esterni nei verbali.

3.1.11.3 Riferimenti a ruoli

Ciascun ruolo deve essere indicato con la lettera iniziale maiuscola (es. Verificatore, Responsabile).

3.1.11.4 Elenchi puntati

Negli elenchi puntati ciascuna voce interna deve finire con un punto e virgola, mentre l'ultima voce dell'elenco deve terminare con un punto. Se l'elenco puntato rappresenta un elenco di definizioni, o punti salienti, dopo il nome del termine si inserisce ":" e poi si scrive la definizione del termine (es. Analisi dei requisiti: "..."). Il nome del termine deve essere in grassetto.

3.1.11.5 Riferimenti a termini nel Glossario

I termini all'interno del documento, che rientrano nel Glossario, vengono indicati con una *G* maiuscola a pedice ed il testo deve essere in corsivo. Deve essere marcata solo la prima occorrenza del temine.

3.1.11.6 Riferimenti a risorse esterne

Ciascun link o collegamento ad una risorsa esterna al documento deve essere indicato con caratteri di colore blu.

3.1.11.7 Riferimenti a documenti di progetto

I nomi dei documenti legati al progetto vengono indicati con le lettere maiuscole (es. Analisi dei Requisiti).

3.1.12 Strumenti a supporto della documentazione

Per la stesura di tutta la documentazione il gruppo utilizza il linguaggio LaTeX. Gli strumenti utilizzati per facilitare la scrittura della documentazione sono:

- **TexMaker**: è un editor LaTeX gratuito, moderno e multipiattaforma per sistemi Linux, Macosx e Windows che integra molti strumenti necessari per sviluppare documenti con LaTeX, in un'unica applicazione. TexMaker include supporto Unicode, controllo ortografico, auto-completamento e un visualizzatore pdf integrato con supporto synctex e modalità di visualizzazione continua.

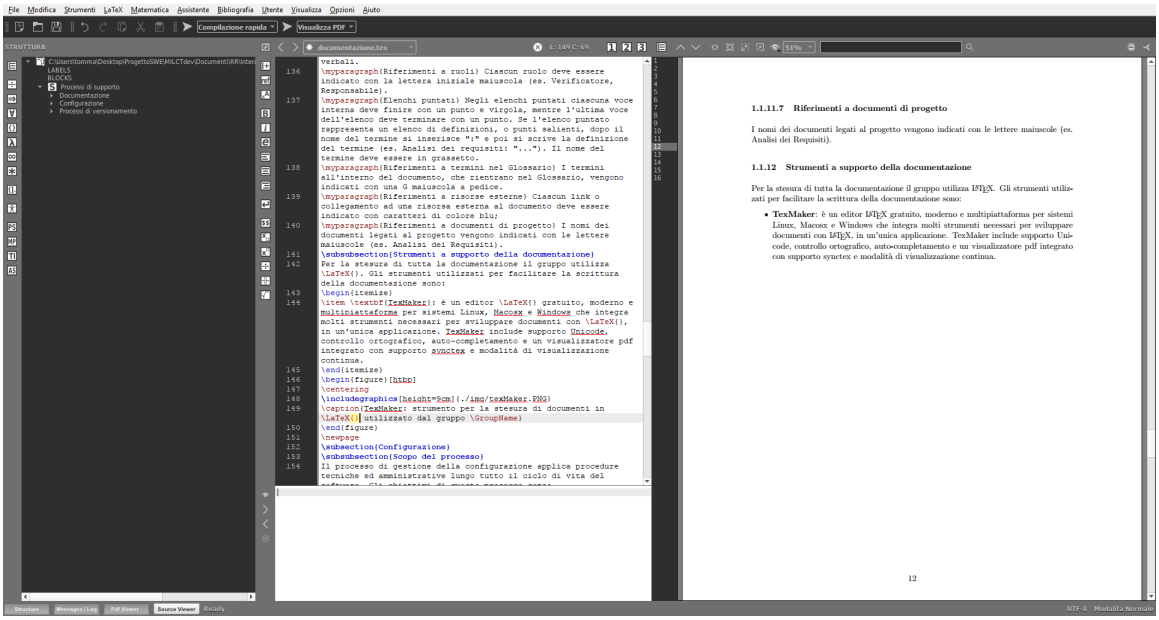

Figure 4: TexMaker: strumento per la stesura di documenti in LaTeX utilizzato dal gruppo MILCTdev

- **Script per la generazione di tabelle fonti-requisiti**: script scritto il linguaggio PHP che genera il codice LaTeX per la creazione delle tabelle fonti-requisiti e requisiti-fonti nell'Analisi dei Requisiti;
- **Script per la ricerca di termini di Glossario**: script scritto il linguaggio PHP che individua, dato un documento .tex, tutte le prime occorrenze di termini di Glossario in tale documento.

3.2 Configurazione

3.2.1 Scopo del processo

Il processo di gestione della configurazione applica procedure tecniche ed amministrative lungo tutto il ciclo di vita del software. Gli obiettivi di questo processo sono:

- identificare e definire gli elementi software in un sistema;
- controllare modifiche e rilasci degli elementi software;
- registrare e riportare lo stato degli elementi software e delle richieste di modifica;
- garantire completezza, correttezza e coerenza degli elementi software;
- controllare conservazione, gestione e consegna degli elementi software.

3.2.2 Processi di versionamento

Per il versionamento del software (durante il suo ciclo di vita) e della documentazione, si è utilizzato il software di controllo di versione *GitG*. In particolare si è utilizzato il servizio GitHub.

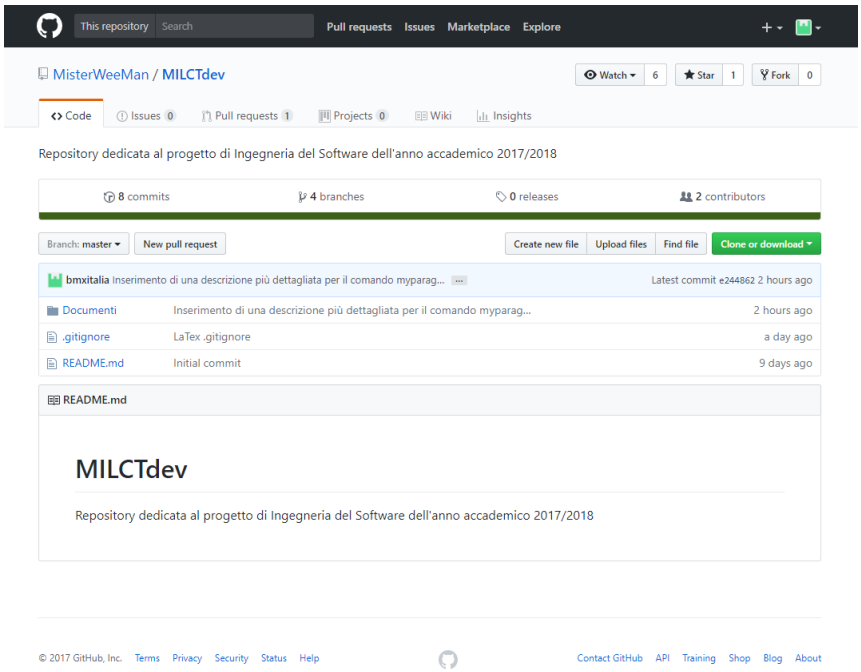

Figure 5: GitHub: strumento per controllo di versione utilizzato dal gruppo MILCTdev

3.2.2.1 Struttura della copia locale del repository

La copia locale della repository è composta da tre "alberi" mantenuti da Git. Il primo è la directory di lavoro che contiene i files attuali su cui si sta lavorando. Il secondo è l'index (*stageG*) che fa da spazio di transito per i files. Per finire c'è l'*headG*, che punta all'ultimo *commitG* fatto.

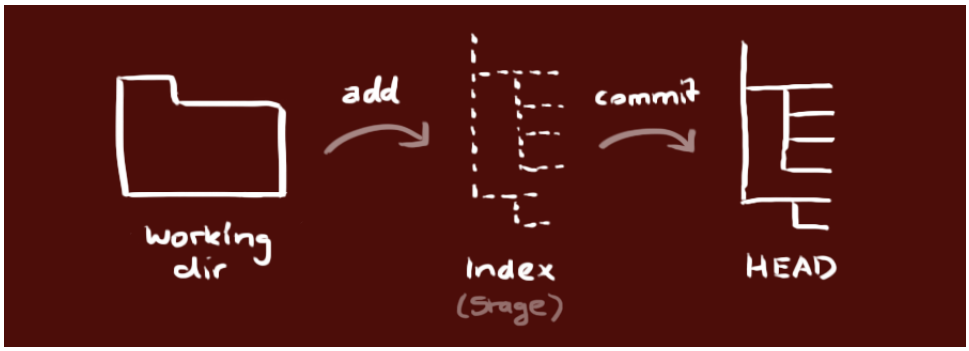

Figure 6: Struttura della repository locale, divisa nei tre alberi: working dir, stage e head

3.2.2.2 Comandi Git maggiormente utilizzati

Questa sezione contiene una serie di comandi Git che permetterà ai membri del team di progetto di interagire con GitHub:

- ***git clone***: comando che permette di clonare il repository remoto su GitHub in locale. La sintassi è *git clone indirizzoRepository* dove indirizzoRepository è l'indirizzo raggiungibile tramite il menù a tendina "Clone or Download" su GitHub;
- ***git add nomeFile***: comando che aggiunge il file alla lista dei file tracciati da Git, contenuti nell'index. Il comando deve essere utilizzato per aggiungere nuovi file o per proporre la modifica di file già esistenti in una versione successiva. Se si modifica un file e non si esegue questo comando, le modifiche non verranno apportate nella versione successiva. Il comando può essere usato per aggiungere una cartella oppure si può eseguire *git add .* per aggiungere tutti i file, comprese le sotto cartelle;
- ***git pull***: comando che permette di aggiornare il contenuto del repository locale con il contenuto di quello remoto. In questo modo eventuali modifiche attuate da altre persone verranno riportate sul repository locale e si può lavorare sui file più recenti;

- ***git status***: comando che visualizza lo stato dei file, dividendoli in tre gruppi:
  1. **File non tracciati**: mostra i file che non sono mai stati inseriti nel controllo versione di Git;
  2. **File modificati ma non aggiornati**: mostra i file modificati, rispetto al commit precedente, che non sono stati aggiunti tramite *git add* e che quindi non verranno inseriti nel prossimo commit;
  3. **File modificati pronti per il commit**: mostra i file modificati, rispetto al commit precedente, che sono stati aggiunti tramite *git add* e che quindi sono pronti per il commit.
- ***git diff***: mentre *git status* mostra i file che sono stati modificati, per entrare nei dettagli delle righe modificate è necessario utilizzare *git diff*. Il comando può essere utilizzato in due modalità:
  1. ***git diff***: mostra le righe che sono state cambiate nei file non ancora preparati per la commit, confrontati con la copia presente nell'ultima commit;
  2. ***git diff –staged***: mostra le righe che sono state cambiate nei file preparati per la commit, confrontati con la copia presente nell'ultima commit;
- ***git commit -m "Messaggio commit"***: comando che permette di rendere definitivi i file aggiunti o modificati tramite il comando *git add*. Se non si lancia il comando *git commit* e si esegue *git push*, non verranno caricate eventuali modifiche o file aggiunti al repository locale sul repository remoto. *-m* è l'opzione per inserire un messaggio significativo (obbligatorio) per far comprendere ai membri del team la tipologia di modifiche che sono state effettuate;
- ***git log***: comando che permette di visualizzare lo storico dei commit;
- ***git checkout***: comando che permette di fare cose differenti a seconda di come viene utilizzato:
  1. ***git checkout -b nomeBranch***: crea un nuovo *branchG* chiamato "nomeBranch". Una volta creato il branch, tutti i comandi che verranno lanciati saranno eseguiti su quel branch;
  2. ***git checkout nomeBranch***: permette di spostarsi sul branch "nomeBranch";
- ***git merge nomeBranch***: comando che permette di riportare le modifiche apportate a "nomeBranch" nel branch attualmente selezionato. Il comando deve essere lanciato dal branch *masterG*;
- ***git push***: comando che aggiorna il repository remoto sovrascrivendolo con il repository locale;
- **configurazioni utente**: se si tratta della prima volta che si utilizza Git allora al primo tentativo di commit verrà lanciato un errore relativo al fatto che non si sono

settati il nome utente e la e-mail. Per far questo si utilizzano i comandi:

1. ***git config –global user.name "nomeUtente"***: comando che permette di configurare Git con il nostro nome. In questo modo ogni qualvolta che si effettuerà una commit, il nostro nome verrà associato alla commit;
2. ***git config –global user.email "e-mail"***: comando identico al precedente ma che permette di configurare l'indirizzo e-mail.

3.2.2.3 Codici di versione della documentazione

Come illustrato nella sezione 3.1.8. il nome di un documento presenza il codice _vX.Y.Z che rappresenta la versione del documento. Il significato delle lettere è il seguente:

- **Z**: è un numero che viene incrementato di uno ogni volta che viene apportata una modifica al documento. Questo numero viene incrementato da colui che ha apportato la modifica;
- **Y**: è un numero che viene incrementato di uno ogni volta che il documento è stato verificato da un Verificatore. É quindi compito del Verificatore incrementare questo numero, nessun altro membro del team può farlo;
- **X**: è un numero che viene incrementato di uno ogni volta che il documento viene approvato dal Responsabile di Progetto. É compito del Responsabile di Progetto incrementare questo numero. Ogni volta che un documento raggiunge l'approvazione può essere generato il file pdf del documento.

Ogni volta che un numero viene incrementato, tutti i numeri alla sua destra devono essere settati a zero.

3.3 Garanzia di qualità

In questa sezione vengono definite le norme e la struttura delle metriche e degli obiettivi di qualità, descritti nel *Piano di Qualifica v0.0.0*. Metriche ed obiettivi devono essere decise dagli Amministratori in collaborazione con i Verificatori.

3.3.1 Notazione per la classificazione

Sia gli obiettivi di qualità che le metriche devono essere classificati secondo la seguente notazione:

{**Classe**}{**Tipo**}{**Oggetto**}*{**codice_identificativo**} : {**Nome**}

- **Classe:** indica se si tratta di un obiettivo di qualità o una metrica. Può essere:
  - **O:** per un obiettivo;
  - **M:** per una metrica.
- **Tipo:** stabilisce se riguarda un prodotto o un processo e può assumere i valori:
  - **PD:** per gli obiettivi di prodotto;
  - **PC:** per gli obiettivi di processo.
- **Oggetto:** nel caso di obiettivi o metriche di prodotto, indica se si riferisce alla parte software oppure ad un documento. Può essere:
  - **D:** per i documenti;
  - **S:** per il software.
- **codice_identificativo:** codice numerico incrementale necessario per l'identificazione;
- **Nome:** titolo che descrive l'obiettivo o la metrica.

3.4 Verifica

3.4.1 Scopo del processo

Il processo di verifica è un processo che determina se i prodotti software di un'attività soddisfano i requisiti o le condizioni imposte ad essi nelle attività precedentemente svolte. Per costi ed efficacia delle prestazioni, la verifica dovrebbe essere integrata, quanto prima possibile, nel processo (ad esempio sviluppo, funzionamento o manutenzione) che la impiega. Questo processo può includere analisi, revisione e prova.

3.4.2 Analisi

3.4.2.1 Analisi Statica

É una tecnica che studia il codice e la documentazione e verifica la conformità alle regole, l'assenza di difetti e la presenza di proprietà positive. Questa tecnica non richiede esecuzione del prodotto software in alcuna sua parte per cui è essenziale finché il sistema non è completamente disponibile.

É attuabile tramite due tecniche:

- **Walkthrough:** attività che richiede la collaborazione di più persone per effettuare una lettura a largo spettro di tutto il documento e il codice in esame, con lo scopo di rivelare la presenza di difetti. Ogni difetto riscontrato verrà discusso tra Verificatore e autore; è importante che ci sia una terza figura che mantenga il controllo della discussione, detta arbitro. Per ogni discussione deve essere redatta una lista di controllo con i difetti rilevati e le decisioni prese.

  Le fasi del walkthrough sono le seguenti:

  - pianificazione;
  - lettura;
  - discussione;
  - correzione dei difetti.

  Ognuna delle precedenti fasi elencate deve essere documentata;

- **Inspection:** attività normalmente svolta da una sola persona che effettua una lettura mirata e strutturata del documento, volta a localizzare gli errori segnalati nella lista di controllo; tramite controlli ripetuti verrà progressivamente ampliata per rendere più efficace l'attività di inspection. Le fasi di inspection sono:

  - pianificazione;
  - definizione della lista di controllo;

  - lettura;
  - correzione dei difetti.

  Ognuna delle precedenti fasi elencate deve essere documentata.

Essendo walkthrough una tecnica poco efficiente verrà impiegata principalmente nella prima parte del progetto. Nelle fasi successive si utilizzerà la lista di controllo prodotta da walkthrough per effettuare inspection.

3.4.2.2 Analisi Dinamica

É una tecnica di analisi del prodotto software che richiede l'esecuzione del programma. Vengono effettuati dei test su parti del sistema per verificare che ognuna di esse produca il risultato desiderato.
Prima di eseguire un qualsiasi test è necessario conoscere la precondizione e postcondizione del codice analizzato per poterne decretare l'esito finale.

Ci sono diversi tipi di test:

- **Test di Unità:** va ad isolare la parte più piccola di software testabile nell'applicazione, chiamata unità, per stabilire se essa funziona esattamente come previsto. Per effettuare test di unità è necessaria la scrittura di codice fittizio. Può essere di due tipologie:
  - **Driver**: è un software che guida il test d'unità sostituendo l'unità chiamante per testare l'unità chiamata;
  - **Stub**: è un software che simula il comportamento delle unità chiamate per testare un'unità chiamante.

  Ogni unità deve essere sottoposta a test prima di poter essere integrata con quelle già testate;
- **Test di Integrazione:** prevede la combinazione di unità già testate in un unico componente per verificare che il loro funzionamento integrato abbia esito atteso. Richiede una strategia di integrazione incrementale che può essere di due tipologie:
  - **Bottom-up**: si sviluppano e si integrano prima le unità con minore dipendenza funzionale e maggiore utilità e poi si risale l'albero delle dipendenze;
  - **Top-down**: si sviluppano prima le unità più esterne poste sulle foglie dell'albero delle dipendenze e poi si scende l'albero;
- **Test di Regressione:** questo test deve essere eseguito ad ogni modifica ad un implementazione del sistema. Vengono ripetuti sul codice modificato i test precedentemente utilizzati, per verificare che le modifiche effettuate non abbiano alterato elementi precedentemente funzionanti. I contenuti di tale test devono essere decisi nel momento in cui si approvano modifiche al software;

- **Test di Sistema:** ha inizio con il completamento del test d'integrazione e verifica il comportamento dinamico del sistema completo rispetto ai requisiti software;
- **Test di Accettazione:** prevede il collaudo del prodotto in presenza della Proponente. Il superamento del test permette il rilascio ufficiale del prodotto sviluppato.

3.4.3 Strumenti utilizzati

- **Correttore automatico TeXMaker (analisi statica):** per la verifica ortografica in tempo reale viene utilizzato questo strumento integrato in TexMaker che sottolinea in rosso le parole errate secondo la lingua italiana;
- **Julia**: Julia è un analizzatore statico per Java basato sulla tecnica scientifica dell'interpretazione astratta che garantisce la precisione e l'affidabilità dei suoi risultati.

4 Processi organizzativi

I processi organizzativi sono i seguenti:

- processo di gestione;
- processo di infrastruttura;
- processo di miglioramento;
- processo di formazione.

Le attività e i compiti in un processo organizzativo sono di responsabilità dell'organizzazione che fa uso del processo. L'organizzazione garantisce l'esistenza e la funzionalità del processo.

4.1 Gestione organizzativa

Il processo di gestione contiene le attività e i compiti generici che possono essere utilizzati da qualunque parte che deve gestire i rispettivi processi. Il manager è responsabile della gestione di prodotto, di progetto e dei compiti dei processi applicabili, come i processi di acquisizione, fornitura, sviluppo, funzionamento, manutenzione e di supporto. Questo processo consiste nelle seguenti attività:

- iniziazione e definizione dello scopo;
- pianificazione;
- esecuzione e controllo;
- verifica e valutazione;
- chiusura.

4.1.1 Comunicazione

Questa sezione è dedicata all'illustrazione delle modalità di comunicazione che verranno adottate dal gruppo MILCTdev durante l'intera durata del progetto. Le comunicazioni potranno essere interne al gruppo, o potranno coinvolgere soggetti esterni ed esso quali Proponente, Committenti o altri stakeholder.

4.1.1.1 Comunicazioni interne

Andando ad illustrare e normare le conversazioni interne al team, va fatto presente che la maggior parte di esse avranno luogo tramite lo strumento di collaborazione aziendale *SlackG*.

Slack è stato preferito ad altri software di messaggistica per la possibilità di creare canali tematici e per l'alta integrabilità con altri servizi utilizzati dal team nel corso di questo progetto. All'interno del workspace Slack, i membri del gruppo dovranno comunicare nel rispetto dei temi dei canali creati, avvalendosi anche di features quali @everyone, @channel, @NomeUtente per l'invio di notifiche rispettivamente a tutti i soggetti interni al gruppo, a quelli che partecipano alla conversazione o a un utente specifico.

Il team ha quindi deciso di dividere le comunicazioni in canali abbastanza specifici da poter permettere un corretto scambio di idee ed informazioni, inerenti solamente a determinate aree del progetto. I canali presenti nel workspace sono quindi:

- **canali dedicati a servizi utilizzati**: questi sono spazi dedicati ad integrazioni con altri servizi quali ad esempio *AsanaG* e GitHub. Viene creato un canale dedicato ad ogni integrazione;
- **canali dedicati alla documentazione**: questi sono spazi dedicati alla documentazione scritta dal team. Anche qui viene creato un canale per ogni documento da redigere, ad esempio sono presenti canali quali "norme di progetto" e "piano di qualifica". All'interno di ogni spazio verranno discusse sia le attività di sviluppo, che di verifica ed approvazione;
- **canali dedicati ad argomenti specifici**: dal canale riguardante incontri e comunicazioni con soggetti esterni, al canale dedicato al materiale informativo, a canali per la discussione dei capitolati d'appalto o della scelta di nome e logo del gruppo;
- **general e random**: canali di default, dedicati a comunicazioni non riferibili ad un singolo processo o ad una singola attività.

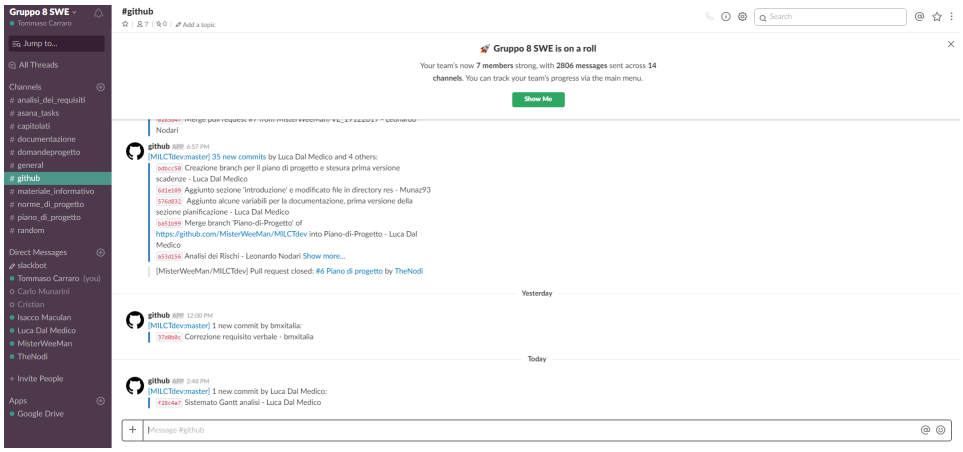

Figure 7: Slack: strumento per la comunicazione del gruppo

4.1.1.2 Comunicazioni esterne

Le comunicazioni intraprese con soggetti esterni al gruppo sono di competenza del Responsabile di Progetto che utilizzerà la casella di posta elettronica creata in fase di istituzione del team MILCTdev: milctdev.team@gmail.com.

Il Responsabile di Progetto è quindi il soggetto incaricato dal gruppo per comunicare con i soggetti esterni, rappresentati da:

- **Kirey Group**, nella figura di Proponente, raggiungibile all'indirizzo di posta elettronica: stefano.bertolin@iks.it;
- **Prof. Tullio Vardanega e Prof. Riccardo Cardin**, nella figura di Committenti, raggiungibili agli indirizzi di posta elettronica istituzionali.

Tutti i membri del gruppo dovranno però essere sempre aggiornati sulle conversazioni avvenute. Il compito di notificarli spetta al Responsabile di Progetto utilizzando l'apposito canale tematico all'interno del workspace Slack.

4.1.2 Incontri del team

Le riunioni del team, interne o esterne, verranno sempre organizzate in accordo con tutto il gruppo, nel rispetto delle modalità di comunicazione illustrate sezione 4.1.1. L'organizzazione di esse spetta al Responsabile di Progetto.

4.1.2.1 Frequenza degli incontri

Esaminati gli impegni dei vari membri del gruppo si è deciso di fissare almeno un incontro al mese, nel quale è richiesta la presenza di tutti i membri.

4.1.2.2 Verbali di riunione

Ad ogni riunione corrisponderà un verbale, il cui compito di redazione sarà onere del Segretario, nominato a turno dal Responsabile di Progetto, il quale dovrà illustrare l'ordine del giorno e tenere nota delle discussioni avvenute in tale sede.

4.1.2.3 Decisioni e vincoli

Il numero minimo di membri richiesto per rendere valido un incontro è cinque. Tale numero impone una collaborazione tra gli elementi del team ed è sufficientemente grande a garantire equità nelle decisioni prese. Nel caso di riunioni straordinarie, il "quorum" si abbassa a tre ma le decisioni prese dovranno essere riviste nell'incontro successivo.

Ogni decisione verrà presa per maggioranza e, nel caso di parità di voti, la parola spetterà al Responsabile di Progetto che, da prima si presterà nel ruolo di mediatore e, nel caso di fallimento delle trattative, potrà prendere lui stesso una decisione.

4.1.3 Ruoli di progetto

I sette membri del MILCTdev team si suddivideranno ruoli che corrispondono alle omonime figure aziendali. Vista la natura didattica del progetto, nell'arco dell'intera durata dello stesso, ogni soggetto coprirà almeno una volta ognuno dei ruoli che verranno di seguito illustrati. Ogni membro dovrà svolgere le attività assegnate al ruolo ricoperto, come organizzato e stabilito dal documento *Piano di Progetto v1.0.0*. Al fine di non minare la qualità di prodotto all'interno del *Piano di Progetto v1.0.0* verranno pianificati dei turni tali da non creare conflitti d'interesse. Un esempio di conflitto d'interesse potrebbe essere dover verificare qualcosa che si era precedentemente scritto.

I ruoli che verranno ricoperti dai membri del team sono illustrati nelle sezioni sottostanti.

4.1.3.1 Responsabile di Progetto

Il Responsabile di Progetto, anche chiamato Project Manager, è una figura di estrema importanza all'interno del team in quanto su di lui ricadono responsabilità di pianificazione, gestione, controllo e coordinamento. Il Project Manager rappresenta inoltre MILCTdev all'esterno del gruppo in quanto intrattiene rapporti con Committente e Proponente. In sunto, egli:

- gestisce, controlla e coordina risorse, umane e non;
- gestisce, controlla e pianifica le attività di progetto;
- analizza e gestisce i rischi;
- approva documenti.

Questa figura è presente durante tutta la durata del progetto.

4.1.3.2 Amministratore

L'Amministratore è una figura di supporto che mette a disposizione strumenti e tecnologie per perseguire qualità ed efficienza all'interno dell'ambiente lavorativo. Egli quindi non opera scelte gestionali ma:

- monitora e lavora per il perseguimento della qualità di prodotto;
- ricerca strumenti per l'automazione di attività, processi e compiti;

- supervisiona e si adopera per la gestione del versionamento e l'archiviazione della documentazione;
- controlla versioni e configurazioni del prodotto software;
- norma l'utilizzo degli strumenti utilizzati durante il progetto.

4.1.3.3 Analista

L'Analista è una figura che non sarà sempre presente durante il progetto, ma di estrema importanza per i compiti svolti. Egli ha l'onere di comprendere appieno il dominio del problema e, tramite le sue azioni, vincolerà anche l'operato di alcuni ruoli, quali, ad esempio, i Progettisti. I suoi compiti sono:

- studiare il dominio del problema ed il problema stesso, definendone complessità e requisiti;
- redigere documenti quali Analisi dei Requisiti e Studio di Fattibilità.

4.1.3.4 Progettista

Il Progettista è colui che, partendo dallo studio e dai vincoli posti dall'Analista, definisce una soluzione che soddisfi i requisiti. Egli è responsabile degli aspetti tecnologici e tecnici del progetto e dovrà:

- produrre una soluzione attuabile al problema;
- sviluppare un'architettura che sfrutti soluzioni note ed ottimizzate che portino ad un prodotto stabile e manutenibile.

4.1.3.5 Programmatore

Il Programmatore è la figura che provvederà alla codifica della soluzione, studiata e spiegata dal Progettista. Egli in particolar modo esegue queste operazioni:

- codifica la soluzione descritta dal Progettista nel rispetto di documenti, tra i quali le Norme di Progetto;
- crea o gestisce componenti di supporto per la verifica e la validazione del codice.

Questa figura ha inoltre il compito di manutenere il codice del prodotto e redigere un eventuale Manuale Utente.

4.1.3.6 Verificatore

Il Verificatore è una figura che è presente sin dalle prime fasi del progetto e lo sarà per tutto il ciclo di vita del software. Egli si focalizza sui processi e, grazie alla sua conoscenza delle Norme di Progetto, garantirà che essi siano conformi alle norme e alle attese. Il Verificatore controlla quindi che:

- ogni stadio del ciclo di vita del prodotto sia conforme al *Piano di Qualifica v0.0.0*;
- ogni processo sia eseguito nel rispetto delle *Norme di Progetto v1.0.0*.

4.1.4 Ambiente di lavoro

In questa sottosezione verranno illustrati gli strumenti utilizzati dal team durante la realizzazione del progetto.

4.1.4.1 Strumenti di comunicazione

Per quanto riguarda la parte relativa alla comunicazione, il team utilizzerà strumenti quali Slack e *GmailG*, già citati nella sezione 4.1.1. Dello strumento di collaborazione aziendale, Slack, si è già discusso in precedenza.; il secondo servizio invece è Gmail, famoso servizio di posta elettronica fornito da Google, che viene usato dal team nell'intraprendere conversazioni con soggetti esterni al gruppo.

4.1.4.2 Strumenti di condivisione

Un altro servizio utilizzato dal team è *Google DriveG*, servizio cloud based, per il quale è presente un'integrazione Slack, che permette al gruppo un rapido scambio di documenti e file.

4.1.4.3 Strumenti di coordinamento

Il sistema adottato per la coordinazione dei membri del team, mediante l'assegnazione di *taskG*, è Asana. Alcune delle funzionalità che questo strumento offre sono:

- creare un task, eventualmente inseribile in una "sezione" o all'interno di un altro task;
- indicare una persona a cui viene assegnato il task/sottotask;
- indicare una data di scadenza del task, andando così a popolare un calendario utile al coordinamento;
- inserire una descrizione del task e creare una conversazione relativa al task stesso.

Con il passaggio alla versione pro, per il quale il team ha provveduto ad effettuare richiesta, le possibilità si amplieranno e permetteranno anche la creazione di propedeuticità tra i task. Questo strumento è stato inoltre integrato in un apposito canale tematico all'interno del workspace Slack, per facilitare la notifica di nuovi eventi.

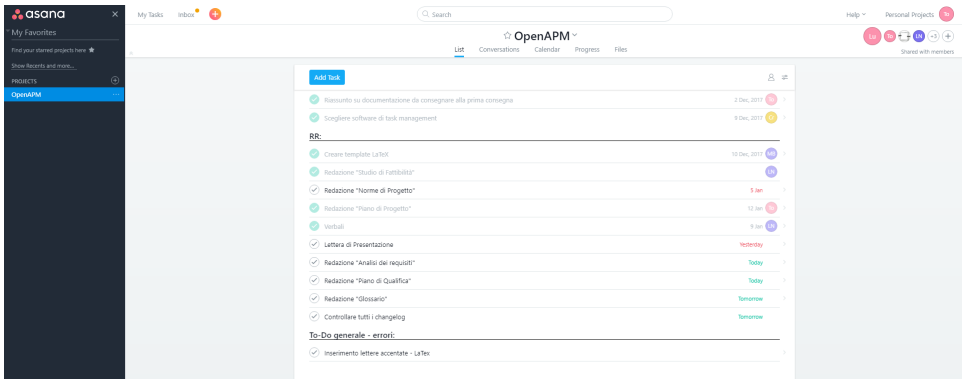

Figure 8: Asana: strumento utilizzato dal gruppo per il coordinamento del progetto didattico

4.1.4.4 Strumenti di versionamento

Per il salvataggio e versionamento dei file il team ha scelto l'utilizzo di repository su GitHub, il quale si basa sul sistema di versionamento Git. La scelta è caduta su questo servizio, data la familiarità di diversi membri del gruppo con lo stesso.

La repository contenente la documentazione, ha la seguente struttura (cartelle), alla quale i membri dovranno attenersi:

- **LatexLayout**: contenente file per l'agevolazione della scrittura e della creazione dei documenti in LaTeX;
- **NOMEREVISIONE**: all'interno di queste cartelle saranno presenti documenti, suddivisi nelle directory INTERNI/ESTERNI, che ospitano i file LaTeX e .pdf di ogni singolo tipo di documento.

Anche questo strumento è stato integrato nel workspace Slack, per favorire la notifica di nuovi eventi al gruppo.

4.1.4.5 Strumenti di grafica

Per la creazione di elementi grafici sono stati utilizzati i seguenti strumenti:

- **Diagrammi di Gantt**: è stato scelto lo strumento *Gantt ProjectG*, software gratuito per la creazione di diagrammi contenenti task e *milestoneG*;

- **Istogrammi e diagrammi a torta**: è stato scelto lo strumento Microsoft Office Excel 2016, il quale permette di creare diagrammi dinamici in maniera veloce ed intuitiva;
- **Diagrammi UML**: è stato scelto il software DIA che permette di creare diagrammi UML di attività, dei casi d'uso, di classe e di sequenza.

4.1.4.6 Sistemi operativi

Data la mancata presenza di requisiti che impongano restrizioni sul sistema operativo da utilizzare, i membri del team potranno indistintamente usare sistemi Windows, MacOSX o Linux.

4.2 Formazione del team

Per la formazione del team è stato deciso che ogni membro studierà individualmente per essere preparato alla realizzazione delle varie parti del progetto. Saranno attribuiti dei task relativi allo studio di materiale complesso necessario allo svolgimento del progetto. Ad esempio un task potrebbe essere: imparare ad utilizzare il linguaggio LaTeX.