

KALEIDOSCODE  
SWEDESIGNER  
SOFTWARE PER DIAGRAMMI UML

NORME DI PROGETTO V2.0.0



Informazioni sul documento

---

<b>Versione</b>	2.0.0
<b>Data Redazione</b>	01/03/2017
<b>Redazione</b>	Sovilla Matteo
<b>Verifica</b>	Pezzuto Francesco
<b>Approvazione</b>	Pace Giulio
<b>Uso</b>	Interno
<b>Distribuzione</b>	<i>KaleidosCode</i>

kaleidos.codec6@gmail.com

## Diario delle Modifiche

Versione	Data	Autore	Descrizione
2.0.0	07/05/2017	Pace Giulio	Approvazione del documento
1.1.0	06/05/2017	Pezzuto Francesco	Verifica del documento
1.0.4	05/05/2017	Sovilla Matteo	Correzione dopo incontro con il prof. Vardanega
1.0.2	24/04/2017	Sovilla Matteo	Correzione dopo RR, aggiunta norme sulla gestione dei task e sui commit GitHub e ampliamento delle sezioni progettazione e strumenti utilizzati
1.0.0	02/04/2017	Sanna Giovanni	Approvazione documento
0.2.1	02/04/2017	Sovilla Matteo	Aggiunta immagini e controllo link
0.2.0	30/03/2017	Sovilla Matteo	Verifica e correzione
0.1.2	28/03/2017	Pezzuto Francesco	Riorganizzazione dei contenuti, aggiunta strumenti, incrementata sezione Analisi dei requisiti, incrementata sezione Codifica, incrementata sezione Documentazione, aggiunti riferimenti informativi, aggiunte/sistematate sezioni in Processi organizzativi aggiunta Lista di controllo con errori trovati finora
0.1.0	07/03/2017	Sovilla Matteo	Verifica e correzione
0.0.8	05/03/2017	Pace Giulio	Stesura sezione Processi organizzativi
0.0.7	05/03/2017	Bonolo Marco	Stesura sezione Processo di fornitura
0.0.6	04/03/2017	Bonolo Marco	Stesura sezioni Progettazione, Codifica in Processo di sviluppo

Versione	Data	Autore	Descrizione
0.0.5	03/03/2017	Pezzuto Francesco	Stesura sezione Analisi dei requisiti in Processo di sviluppo e creazione template per documento
0.0.4	02/03/2017	Bonato Enrico	Stesura sezione Documentazione in Processi di supporto
0.0.3	01/03/2017	Bonolo Marco	Stesura sezione Riunioni in Processi organizzativi
0.0.2	01/03/2017	Pezzuto Francesco	Stesura Introduzione e sezione Comunicazioni in Processi organizzativi
0.0.1	01/03/2017	Pace Giulio	Creazione scheletro del documento e stesura parziale dei documenti

# Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	Scopo del documento . . . . .	1
1.2	Scopo del prodotto . . . . .	1
1.3	Glossario . . . . .	1
1.4	Riferimenti utili . . . . .	1
1.4.1	Riferimenti normativi . . . . .	1
1.4.2	Riferimenti informativi . . . . .	1
<b>2</b>	<b>Processi primari</b>	<b>3</b>
2.1	Fornitura . . . . .	3
2.1.1	Scopo . . . . .	3
2.1.2	Risultati . . . . .	3
2.1.3	Descrizione . . . . .	3
2.1.4	Identificazione opportunità . . . . .	3
2.1.5	Accordo contrattuale . . . . .	4
2.1.6	Esecuzione del contratto . . . . .	4
2.1.7	Consegna e supporto del prodotto e/o servizio . . . . .	4
2.1.8	Chiusura . . . . .	4
2.2	Sviluppo . . . . .	5
2.2.1	Scopo . . . . .	5
2.2.2	Aspettative . . . . .	5
2.2.3	Descrizione . . . . .	5
2.2.4	Analisi dei requisiti . . . . .	5
2.2.5	Progettazione . . . . .	7
2.2.6	Codifica . . . . .	9
<b>3</b>	<b>Processi di supporto</b>	<b>13</b>
3.1	Documentazione . . . . .	13
3.1.1	Descrizione . . . . .	13
3.1.2	Approvazione dei documenti . . . . .	13
3.1.3	Stesura dei documenti . . . . .	13
3.1.4	Struttura dei documenti . . . . .	14
3.1.5	Norme tipografiche . . . . .	16
3.1.6	Componenti grafiche . . . . .	19
3.1.7	Classificazione documenti . . . . .	19
3.1.8	Versionamento . . . . .	20
3.1.9	Verbale . . . . .	20
3.1.10	Glossario . . . . .	21
3.2	Verifica . . . . .	22
3.2.1	Scopo del processo . . . . .	22
3.2.2	Attività . . . . .	22
3.3	Validazione . . . . .	23
3.3.1	Scopo del processo . . . . .	23
3.3.2	Attività . . . . .	23
3.4	Strumento di versionamento . . . . .	23

3.4.1	Norme di commit . . . . .	24
3.5	Altri strumenti . . . . .	24
3.5.1	Google Drive . . . . .	24
3.5.2	ProjectLibre . . . . .	25
3.5.3	L <sup>A</sup> T <sub>E</sub> X <sub>G</sub> . . . . .	26
3.5.4	Strumenti per il controllo ortografico . . . . .	26
3.5.5	Strumenti per i diagrammi UML . . . . .	26
3.5.6	SWEgo . . . . .	26
3.5.7	Zapier . . . . .	26
3.5.8	Strumenti per l'integrazione continua . . . . .	27
3.5.9	Strumenti per la validazione W3C <sub>G</sub> . . . . .	27
3.5.10	Script . . . . .	27
<b>4</b>	<b>Processi organizzativi</b>	<b>28</b>
4.1	Scopo . . . . .	28
4.2	Aspettative . . . . .	28
4.3	Gestione organizzativa . . . . .	28
4.3.1	Ruoli di progetto . . . . .	28
4.4	Strumento di coordinamento . . . . .	30
4.4.1	Norme di utilizzo . . . . .	30
4.5	Comunicazioni . . . . .	32
4.5.1	Esterne . . . . .	32
4.5.2	Interne . . . . .	32
4.6	Composizione e-mail . . . . .	32
4.6.1	Destinatario . . . . .	32
4.6.2	Mittente . . . . .	33
4.6.3	Oggetto . . . . .	33
4.6.4	Corpo . . . . .	33
4.6.5	Allegati . . . . .	33
4.7	Composizione di conversazioni su Slack . . . . .	33
4.8	Riunioni . . . . .	33
4.8.1	Frequenza . . . . .	33
4.8.2	Convocazione riunione . . . . .	34
4.8.3	Svolgimento riunione . . . . .	35
<b>A</b>	<b>Lista di controllo</b>	<b>36</b>
A.1	Documentazione . . . . .	36
A.2	Diagrammi UML . . . . .	36

## Elenco delle figure

1	Schermata di presentazione di Google Drive . . . . .	25
2	Logo di ProjectLibre . . . . .	25
3	Schermata di tracciamento requisiti in SWEgo . . . . .	27
4	Lista di Task in Asana . . . . .	30
5	Schermata principale dell'estensione Instagantt per Asana . . . . .	31

# 1 Introduzione

## 1.1 Scopo del documento

Questo documento definisce le norme che i membri del gruppo *KaleidosCode* adotteranno nello svolgimento del progetto *SWEDesigner*.

Tutti i membri del gruppo sono tenuti a leggere il documento e a seguirne le norme descritte per uniformare il materiale prodotto, ridurre il numero di errori e migliorare l'efficienza.

In particolare verranno definite norme riguardanti:

- Interazioni tra i membri del gruppo;
- Stesura di documenti e convenzioni;
- Modalità di lavoro durante le varie fasi del progetto;
- Ambiente di lavoro.

## 1.2 Scopo del prodotto

Lo scopo del progetto è la realizzazione di un software di costruzione di diagrammi UML<sub>G</sub> con la relativa generazione di codice Java<sub>G</sub> e Javascript<sub>G</sub> utilizzando tecnologie web.

## 1.3 Glossario

Al fine di evitare ogni ambiguità di linguaggio e massimizzare la comprensione dei documenti, i termini tecnici, di dominio, gli acronimi e le parole che necessitano di essere chiarite sono riportate nel documento *Glossario v2.0.0*.

La prima occorrenza di questi vocaboli è marcata da una “G” maiuscola in pedice.

## 1.4 Riferimenti utili

### 1.4.1 Riferimenti normativi

- ISO<sub>G</sub> 12207:  
[https://en.wikipedia.org/wiki/ISO/IEC\\_12207](https://en.wikipedia.org/wiki/ISO/IEC_12207) (03/03/2017).

### 1.4.2 Riferimenti informativi

- Specifiche UTF-8<sub>G</sub>:  
<http://www.unicode.org/versions/Unicode6.1.0/ch03.pdf> (02/04/2017);
- Capitolato<sub>G</sub> d'appalto:  
<http://www.math.unipd.it/~tullio/IS-1/2016/Progetto/C6.pdf> (02/04/2017);
- Notazione CamelCase:  
[https://en.wikipedia.org/wiki/Camel\\_case](https://en.wikipedia.org/wiki/Camel_case) (02/04/2017);

- **Stile d'indentazione K&R:**  
[https://it.wikipedia.org/wiki/Stile\\_d%27indentazione#Stile\\_K.26R](https://it.wikipedia.org/wiki/Stile_d%27indentazione#Stile_K.26R) (02/04/2017);
- **Glossario:** *Glossario v2.0.0*;
- **Piano di Progetto:** *Piano di progetto v2.0.0*;
- **Piano di Qualifica:** *Piano di qualifica v2.0.0*.

## 2 Processi primari

### 2.1 Fornitura

#### 2.1.1 Scopo

Lo scopo del *Processo di fornitura* è quello di consegnare un prodotto e/o un servizio che soddisfi i requisiti concordati.

#### 2.1.2 Risultati

I risultati ottenuti in seguito ad una corretta attuazione del *Processo di fornitura* sono:

- Stabilire un accordo tra Fornitore e Proponente in merito allo sviluppo, al mantenimento, al funzionamento, alla consegna e all'installazione del prodotto e/o servizio;
- Realizzare un prodotto e/o servizio che soddisfi i requisiti concordati;
- Consegnare il prodotto al Proponente in conformità con i requisiti concordati;
- Installare il prodotto in conformità con i requisiti concordati.

#### 2.1.3 Descrizione

Prendendo come riferimento lo standard [ISO/IEC 12207], il Fornitore deve svolgere le seguenti attività:

- Identificazione opportunità (*Studio di fattibilità*);
- Accordo contrattuale;
- Esecuzione del contratto;
- Consegna e supporto del prodotto e/o servizio;
- Chiusura.

#### 2.1.4 Identificazione opportunità

Successivamente alla pubblicazione dei capitolati d'appalto, il *Responsabile di progetto* ha il compito di convocare il numero di riunioni necessarie al confronto tra i membri del gruppo sui capitolati proposti. Gli *Analisti* hanno così modo di ricavare sufficienti informazioni riguardanti le conoscenze e preferenze di ogni membro del gruppo. Sulla base delle decisioni prese, gli *Analisti* devono redigere uno *Studio di fattibilità* dei capitolati secondo:

- **Dominio tecnologico:** conoscenze sulle tecnologie impiegate nello sviluppo del progetto in questione;
- **Dominio applicativo:** conoscenze sul dominio di applicazione del prodotto;
- **Individuazione di rischi e criticità:** punti critici ed eventuali rischi percorribili durante lo sviluppo.

Nello *Studio di fattibilità v1.0.0* sono racchiuse le motivazioni che hanno spinto il nostro gruppo a candidarsi come Fornitore per il proponente *Zucchetti s.p.a.*.

#### **2.1.5 Accordo contrattuale**

Il Fornitore deve accordarsi con il proponente *Zucchetti s.p.a.* per chiarire, definire e accettare le richieste presenti nel documento di presentazione del capitolato fornito.

#### **2.1.6 Esecuzione del contratto**

Il Fornitore è tenuto a collaborare con il proponente *Zucchetti s.p.a.* per tutta la durata del progetto al fine di raggiungere i seguenti obiettivi:

- Chiarire ogni dubbio riguardante i vincoli sui requisiti;
- Chiarire ogni dubbio riguardante i vincoli di progetto.

Il Fornitore è tenuto a procurare al proponente *Zucchetti s.p.a.* e al committente *Prof. Vardanega Tullio* i seguenti documenti:

- *Piano di progetto*;
- *Analisi dei requisiti*;
- *Piano di qualifica*.

#### **2.1.7 Consegna e supporto del prodotto e/o servizio**

Dopo aver terminato le fasi di sviluppo, verifica e validazione, il Fornitore è tenuto a consegnare al committente *Prof. Vardanega Tullio* il prodotto realizzato in conformità con i requisiti richiesti. Dovrà quindi consegnare un CD-ROM/DVD comprensivo di:

- Utilità di installazione e istruzioni per l'uso;
- Sorgenti completi e utilità di compilazione;
- Documentazione completa e finale;
- Eventuali utilità di collaudo.

Il Fornitore, dopo la consegna del prodotto, non si occuperà della fase di manutenzione del suddetto.

#### **2.1.8 Chiusura**

La chiusura dell'accordo contrattuale tra Fornitore e Proponente è sancita dalla consegna del prodotto realizzato.

## 2.2 Sviluppo

### 2.2.1 Scopo

In questa fase vengono incluse tutte le attività volte a creare il prodotto.

### 2.2.2 Aspettative

I risultati ottenuti in seguito ad una corretta attuazione del *Processo di sviluppo* sono:

- Realizzare un prodotto e/o servizio che soddisfi i requisiti concordati;
- Realizzare un prodotto e/o servizio che soddisfi le attività di validazione e verifica;
- Determinare eventuali vincoli tecnologici e requisiti;
- Determinare gli obiettivi di sviluppo.

### 2.2.3 Descrizione

Prendendo come riferimento lo standard [ISO/IEC 12207], si devono svolgere le seguenti attività:

- Analisi dei requisiti;
- Progettazione;
- Codifica.

### 2.2.4 Analisi dei requisiti

#### Scopo

Determinare tutti i requisiti del progetto. Il risultato di questa attività è un documento in cui vengono elencati i requisiti e i relativi casi d'uso.

#### Aspettative

Produrre il documento *Analisi dei requisiti* in conformità ai requisiti richiesti dal PropONENTE.

#### Descrizione

Vengono analizzati e tracciati tutti i requisiti attraverso l'analisi della specifica del capitolato e la convocazione di riunioni con il proponente *Zucchetti s.p.a.* volte al chiarimento di eventuali dubbi o all'approfondimento di requisiti già noti.

#### Classificazione dei requisiti

Viene stilata una lista dei requisiti individuati nel capitolato e nelle riunioni avvenute con il PropONENTE. La classificazione degli stessi deve avvenire secondo la seguente codifica:

R[Importanza][Tipo][Codice]

dove:

- **Importanza** può assumere i seguenti valori:

- **0** se il requisito è obbligatorio;
- **1** se il requisito è desiderabile;
- **2** se il requisito è opzionale.

- **Tipo** può assumere i seguenti valori:

- **F** se il requisito è funzionale;
- **Q** se il requisito è di qualità;
- **P** se il requisito è prestazionale;
- **V** se il requisito è di vincolo.

- **Codice** identifica univocamente ciascun requisito in modo gerarchico.

Inoltre, per ciascun requisito deve essere indicata:

- **Fonte** dell'individuazione del requisito che può essere:

- **Capitolato**;
- **Riunione**;
- **Caso d'uso**;
- **Interno** (discussioni del gruppo);

- **Descrizione** breve e chiara.

Tutti i requisiti devono essere inseriti nell'apposito sistema di tracciamento “SWEgo” realizzato all'interno del gruppo, che provvederà alla generazione dei codici identificativi univoci.

### Classificazione dei casi d'uso

L'analisi e l'identificazione dei casi d'uso, o use case (UC), deve procedere dal generale al particolare.

Ciascun caso d'uso sarà classificato gerarchicamente con la seguente dicitura:

UC[Codice del padre].[Codice identificativo]

dove:

- **Codice del padre** rappresenta il codice univoco del relativo caso d'uso padre qualora esistesse, altrimenti è omesso;
- **Codice identificativo** rappresenta il codice univoco e progressivo del corrispondente caso d'uso. Il codice può includere diversi livelli di gerarchia che devono essere separati da un punto.

Inoltre, per ciascun caso d'uso deve essere indicato:

- **Nome** del caso d'uso;

- **Attori** coinvolti;
- **Descrizione** chiara e sufficientemente dettagliata;
- **Precondizione**;
- **Scenario principale degli eventi** che descrive il principale flusso di svolgimento e l'eventuale sequenza dei casi d'uso figli;
- **Scenari alternativi** che descrivono la sequenza di eventuali casi d'uso non appartenenti allo scenario principale;
- **Requisiti** ricavati dal caso d'uso;
- Eventuali **Inclusioni**;
- Eventuali **Estensioni**;
- Eventuali **Generalizzazioni**;
- **Postcondizione**.

Infine, tutti i principali casi d'uso devono essere rappresentati da un diagramma UML. Tutti i casi d'uso devono essere inseriti nell'apposito sistema di tracciamento “SWEgo” realizzato all'interno del gruppo, che provvederà alla generazione dei codici identificativi univoci.

## 2.2.5 Progettazione

### Scopo

In questa fase viene descritta una soluzione del problema soddisfacente per tutti gli stakeholder<sub>G</sub>.

### Aspettative

I risultati ottenuti in seguito ad una corretta attuazione di tale fase sono:

- Definire l'architettura logica di sistema che identifica gli elementi del sistema;
- Definire l'architettura logica di sistema in conformità ai requisiti definiti;
- Tracciare, verificare, mantenere coerenti e sincronizzati i requisiti di sistema con l'architettura logica di sistema;
- Garantire la qualità attraverso la correttezza per costruzione.

## Descrizione

Partendo dai requisiti sviluppati nella fase precedente, i *Progettisti* devono sviluppare l'architettura logica del sistema seguendo le sottostanti linee guida:

- Utilizzare componenti con specifiche chiare e coese;
- Realizzare l'architettura rispettando le risorse e i costi prefissati;
- Adottare strutture che si adattino al cambiamento;
- Utilizzare componenti riusabili;
- Suddividere il sistema fino a quando ogni componente ha una complessità trattabile;
- Riconoscere le componenti terminali.

## Metriche

Nello sviluppo del codice i *Progettisti* devono tenere sotto controllo i seguenti indicatori e accertarsi che i loro valori siano conformi a quanto indicato nel *Piano di qualifica v2.0.0*:

- **Grado di accoppiamento**

Viene calcolato in base a due indici:

- Accoppiamento afferente: numero di classi esterne al package<sub>G</sub> che dipendono da sue classi interne. Un numero alto indica che troppe classi dipendono da tale package, quindi eventuali modifiche provocherebbero forti ripercussioni sull'esterno. Se il numero è basso il package risulta poco utile;
- Accoppiamento efferente: numero di classi interne al package dipendenti da classi esterne. Un numero alto può essere sintomo di una scarsa progettazione;

- **Numero di parametri**

Indica il numero di parametri formali di un metodo. Più alto è il numero dei parametri formali, più aumenta la quantità di memoria occupata nella pila dei processi;

- **Numero di campi dati**

Indica il numero di campi dati interni ad una classe. Un numero elevato può rendere difficile la manutenibilità del codice della classe oltre ad essere indice di cattiva programmazione.

È possibile ridurre il numero di campi dati attraverso l'incapsulamento di ulteriori classi.

## Diagrammi UML

Per definire l'architettura logica del sistema nella fase di progettazione, verranno utilizzati diversi tipi di diagrammi UML 2.0:

- Diagrammi delle attività;
- Diagrammi delle classi;
- Diagrammi dei package<sub>G</sub>;
- Diagrammi di sequenza.

## Design pattern<sub>G</sub>

I *Progettisti* devono descrivere i design pattern utilizzati per realizzare l'architettura. È necessario fornire una breve descrizione ed un diagramma semplificato per ognuno di essi.

## Classificazione dei test

Ciascun test sarà identificato univocamente con la seguente dicitura:

T[**Tipo**][Codice identificativo]

In questa codifica:

- **Tipo** può assumere i valori:
  - **I** indica i test di integrazione;
  - **S** indica i test di sistema;
  - **U** indica i test di unità;
  - **V** indica i test di validazione;
- **Codice identificativo** rappresenta il codice univoco e progressivo del corrispondente test.

Tutti i test devono essere inseriti nell'apposito sistema di tracciamento “SWEgo” realizzato all'interno del gruppo, che provvederà alla generazione dei codici identificativi univoci.

## Specifiche tecniche

Per la redazione del documento è necessario effettuare la progettazione architettonica ad alto livello, utilizzando i diagrammi UML sopra elencati. Ogni requisito deve essere tracciato a un componente che lo soddisfa mediante il software utilizzato in modo tale da garantire il soddisfacimento dei requisiti.

Per la progettazione ad alto livello è necessario definire i test di integrazione, sistema e validazione.

## Definizione di prodotto

Per la redazione del documento è necessario effettuare la progettazione architettonica di dettaglio, utilizzando i diagrammi UML sopra elencati. Ogni requisito deve essere tracciato alla/e classe/i che lo soddisfano mediante il software utilizzato in modo tale da garantire che ogni classe soddisfi almeno un requisito.

È necessario definire i test di unità mancanti.

### 2.2.6 Codifica

#### Scopo

Lo scopo della fase di codifica è quello di implementare il prodotto costruendo unità software eseguibili che riflettano la struttura definita in fase di progettazione.

## Aspettative

I risultati ottenuti in seguito ad una corretta attuazione di tale fase sono:

- Realizzare le unità software in conformità ai requisiti;
- Tracciare le unità software e le relativi componenti dell'architettura logica di sistema;
- Definire criteri di verifica delle unità software.

## Descrizione

Partendo dall'architettura logica di sistema definita nella fase precedente, i *Programmatori* devono implementare le unità software definendone anche i criteri di verifica. Nello svolgimento di tali attività devono attenersi ai metodi e agli standard di codifica utilizzati.

## Standard di codifica

Ulteriori norme di codifica potrebbero essere aggiunte nelle fasi successive.

### Intestazione dei file

Per i file contenenti codice si deve:

- Usare le seguente intestazione all'inizio di ogni file:

```
/*
 * File-Name: Nome del file
 * File-Author: Cognome Nome dell'autore
 * File-Date: Data di creazione
 * File-Summary: Breve descrizione del file
 * File-Description: Descrizione dettagliata del file
 **/
```

- Prima di ogni classe scrivere un commento con la seguente struttura:

```
/*
Class-Name: Nome della classe
Class-Summary: Breve descrizione della classe
**/
```

- Per ogni metodo si deve scrivere un commento così strutturato:

```
/*
Method-Name: Nome della classe
Method-Summary: Breve descrizione della classe
Method-Input: breve descrizione dei parametri della funzione nel caso ci siano
Method-Output: breve descrizione dei valori di ritorno nel caso ci siano
**/
```

## Codifica e convenzioni

Tutti i file devono seguire la convenzione UTF-8 per la codifica dei caratteri e LF (U+000A) per andare a capo.

Per indicare il nome di variabili, classi, metodi e funzioni si deve usare la lingua inglese ed la notazione “CamelCase<sub>G</sub>”. È preferibile evitare di usare nomi ambigui che potrebbero generare confusione.

Lo stile di indentazione del codice da seguire è una variante del “K&R<sub>G</sub>” dove lo stile di indentazione delle parentesi del corpo di una funzione è uguale a quello usato per i blocchi. L’indentazione deve essere effettuata utilizzando il tasto di tabulazione.

È preferibile evitare lo sviluppo di metodi e funzioni ricorsive. Qualora fossero comunque usate si deve dimostrarne la corretta terminazione e valutarne l’impatto sulle performance del prodotto in termini di memoria utilizzata.

## Metriche

Nello sviluppo del codice i *Programmatori* devono tenere sotto controllo i seguenti indicatori e accertarsi che i loro valori siano conformi a quanto indicato nel *Piano di qualifica v2.0.0*:

- **Rapporto linee di commento su linee di codice**

Indica il rapporto tra linee di commento e linee di codice in un file (linee vuote escluse). Ritenendo importante la rapidità di comprensione del codice, questa metrica è utile per stimare la manutenibilità;

- **Complessità ciclomatica**

Indica il numero di cammini linearmente indipendenti attraverso il grafo di controllo di flusso del metodo/funzione: i nodi del grafo corrispondono a gruppi indivisibili di istruzioni mentre gli archi connettono due nodi se il secondo gruppo può essere eseguito immediatamente dopo il primo.

È possibile ridurre l’indice di complessità attraverso la suddivisione del metodo/funzione in più parti;

- **Livello di annidamento**

Indica quante strutture di controllo sono inserite l’una all’interno dell’altra. Un alto livello di annidamento può portare ad una complessità maggiore del codice causando difficoltà nella verifica, comprensione e modifica dello stesso;

- **Chiamate innestate di metodi**

Indica quante chiamate innestate di metodi sono inserite l’una all’interno dell’altra. Un alto valore può portare a una saturazione dello stack<sub>G</sub>;

- **Copertura del codice**

Rappresenta la percentuale di codice eseguita durante i test. Maggiore è questo valore, più esaustivi saranno i test e maggiori sono le probabilità di individuare gli eventuali errori;

- **Numero di linee per metodo**

Indica il numero di statement<sub>G</sub> che compongono un metodo. Se un metodo risulta

tropo lungo il suo funzionamento risulterà più complicato da comprendere, quindi può essere opportuno dividerlo in più sotto-funzioni. Un metodo troppo lungo potrebbe addirittura essere sintomo di cattiva progettazione della classe;

- **Validazione W3C<sub>G</sub>**

L'applicativo web deve superare il test di validazione offerto da W3C con 0 errori gravi. Sono accettati avvisi ed inesattezze che non compromettano le funzionalità del sito fino a un massimo di 10 per pagina;

## 3 Processi di supporto

### 3.1 Documentazione

#### 3.1.1 Descrizione

Di seguito sono riportate le regole che si dovranno seguire per la scrittura dei documenti.

#### 3.1.2 Approvazione dei documenti

Ogni documento deve seguire le seguenti fasi:

- Il documento viene redatto dai vari redattori;
- I *Verifieri* procedono al controllo di eventuali errori o imprecisioni, e verificano che i documenti rispettino i vincoli definiti nel *Piano di qualifica v2.0.0*;
- in caso di errori i *Verifieri* procedono a segnalarli ai redattori i quali provvedono a correggere gli errori ripartendo dalla prima fase;
- Se il documento passa la fase di verifica viene consegnato al *Responsabile di progetto* che decide se approvarlo o meno.

Nel caso venga rifiutato il *Responsabile di progetto* deve indicare le criticità ai redattori i quali provvedono a correggere gli errori ripartendo dalla prima fase.

#### 3.1.3 Stesura dei documenti

Qualora fosse necessario stendere un documento contenente sezioni di testo dove lo stile di stesura (frasi, soggetti, verbi, ecc.) andrebbe uniformato ma ci sono più redattori all'opera, è necessario fissarne uno comune in modo tale da ottimizzare il lavoro. Le decisioni sullo stile di stesura saranno prese all'interno del gruppo attraverso gli strumenti di comunicazione adottati oppure riunioni.

#### Indice Gulpease

Il *Piano di qualifica v2.0.0* fornisce delle direttive in merito alla leggibilità richiesta ai documenti prodotti. Ciascun redattore deve quindi tenere sotto controllo l'indice Gulpease di quanto prodotto. Definito nel 1988 all'Università degli Studi di Roma "La Sapienza" per valutare la leggibilità di un documento redatto in lingua italiana, l'indice Gulpease si basa sul calcolo del numero di caratteri contenuto in una parola rapportato con altri fattori quali il numero di parole e di frasi. La formula per il calcolo dell'indice Gulpease è la seguente:

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

Il risultato indica quindi la complessità del documento con un valore compreso tra 0 e 100, dove 100 indica la più alta leggibilità. Attraverso gli studi condotti, risulta che testi con un indice:

- **inferiore a 80** sono difficili da leggere per chi ha la licenza elementare;

- **inferiore a 60** sono difficili da leggere per chi ha licenza media;
- **inferiore a 40** sono difficili da leggere per chi ha un diploma superiore.

Tale indice, però, non indica la comprensibilità del testo. Il documento potrebbe contenere frasi incomprensibili ed avere comunque un alto indice Gulpease. Per la tipologia dei documenti redatti, la formalità nella scrittura e gli argomenti trattati risulta difficile adeguare la stesura del testo ad un indice Gulpease ottimale. Ogni documento sarà quindi controllato anche da un essere umano che avrà il compito di valutare se parti di testo dovranno essere semplificate o meno. Inoltre, i limiti imposti da tale indice saranno sufficientemente rilassati per accettare anche frasi poco più complesse.

### Norme di codifica L<sup>A</sup>T<sub>E</sub>X

Tutti i file di documentazione devono avere la seguente intestazione:

```
% Document-Author: KaleidosCode  
% Document-Date: GG/MM/AAAA  
% Document-Description: Descrizione documento
```

indicando data di creazione e breve descrizione del documento.

Per permettere una migliore lettura del codice L<sup>A</sup>T<sub>E</sub>X, lo stile di indentazione da adottare per la stesura dei file di documentazione prevede di indentare il contenuto con una tabulazione ogni qualvolta si scriva un nuovo elemento di testo: sezione, sottosezione, sotto-sottosezione, paragrafo, ecc.

#### 3.1.4 Struttura dei documenti

##### Prima pagina

La prima pagina di ogni documento riporta i seguenti campi:

- Nome dell'università;
- Nome del gruppo;
- Nome del progetto;
- Descrizione del progetto;
- Titolo documento con versione;
- Logo del gruppo;
- Versione;
- Data redazione;
- Cognome e nome dei redattori del documento;
- Cognome e nome dei *Verificatori* del documento;

- Cognome e nome del *Responsabile di progetto* del documento;
- Uso;
- Lista di distribuzione;
- Versione del documento;
- E-mail del gruppo.

### Diario delle modifiche

Subito dopo la prima pagina è presente una tabella che riassume le modifiche apportate al documento ordinate in modo decrescente rispetto alla data. Per ogni riga viene indicato:

- Versione del documento;
- Data della modifica;
- Cognome Nome di chi ha apportato la modifica;
- Descrizione della modifica.

### Indici

In ogni documento (verbali esclusi) viene usato un indice per le sezioni e, nel caso siano presenti, anche per grafici e figure. L'indice deve essere posizionato dopo il diario delle modifiche.

### Struttura generale delle pagine

Nell'intestazione delle pagine sono presenti:

- Logo del gruppo;
- Nome del documento.

A più pagina si possono trovare:

- Nome del gruppo;
- Nome del progetto;
- Pagina corrente e pagine totali indicate come "Pagina X di Y" dove X indica la pagina attuale e Y le pagine totali.

### Templates

Per rendere omogenea e semplice la stesura dei documenti è stato creato un template in L<sup>A</sup>T<sub>E</sub>X che rispetta le regole stilistiche riportate in questo documento. Questo template è reperibile nel repository <sub>G</sub> GitHub <sub>G</sub> attraverso il path "Documentazione/Templates".

È stato creato anche un template per facilitare la stesura dei verbali che si trova nella medesima directory sopra descritta.

### 3.1.5 Norme tipografiche

In questa sezione vengono descritte le regole di tipografia e ortografia comuni per tutti i documenti.

#### Punteggiatura

- **Parentesi:** il testo racchiuso tra parentesi non deve iniziare o finire con spaziature inoltre alla fine non devono essere presenti caratteri di punteggiatura;
- **Punteggiatura:** i caratteri di punteggiatura non devono mai essere preceduti da spaziatura;
- **Maiuscole:** l'iniziale maiuscola viene usata per il nome del team, del progetto, dei documenti, dei ruoli, delle varie fasi di lavoro e per le parole Fornitore, Proponente e Committente qualora usate pronominalmente.  
Inoltre viene usata negli elenchi puntati e nei casi indicati dalla lingua italiana.

#### Stile di testo

- **Corsivo:** usato nei seguenti casi:
  - **Citazioni:** viene usato il corsivo quando una frase viene citata;
  - **Abbreviazioni:** viene usato per evidenziare abbreviazioni;
  - **Nomi particolari:** i nomi di figure particolari come *Progettista* o *Analista*;
  - **Documenti:** viene usato il corsivo per i nomi dei vari documenti.
- **Grassetto:** usato nei seguenti casi:
  - **Elenchi puntati:** viene usato il grassetto per parole o frasi chiave all'interno di un elenco.  
viene inoltre usato per parole o particolari passaggi importanti;
- **Maiuscolo:** usato soltanto per acronimi o eventuali macro L<sup>A</sup>T<sub>E</sub>X nei documenti;
- **L<sup>A</sup>T<sub>E</sub>X:** per ogni riferimento a L<sup>A</sup>T<sub>E</sub>X bisogna utilizzare il comando \LaTeX.

#### Composizione del testo

- **Elenchi puntati:** prima lettera minuscola (salvo casi precedenti) e devono terminare con punto e virgola, tranne l'ultimo elemento che terminerà con il punto;
- **Note a piè pagina:** prima lettera maiuscola e devono terminare con il punto.

## Formati

- **Numeri:** uso dello standard SI/ISO 31-10;
- **Percorsi:** deve essere usato il comando L<sup>A</sup>T<sub>E</sub>X \url per indirizzi web o mail;
- **Ore:** le ore devono seguire lo standard ISO 8601 quindi espresse come:

HH:MM

dove HH indica l'ora espressa con 2 cifre (00-23) e MM indica i minuti espressi sempre con 2 cifre (00-59);

- **Date:** le date devono seguire lo standard italiano-europeo UNI EN 28601 quindi saranno espresse come:

GG/MM/AAAA

dove GG indica 2 cifre per il giorno, MM indica 2 cifre per il mese e AAAA indica 4 cifre per l'anno.

## Riferimenti vari

Per i vari riferimenti si devono usare i seguenti comandi L<sup>A</sup>T<sub>E</sub>X (che garantiscono la corretta scrittura, con la prima lettera di ogni parola che non sia una preposizione maiuscola):

- **Ruoli:** per i ruoli si devono usare i seguenti comandi:

- \responsabileiprogetto = *Responsabile di progetto*;
- \amministratore = *Amministratore*;
- \analista = *Analista*;
- \progettista = *Progettista*;
- \programmatore = *Programmatore*;
- \verificatore = *Verificatore*;
- \segretario = *Segretario*;
- \amministratori = *Amministratori*;
- \analisti = *Analisti*;
- \progettisti = *Progettisti*;
- \programmatori = *Programmatori*;
- \verificatori = *Verifieri*;
- \segretari = *Segretari*.

- **Documenti:** per i documenti si devono usare i seguenti comandi:

- \pianodiprogetto = *Piano di progetto*;
- \pianodiqualifica = *Piano di qualifica*;
- \normediprogetto = *Norme di progetto*;
- \studiodifattibilita = *Studio di fattibilità*;
- \analisiderequisiti = *Analisi dei requisiti*;
- \specificatecnica = *Specifica tecnica*;
- \definizioneprodotto = *Definizione di prodotto*;
- \manualeutente = *Manuale utente*;
- \glossario = *Glossario*.

- **Documenti con versione:** per i documenti con versione si devono usare i seguenti comandi:

- \pianodiprogettov = *Piano di progetto v2.0.0*;
- \pianodiqualificav = *Piano di qualifica v2.0.0*;
- \normediprogettov = *Norme di progetto v2.0.0*;
- \studiodifattibilitav = *Studio di fattibilità v1.0.0*;
- \analisiderequisitiv = *Analisi dei requisiti v2.0.0*;
- \specificatenicav = *Specifica tecnica v1.0.0*;
- \definizioneprodottov = *Definizione di prodotto v1.0.0*;
- \manualeutentev = *Manuale utente v1.0.0*;
- \glossariov = *Glossario v2.0.0*.

- **Revisione:** per le revisioni si devono usare i seguenti comandi:

- \revisionedeirequisiti = **Revisione dei Requisiti**;
- \revisionediacettazione = **Revisione di Accettazione**;
- \revisionediprogettazione = **Revisione di Progettazione**;
- \revisionediqualifica = **Revisione di Qualifica**.

- **Nome del gruppo:** per il nome del gruppo definito come “*KaleidosCode*” si dovrà usare \kaleidoscode;
- **Nome del Proponente:** per riferirsi al Proponente come “*Zucchetti s.p.a.*” si dovrà usare \proponente;
- **Nome del Committente:** per riferirsi al Committente come “*Prof. Vardanega Tullio*” si dovrà usare \vardanega;
- **Nome del progetto:** per riferirsi al progetto come “*SWEDesigner*” si dovrà usare \progetto.

Inoltre i nomi di file senza percorso completo si devono scrivere usando il formato monospace e per scrivere nomi dei componenti si deve usare il formato "Cognome Nome".

## Sigle

Per rendere più accessibili tabelle e diagrammi si devono usare (dove necessario) le seguenti sigle:

- **AdR**: *Analisi dei requisiti*;
- **GL**: *Glossario*;
- **NdP**: *Norme di progetto*;
- **PdP**: *Piano di progetto*;
- **PdQ**: *Piano di qualifica*;
- **SdF**: *Norme di progetto*;
- **ST**: *Specifiche tecniche*;
- **RA**: **Revisione di Accettazione**;
- **RP**: **Revisione di Progettazione**;
- **RQ**: **Revisione di Qualifica**;
- **RR**: **Revisione dei Requisiti**.

### 3.1.6 Componenti grafiche

#### Tabelle

Ogni tabella deve essere accompagnata da una descrizione che specifichi anche l'indice ad essa associata per renderla tracciabile all'interno del documento.

#### Immagini

Le immagini devono essere convertite in formato png o jpg prima di essere incorporate nei documenti e ciascuna deve essere accompagnata da una descrizione che specifichi anche l'indice ad essa associata.

### 3.1.7 Classificazione documenti

#### Documenti formali

Un documento è ritenuto formale solo dopo essere stato approvato dal *Responsabile di progetto*, dopo l'approvazione viene considerato pronto per la revisione da parte del Committente.

#### Documenti informali

Un documento è ritenuto informale fino all'approvazione del *Responsabile di progetto*, fino a quel momento è considerato ad uso interno.

### 3.1.8 Versionamento

Ogni documento deve essere accompagnato dal numero della versione attuale così formato:

vX.Y.Z

dove:

- **X**: numero di uscite formali del documento e aumenta ad ogni approvazione da parte del *Responsabile di progetto*;
- **Y**: usato per modifiche sostanziali e verifiche;
- **Z**: usato per indicare modifiche minori.

Per riferirsi a una specifica versione del documento si deve usare il seguente formato:

*Nome Documento vX.Y.Z*

mentre il nome da applicare ai file è:

NomeDocumento\_vX.Y.Z.pdf

### Variazione indici

Le variazioni degli indici avvengono nel seguente modo:

- **X**: deve essere aggiornata dal *Responsabile di progetto* dopo la sua approvazione;
- **Y**: deve essere incrementato da chi esegue la modifica o la verifica;
- **Z**: deve essere incrementato da chi esegue la modifica.

Queste modifiche sono visibili dal diario delle modifiche.

### 3.1.9 Verbale

#### Riunione interna (chat<sub>G</sub>/videoconferenze incluse)

Il verbale di riunione interna è un documento informale che permette di tenere traccia degli argomenti discussi in ogni riunione. Il segretario, scelto a rotazione, ha il compito di redigere questo documento. Il verbale prodotto deve poi essere condiviso con tutti i membri del gruppo mediante Google Drive<sub>G</sub>, in modo da rendere sempre disponibile e accessibile il contenuto dello stesso.

Qualora un verbale di riunione interna debba essere fornito in sede di revisione, dovrà essere redatto in L<sup>A</sup>T<sub>E</sub>X utilizzando il template fornito allo scopo.

#### Riunione esterna

In caso di riunione esterna con il Proponente e/o Committente, il verbale è un documento ufficiale che può avere valore normativo e deve quindi essere redatto in L<sup>A</sup>T<sub>E</sub>X utilizzando il template fornito allo scopo, come già scritto precedentemente. Anche questi verbali devono essere condivisi tra i membri del gruppo attraverso Google Drive.

### Struttura verbale

La prima pagina di ogni verbale riporta i seguenti campi:

- Nome dell'università;
- Nome del gruppo;
- Nome del progetto;
- Descrizione del progetto;
- Oggetto;
- Logo del gruppo;
- Data redazione;
- Cognome e nome del redattore del documento;
- Cognome e nome del *Verificatore* del documento;
- Cognome e nome del *Responsabile di progetto* del documento;
- Uso;
- Lista di distribuzione;
- E-mail del gruppo.

La pagina successiva deve contenere le informazioni sulla riunione:

- Data riunione;
- Luogo dell'incontro;
- Ora di inizio;
- Ora di fine;
- Partecipanti.

Infine deve essere presente:

- L'ordine del giorno;
- Il riassunto delle discussioni fatte.

#### 3.1.10 Glossario

Il *Glossario* conterrà alcune parole presenti negli altri documenti, che possono essere termini generanti ambiguità d'interpretazione oppure far parte del contesto di applicazione. I termini sono raggruppati per lettera iniziale e ordinati secondo criterio lessicografico. Ogni termine presente nel *Glossario* deve aver associato una descrizione concisa. I termini andranno inseriti parallelamente alla stesura degli altri documenti.

## 3.2 Verifica

### 3.2.1 Scopo del processo

Scopo della verifica è quello di accertare che documenti e file di codice prodotti non contengano errori e che rispettino le regole riportate in questo documento.

### 3.2.2 Attività

Compito dei *Verificatori* è quello di redigere una lista con gli errori più comuni trovati durante lo svolgimento della loro attività nella documentazione, che verrà integrata progressivamente. Tale lista si trova in appendice a questo documento e serve a migliorare i processi di verifica successivi.

#### Gestione degli errori trovati in documentazione

Qualora i *Verificatori* trovassero degli errori all'interno dei documenti devono identificare il tipo di errore trovato e agire di conseguenza secondo la lista proposta in seguito:

- Errore **piccolo**: errori ortografici, grammaticali, di stesura delle frasi dovranno essere corretti dal *Verificatore* stesso;
- Errore **medio**: errori di organizzazione del documento, di gestione dell'indice possono essere corretti dal *Verificatore*;
- Errore **grave**: errori quali la mancanza di sezioni nel documento o una grande disorganizzazione dell'indice e dei contenuti, devono essere segnalati ai redattori, attraverso lo strumento di comunicazione Slack<sub>G</sub>, i quali devono provvedere alla correzione e sistemazione.

#### Gestione degli errori trovati nel codice

Per tenere traccia degli errori al codice si deve usare il servizio “Issues” messo a disposizione da GitHub.

Qualora i *Verificatori* trovassero degli errori all'interno del codice o attraverso il test dello stesso, devono seguire la procedura sottostante:

- Il *Verificatore* deve aprire una nuova issue con la descrizione del problema;
- Il *Responsabile di progetto* deve verificare questa issue e provvedere ad assegnarla ad un programmatore;
- Una volta risolta e verificata viene segnata come risolta da parte del *Responsabile di progetto*.

#### Analisi Statica

Una tecnica di verifica che si applica al codice sorgente e alla documentazione e non richiede l'esecuzione del prodotto software. Le principali strategie sono due:

- Inspection: questa tecnica si applica quando si conoscono le problematiche oggetto di verifica. Consiste in una ricerca mirata attraverso il documento per l'identificazione dei problemi, nel caso di documentazione, basandosi su una lista di errori precedentemente stilata (come scritto sopra, tale lista si trova in appendice).
- Walkthrough: questa tecnica consiste nello scorrere l'intero documento alla ricerca di errori generici. Viene applicata soprattutto nelle prime fasi di stesura del documento o di codifica a causa dell'impegno necessario per attuarla, che aumenta notevolmente con l'aumentare delle dimensioni del documento da verificare.

Per eseguire una prima verifica automatica dei documenti, viene usato il controllo ortografico LanguageTool v3.6, integrabile nell'editor L<sup>A</sup>T<sub>E</sub>X consigliato, TexStudio.

### **Analisi Dinamica**

Questa tecnica può essere applicata solo al software, nella sua interezza o solo in parte. Consiste nell'usare una serie di test automatici creati dal gruppo che mettano alla prova il software prodotto e restituiscano risultati relativi a eventuali problemi rilevati. I test devono essere ripetibili durante l'intero ciclo di vita e devono coprire un insieme finito di casi, con valori di ingresso, uno stato iniziale ed esito decidibile.

### **Analisi dei rischi**

Il *Responsabile di progetto* deve monitorare i rischi indicati nel *Piano di progetto v2.0.0*. Qualora ne identifichi di nuovi deve:

- Mettere al corrente il team;
- Elaborare una nuova strategia per la gestione dei nuovi rischi;
- Aggiornare il documento *Piano di progetto*.

## **3.3 Validazione**

### **3.3.1 Scopo del processo**

La validazione si occupa di accertare che il prodotto realizzato sia conforme alle attese.

### **3.3.2 Attività**

Di questa fase si occupa il *Responsabile di progetto* il quale deve verificare che i documenti e il codice sorgente rispetti a pieno i requisiti imposti dal Committente e dal Proponente. Una volta che i documenti ottengono l'approvazione dei *Verificatori*, il *Responsabile di progetto* controlla eventuali errori o mancanze. Nel caso se ne identifichino, si devono segnalare le criticità ai redattori che provvederanno a correggerle.

## **3.4 Strumento di versionamento**

Sono stati presi in considerazione diversi software di versionamento (Git<sub>G</sub>, SVN, Mercurial) prima di decidere di usare **Git**. I motivi principali della scelta sono:

- **Flessibilità:** Git è un repository distribuito con la possibilità di *commit* e *revert* locali;
- **Esperienza del team:** Git è già stato usato da alcuni componenti del gruppo *KaleidosCode*.

In particolare, si è deciso di utilizzare il servizio di hosting GitHub, vista la richiesta del Proponente all'interno del capitolato d'appalto. In ogni repository il branch<sub>G</sub> “Master” deve contenere solamente versioni di documentazione e/o codice approvate dal *Responsabile di progetto*. Per questo motivo deve essere creato almeno un branch alternativo nel quale lavorare liberamente. Nel caso di repository contenenti codice, i branch “Master” e di sviluppo devono essere sottoposti ad analisi statica e dinamica; saranno rigettati se non conformi. Qualora più persone debbano lavorare ad uno stesso file, dovranno creare dei nuovi branch ad ogni implementazione di una modifica rilevante. L'operazione di integrazione continua verrà svolta dallo strumento descritto in sezione 3.5.8. Al completamento di un incremento, è possibile effettuare una richiesta di merge<sub>G</sub>. Un *Verificatore* avrà in compito di controllare il merge e di accettarlo oppure di segnalare la presenza di eventuali errori come descritto in sezione 3.2.2.

### 3.4.1 Norme di commit

Al termine di ogni modifica significativa apportata ad uno dei documenti versionabili caricati nel repository GitHub è obbligatorio effettuare un commit. Ogni commit deve essere corredata di descrizione sintetica ma precisa di quanto modificato, in modo da permettere una agevole lettura della cronologia qualora fosse necessario.

Si definiscono le seguenti convenzioni per la scrittura del titolo:

- + [NomeDocumento | NomeSezione | NomeOggetto] indica la creazione del documento *NomeDocumento*, la stesura della sezione *NomeSezione* o in generale la creazione di *NomeOggetto*;
- - [NomeDocumento | NomeSezione | NomeOggetto] indica l'eliminazione del documento *NomeDocumento*, la cancellazione della sezione *NomeSezione* o in generale la rimozione di *NomeOggetto*;
- ~ [NomeDocumento | NomeSezione | NomeOggetto] indica la modifica del documento *NomeDocumento*, della sezione *NomeSezione* o in generale di *NomeOggetto*. In particolare in questo caso è necessario che la descrizione associata sia chiara e sufficiente a spiegare l'effettiva natura della modifica apportata.
- **TASK:** [NomeTask] è il titolo che deve essere utilizzato al completamento dell'attività associata al task “NomeTask” di Asana. Questo verrà automaticamente aggiornato e segnato come completato da Zapier (sezione 3.5.7).

## 3.5 Altri strumenti

### 3.5.1 Google Drive

In questo servizio di condivisione vengono messi solo file che non necessitano di controllo di versione. Esso conterrà:



**Figura 1:** Schermata di presentazione di Google Drive

- Documenti che sono recuperabili da altre fonti (es.  $\text{internet}_G$ );
- File di installazione dei software utilizzati dal gruppo. In questo modo si punta a garantire l'uso della stessa versione software per ogni componente del gruppo;
- I manuali (software, libri, pdf, ...) di consultazione, così da avere uniformità di versione e di informazione;
- I file pdf dei verbali redatti.

La possibilità di installare Drive sul proprio PC dà modo ad ogni componente del gruppo di avere a disposizione documentazione e software anche offline. Google Drive viene utilizzato come strumento di supporto allo sviluppo della documentazione e del software presente su Git.

### 3.5.2 ProjectLibre



**Figura 2:** Logo di ProjectLibre

Per pianificare le attività legate allo sviluppo del progetto e la gestione delle risorse si è scelto di utilizzare **ProjectLibre**. ProjectLibre è un programma open source per il project management. È basato su Java ed è quindi eseguibile su ogni sistema operativo. È il successore ufficiale di OpenProj. Tale software è stato scelto in quanto:

- È portatile, essendo basato su Java;
- È open-source;
- Genera diagrammi Gantt<sub>G</sub>;

- Genera automaticamente diagrammi Program Evaluation and Review Technique (PERT<sub>G</sub>) a partire dal Gantt;
- Genera automaticamente la Work Breakdown Structure<sub>G</sub> (WBS), a partire dal Gantt con allocazione di risorse;
- Calcola i parametri Schedule Variance<sub>G</sub> (SV) e Budget Variance<sub>G</sub> (BV);
- Permette il salvataggio su file XML<sub>G</sub>: essendo un file testuale è possibile fare il merge dei file in caso di conflitti su repository.

### 3.5.3 L<sup>A</sup>T<sub>E</sub>X<sub>G</sub>

Per la stesura dei documenti è stato scelto di utilizzare il linguaggio L<sup>A</sup>T<sub>E</sub>X. La scelta è stata quasi obbligata in quanto L<sup>A</sup>T<sub>E</sub>X permette di separare il contenuto dalla formattazione, permettendo un versionamento più semplice e consentendo di definire template condivisi per uniformare i documenti. Altre soluzioni come Microsoft Office, LibreOffice o Google Docs non avrebbero consentito un livello altrettanto elevato di separazione portando ad un aumento del lavoro da parte dei membri del gruppo e una maggior difficoltà di uniformazione dei contenuti. Per la scrittura di documenti L<sup>A</sup>T<sub>E</sub>X l'editor consigliato è **TeXstudio**.

### 3.5.4 Strumenti per il controllo ortografico

Per eseguire una verifica automatica dei documenti approssimativa, viene usato il programma di controllo ortografico LanguageTool versione 3.6, integrabile nell'editor L<sup>A</sup>T<sub>E</sub>X TexStudio.

### 3.5.5 Strumenti per i diagrammi UML

Per creare i diagrammi UML di package, classi, attività e sequenza relativi alla progettazione è stato scelto di utilizzare Astah, vista la semplice interfaccia e la presenza di una versione gratuitamente scaricabile. I diagrammi dei casi d'uso presenti nel documento *Analisi dei requisiti v2.0.0* sono invece generati automaticamente utilizzando PlantUML.

### 3.5.6 SWEgo

Per effettuare il tracciamento di casi d'uso, requisiti, componenti, classi e test è stato scelto di realizzare un'applicazione web accessibile attraverso browser<sub>G</sub> ed ospitata su un server dedicato: SWEgo. Tale applicativo permette di aggiungere casi d'uso, attori partecipanti, requisiti, componenti, classi e test. Casi d'uso e requisiti sono automaticamente identificati attraverso i codici auto-generati dall'applicazione, lo stesso per componenti, classi e test; È inoltre possibile esportare il codice L<sup>A</sup>T<sub>E</sub>X che elenca casi d'uso, requisiti, componenti, classi, test e le loro relazioni.

### 3.5.7 Zapier

Per permettere un'efficace gestione dei task di Asana, si è scelto di utilizzare Zapier. Si tratta di un servizio web che permette di integrare altre applicazioni. Sarà usato

### SWEgo

Inserisci use case  
 Modifica use case  
 Elimina use case  
 Visualizza use case  
 Inserisci requisito  
 Modifica requisito  
 Elimina requisiti  
 Visualizza requisiti  
 Inserisci attore  
 Visualizza attori  
 Traccia requisiti-use case  
 Visualizza tracciamento  
 Logout

ID	Descrizione	Importanza	Tipo	Fonre	Padre	EliminaModifica
R0F1	Il sistema deve essere in grado di realizzare diagrammi delle classi	Obbligatorio	Funzionale	Capitolato	Nessuno	<a href="#">EliminaModifica</a>
R0F11	Il sistema deve essere in grado di realizzare un diagramma dei package	Obbligatorio	Funzionale	Caso d'uso	Nessuno	<a href="#">EliminaModifica</a>
R0F14	Il sistema deve permettere di gestire un progetto.	Obbligatorio	Funzionale	Caso d'uso	Nessuno	<a href="#">EliminaModifica</a>
R0F14.1	Il sistema deve permettere di creare un nuovo progetto	Obbligatorio	Funzionale	Caso d'uso	R0F14	<a href="#">EliminaModifica</a>
R0F14.1.1	Il sistema deve permettere di definire il nome del nuovo progetto	Obbligatorio	Funzionale	Caso d'uso	R0F14.1	<a href="#">EliminaModifica</a>
R0F14.2	Il sistema deve permettere di caricare un progetto	Obbligatorio	Funzionale	Caso d'uso	R0F14	<a href="#">EliminaModifica</a>
R0F14.3	Il sistema deve permettere di chiudere il progetto attuale	Obbligatorio	Funzionale	Caso d'uso	R0F14	<a href="#">EliminaModifica</a>
R0F14.3.1	Il sistema deve chiedere se al momento della chiusura si desidera salvare le modifiche effettuate successivamente all'ultimo salvataggio.	Obbligatorio	Funzionale	Caso d'uso	R0F14.3	<a href="#">EliminaModifica</a>
R0F14.4	Il sistema deve permettere di annullare l'effetto dell'ultimo comando eseguito	Obbligatorio	Funzionale	Caso d'uso	R0F14	<a href="#">EliminaModifica</a>
R0F14.5	Il sistema deve permettere di ripristinare l'effetto dell'ultimo comando annullato	Obbligatorio	Funzionale	Caso d'uso	R0F14	<a href="#">EliminaModifica</a>
R0F14.6	Il sistema deve permettere di leggere il codice prodotto	Obbligatorio	Funzionale	Caso d'uso	R0F14	<a href="#">EliminaModifica</a>
R0F14.7	Il sistema deve permettere di esportare il codice prodotto	Obbligatorio	Funzionale	Caso d'uso	R0F14	<a href="#">EliminaModifica</a>
R0F14.8	Il sistema deve permettere di salvare il progetto attuale	Obbligatorio	Funzionale	Caso d'uso	R0F14	<a href="#">EliminaModifica</a>
R0F14.8.1	Il sistema deve permettere di salvare il progetto attuale sovrascrivendolo.	Obbligatorio	Funzionale	Caso d'uso	R0F14.8	<a href="#">EliminaModifica</a>
R0F14.8.2	Il sistema deve permettere di salvare con nome il progetto in una directory scelta.	Obbligatorio	Funzionale	Caso d'uso	R0F14.8	<a href="#">EliminaModifica</a>
R0F15	Il sistema deve permettere di editare il diagramma delle classi	Obbligatorio	Funzionale	Caso d'uso	Nessuno	<a href="#">EliminaModifica</a>
R0F15.1	Il sistema deve permettere di aggiungere una nuova classe al diagramma delle classi	Obbligatorio	Funzionale	Caso d'uso	R0F15	<a href="#">EliminaModifica</a>
R0F15.10	Il sistema deve permettere di aggiungere un commento all'interno del diagramma delle classi	Obbligatorio	Funzionale	Caso d'uso	R0F15	<a href="#">EliminaModifica</a>
R0F15.11	Il sistema deve permettere il collegamento di un commento ad un elemento presente nel diagramma delle classi	Obbligatorio	Funzionale	Caso d'uso	R0F15	<a href="#">EliminaModifica</a>
R0F15.12	Il sistema deve permettere la modifica di un commento presente nel diagramma delle classi	Obbligatorio	Funzionale	Caso d'uso	R0F15	<a href="#">EliminaModifica</a>

**Figura 3:** Schermata di tracciamento requisiti in SWEgo

in particolare per collegare Asana e GitHub in modo tale che al commit GitHub con titolo "TASK: [NomeTask]" venga automaticamente aggiornato a "completato" lo stato dell'omonimo task Asana.

#### 3.5.8 Strumenti per l'integrazione continua

Per l'operazione di integrazione continua è stato scelto Jenkins.

Tale strumento permette di impostare la compilazione del codice e può essere utilizzato con i principali strumenti di gestione del codice sorgente tra cui Git (sezione 3.4). La sua esecuzione è azionata ad ogni commit. Jenkins sarà utilizzato principalmente nella fase di codifica.

#### 3.5.9 Strumenti per la validazione W3C<sub>G</sub>

L'applicativo web verrà testato anche attraverso gli strumenti di validazione online:

- <https://validator.w3.org/> (02/04/2017) per le pagine web;
- <https://jigsaw.w3.org/css-validator/> (02/04/2017) per i fogli di stile.

#### 3.5.10 Script

È stato deciso di usare script<sub>G</sub> dedicati per le seguenti operazioni:

- Calcolo dell'indice Gulpease, dove uno script dedicato provvede a fornire i valori calcolati per i vari documenti.

## 4 Processi organizzativi

### 4.1 Scopo

Lo scopo dei processi è quello di produrre il *Piano di progetto*, al fine di pianificare e gestire i ruoli che i membri dovranno assumere nella dinamica della progettazione di *SWEDesigner*.

### 4.2 Aspettative

I risultati ottenuti in seguito ad una corretta attuazione di tali processi sono:

- Definire i ruoli dei membri del gruppo;
- Pianificare e calendarizzare l'esecuzione dei compiti programmati;
- Produrre il *Piano di progetto*.

### 4.3 Gestione organizzativa

#### 4.3.1 Ruoli di progetto

L'assegnazione dei ruoli viene pianificata all'interno del *Piano di progetto*. Le ore di lavoro devono essere distribuite in modo più possibile omogeneo tra i membri del gruppo. Ogni membro deve inoltre ricoprire ciascun ruolo almeno una volta.

#### Responsabile di progetto

Il *Responsabile di progetto* è il portavoce nonché il responsabile delle scelte del gruppo. In particolare si occupa di:

- Approvare l'offerta economica;
- Gestire le risorse;
- Pianificare e coordinare le attività;
- Analizzare e gestire i rischi;
- Approvare i documenti;
- Assicurarsi che vengano rispettate le *Norme di progetto*;
- Assicurarsi che vengano rispettate le pianificazioni definite nel *Piano di progetto*.

#### Amministratore

L'*Amministratore* è colui che si occupa di gestire l'efficienza dell'ambiente di lavoro. In particolare si occupa di:

- Studiare e fornire strumenti che migliorano l'ambiente di lavoro;

- Gestire l'archiviazione, il versionamento e la configurazione dei documenti e del software;
- Eliminare o ridurre per quanto possibile le difficoltà nella gestione di processi e di risorse;
- Automatizzare il lavoro dove possibile.

### **Analista**

L'*Analista* deve identificare e tracciare il dominio del problema. In particolare si occupa di:

- Trasformare le richieste del cliente in specifiche per il prodotto;
- Elaborare le specifiche nell'*Analisi dei requisiti* e nello *Studio di fattibilità*.

### **Progettista**

Il principale ambito di lavoro del *Progettista* è lo stack tecnologico<sub>G</sub>. In particolare si occupa di:

- Indicare le tecnologie più idonee allo sviluppo del progetto;
- Descrivere il funzionamento del sistema e progettarne l'architettura;
- Produrre una soluzione ammissibile in termini di risorse impiegate.

### **Programmatore**

Il *Programmatore* è l'addetto alla codifica. In particolare si occupa di:

- Implementare le soluzioni indicate dal *Progettista*;
- Scrivere codice opportunamente commentato, versionato e mantenibile, in accordo al documento *Norme di progetto*;
- Stilare la documentazione del codice;
- Realizzare e fornire gli strumenti per la verifica e la validazione del prodotto.

### **Verificatore**

Il *Verificatore* è l'addetto a tutte le attività di verifica. In particolare si occupa di:

- Controllare che le regole stabilite dalle *Norme di progetto* siano rispettate durante ogni attività di progetto.

## **4.4 Metriche di processo**

Per poter tenere continuamente sotto controllo il progresso e lo stato del progetto, si è deciso di utilizzare le seguenti metriche.

### Schedule Variance<sub>G</sub>

È una metrica di progetto standard, indica se si è in linea, in anticipo o in ritardo rispetto alla schedulazione pianificata delle attività di progetto. È pari alla differenza tra il valore delle attività realizzate ed il valore delle attività pianificate alla data corrente; più precisamente:

$$\text{Schedule Variance (SV)} = \text{Earned Value (BCWP)} - \text{Planned Value (BCWS)}$$

Dove:

- Earned Value o BCWP (Budgeted Cost of Work Performed) è il valore (denaro) delle attività realizzate alla data corrente;
- Planned Value o BCWS (Budgeted Cost of Work Scheduled) è il costo pianificato (denaro) per realizzare le attività di progetto alla data corrente.

Per avere una migliore visione sull'utilizzo del tempo a disposizione, verrà calcolato anche lo Schedule Performance Indicator ed il To Complete Schedule Performance Indicator<sup>1</sup>. Lo Schedule Performance Indicator,

$$SPI = \frac{BCWP}{BCWS}$$

mostra l'efficienza del team nell'utilizzo del tempo allocato per il progetto; un SPI > 1 indica che il team di lavoro è molto efficiente, viceversa, un SPI < 1 indica che non lo è. Il To Complete Schedule Performance Indicator,

$$TSPI = \frac{(Budget Totale - BCWP)}{(Budget Totale - BCWS)}$$

mostra quanto il team dovrà essere efficiente nel tempo rimasto per la realizzazione del progetto; un TSPI < 1 indica che il team può rilassare l'attività lavorativa nel tempo rimasto; viceversa, un TSPI > 1 indica che il team ha bisogno di lavorare più duramente.

### Budget Variance<sub>G</sub>

È una metrica di progetto standard, indica se si è speso di più o di meno rispetto a quanto preventivato alla data corrente. È pari alla differenza tra costo pianificato e costo effettivamente sostenuto alla data corrente; più precisamente:

$$\text{Budget Variance (BV)} = \text{Planned Value (BCWS)} - \text{Actual Cost (ACWP)}$$

Dove l'Actual Cost o ACWP (Actual Cost of Work Performed) è il costo effettivamente sostenuto (denaro) alla data corrente.

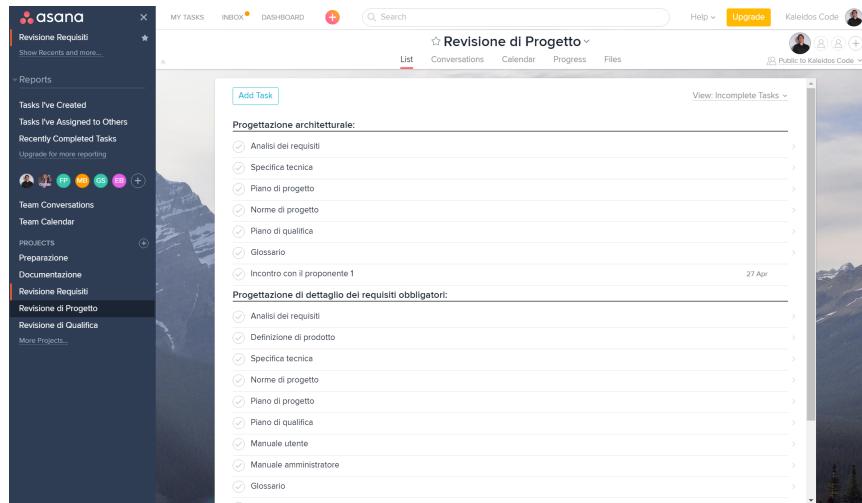
---

<sup>1</sup>I due indici verranno calcolati periodicamente durante lo sviluppo, ma non saranno presenti nei resoconti di verifica.

## 4.5 Strumento di coordinamento

Come piattaforma di gestione del gruppo è stato scelto **Asana**. Asana fornisce:

- Un sistema di gestione dei task;
- Un calendario per organizzare i compiti;
- La visualizzazione del repository associato al progetto;
- Un sistema di rendicontazione del tempo;
- La possibilità di integrare Google Drive, GitHub e web app come Instagantt.



**Figura 4:** Lista di Task in Asana

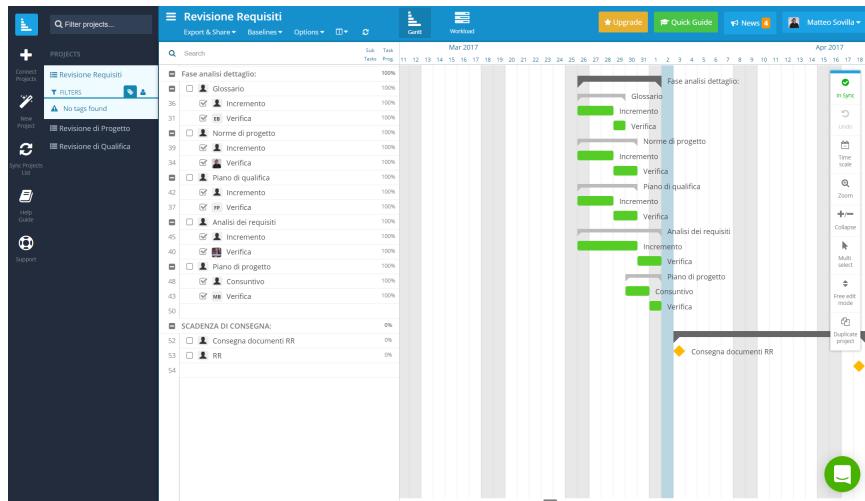
Sono state valutate diverse alternative ma, dopo un'attenta fase di test, nessuna di queste è stata ritenuta all'altezza di Asana per quantità e qualità delle caratteristiche proposte. Sono stati provati i seguenti software:

- **Wrike**: scartato perché la versione gratuita gestisce solo fino a cinque utenti;
- **Trello**: scartato perché carente in funzionalità rispetto alle alternative;
- **Teamwork**: scartato perché pur raggiungendo la completezza di Asana in quanto a caratteristiche non fornisce un'interfaccia altrettanto immediata, aumentando il tempo speso dal team nell'apprendimento dell'uso degli strumenti.

### 4.5.1 Norme di utilizzo

#### Creazione e assegnazione dei task

La gestione delle attività sarà effettuata utilizzando i task di Asana. All'inizio di ogni periodo, il *Responsabile di progetto* si occuperà di creare i task corrispondenti alle attività da svolgere individuate. Ciascun task riporterà una breve descrizione della corrispondente attività e l'indicazione sul periodo temporale di svolgimento ad essa assegnata. Durante l'avanzamento dello sviluppo si divideranno ricorsivamente i task in sotto-attività più semplici, fino a quando ciascuno di essi potrà essere assegnato ad un'unica persona, prevedendone lo svolgimento in un periodo massimo di due giornate.



**Figura 5:** Schermata principale dell'estensione Instagant per Asana

## Avanzamento delle attività

Al termine di ciascuna attività, il componente del gruppo ad essa assegnato è tenuto a cambiare lo stato del corrispondente task per indicarne il completamento. Per tutte le attività associabili a commit su GitHub, questo sarà fatto automaticamente all'atto del rispettivo commit, configurando Zapier come indicato nella sezione 3.5.7. Per le attività non classificabili come sopra sarà il componente preposto a dover aggiornare manualmente lo stato del task.

## Attività di verifica

Con la creazione dei task corrispondenti alle macro-attività di periodo, il *Responsabile di progetto* assegnerà opportunamente ad ognuno di essi uno o più *Verificatori* in base al carico di lavoro atteso e alle disponibilità di ciascuno.

I *Verificatori* sono gli unici componenti del gruppo, oltre al *Responsabile di progetto*, a cui è concesso creare dei task su Asana. Al termine della loro attività di verifica, qualora fossero riscontrati errori, dovranno creare i task corrispondenti alle attività di correzione ed assegnarli a chi di dovere.

I *Verificatori* segnaleranno il completamento del task corrispondente alla loro attività di verifica di un determinato prodotto nel momento in cui lo riterranno valido e conforme alle norme definite.

## Visualizzazione dei Task

Per vedere i task assegnati basta spostarsi nella pagina “MY TASKS” presente tra i link in alto a sinistra.

Per avere una visione d’insieme sui task dell’intero gruppo basta spostarsi nella pagina “Team Calendar” presente nella tendina laterale. Da qui è possibile vedere le scadenze indicative disposte in un calendario.

## 4.6 Comunicazioni

### 4.6.1 Esterne

Per le comunicazioni esterne è stata creata la casella di posta elettronica:

[kaleidos.codec6@gmail.com](mailto:kaleidos.codec6@gmail.com)

Tale indirizzo deve essere l'unico canale di comunicazione tra il gruppo di lavoro e l'esterno. Il *Responsabile di progetto* è l'unico ad accedere all'indirizzo ed è quindi l'unico a poter comunicare con il Proponente ed il Committente del progetto. È compito del *Responsabile di progetto* informare i membri del gruppo delle discussioni avvenute e, qualora fosse necessario, inoltrare loro il messaggio attraverso una mailing list<sub>G</sub>.

### 4.6.2 Interne

Per le comunicazioni interne viene utilizzato il sistema di comunicazione Slack. Tale sistema deve essere utilizzato dai membri del gruppo per comunicare tra loro. In questo modo, ogni componente è costantemente informato sullo scambio di informazioni interne. Qualora fosse necessario l'uso di e-mail, come ad esempio nel caso di un inoltro di messaggio da parte del *Responsabile di progetto*, è stata creata una mailing list:

[kaleidos.code@group.com](mailto:kaleidos.code@group.com)

Per facilitare le comunicazioni tra i membri del gruppo, viene utilizzato anche il sistema di messaggistica e videoconferenza Google Hangout. L'uso di quest'ultimo, nel caso in cui vengano prese decisioni o emergano contenuti utili allo sviluppo del progetto, comporta l'obbligo di redigere un verbale da parte di un membro del gruppo, che renderà disponibile attraverso Google Drive una volta terminata la conversazione. La verbalizzazione ha l'obiettivo di tenere traccia di ogni argomento discusso, in quanto una comunicazione verbale non documentata non è accettabile per il corretto svolgimento del progetto. Nel caso un verbale debba essere fornito in fase di revisione, sarà necessario stenderlo in L<sup>A</sup>T<sub>E</sub>X utilizzando il template fornito allo scopo.

Anche per eventuali decisioni prese su Slack, si richiede che la conversazione venga documentata come sopra descritto.

## 4.7 Composizione e-mail

In questo paragrafo viene descritta la struttura che deve avere un messaggio sia per una comunicazione esterna che per una conversazione interna attraverso mailing list.

### 4.7.1 Destinatario

- **Interno:** l'unico indirizzo utilizzabile è [kaleidos.codec6@gmail.com](mailto:kaleidos.codec6@gmail.com);
- **Esterno:** varia a seconda che ci si debba rivolgere al Proponente, al *Prof. Vardanega Tullio* o al *Prof. Cardin Riccardo*.

#### 4.7.2 Mittente

- **Interno:** l'indirizzo di chi scrive il messaggio;
- **Esterno:** l'unico indirizzo utilizzabile è [kaleidos.codec6@gmail.com](mailto:kaleidos.codec6@gmail.com) e deve essere usato solamente dal *Responsabile di progetto*.

#### 4.7.3 Oggetto

L'oggetto deve essere chiaro ed esaustivo, possibilmente non confondibile con altri preesistenti.

L'oggetto di una comunicazione, una volta avviata, non deve mai essere cambiato.

Nel caso si debba comporre un messaggio di risposta vi è l'obbligo di aggiungere la particella “Re:” all'inizio dell'oggetto per poter distinguere i livelli di risposta; se si dovesse trattare di un inoltro, si deve usare invece la particella “I:”.

#### 4.7.4 Corpo

Il corpo di un messaggio deve contenere tutte le informazioni necessarie alla piena comprensione della comunicazione.

Se alcune parti del messaggio hanno uno o più destinatari specifici, sarà necessario aggiungere il loro nome prima del relativo paragrafo attraverso la segnatura `@destinatario`. In caso di risposta o inoltro del messaggio, il contenuto aggiunto deve essere sempre collocato in testa. Si consiglia di non cancellare il resto del messaggio, per consentire una visione completa della discussione.

#### 4.7.5 Allegati

Qualora fosse necessario, è permesso l'invio di allegati.

### 4.8 Composizione di conversazioni su Slack

In questo paragrafo viene descritto come usare Slack per le conversazioni interne.

È stato creato un canale Slack per ogni documento da redigere dove è possibile effettuare comunicazione di tematiche relative al documento oggetto del canale.

È possibile creare un canale Slack per comunicare solamente con alcuni membri del gruppo qualora ce ne fosse bisogno.

Se fosse necessario inviare un messaggio di struttura complessa (ad esempio, diviso per punti) è possibile inviare un post attraverso la funzionalità presente nella chat.

Qualora si debbano inviare file o altri contenuti, è possibile allegarli al messaggio attraverso l'apposita funzionalità.

Se fosse necessario riferirsi ad uno o più membri specifici di un canale, basterà aggiungere il loro nome prima della relativa comunicazione attraverso la segnatura `@slackUsername`.

### 4.9 Riunioni

#### 4.9.1 Frequenza

Le riunioni interne del gruppo di lavoro avranno una cadenza settimanale.

#### 4.9.2 Convocazione riunione

##### Interna

Il *Responsabile di progetto* ha il compito di convocare le riunioni generali interne, ossia le riunioni a cui sono tenuti a partecipare tutti i membri del gruppo. Se un componente qualsiasi ritiene necessaria la convocazione di una riunione, deve inoltrare la richiesta al responsabile il quale decide se respingerla o accettarla. Il *Responsabile di progetto* deve convocare l'assemblea con almeno quattro giorni di preavviso attraverso l'invio di un post su Slack nel canale dedicato, il cui corpo è formato da:

- Oggetto: Convocazione riunione n. X (dove X indica il numero progressivo di riunioni effettuate);
- Corpo:
  - Data e ora prevista;
  - Luogo previsto;
  - Ordine del giorno.

Ogni componente deve rispondere al messaggio nel minor tempo possibile confermando la presenza o in caso contrario motivando l'eventuale assenza. In caso di mancata risposta il *Responsabile di progetto* è obbligato a contattare direttamente il componente che non ha fornito una risposta. Una volta ricevute tutte le risposte, il *Responsabile di progetto* può decidere se confermare, rinviare o annullare la riunione, basandosi sul numero di presenze e/o assenze, per garantire il maggior numero possibile di presenti. La conferma, il rinvio, l'annullamento devono essere notificati tramite messaggio nel canale di cui sopra.

##### Esterna

Il *Responsabile di progetto* ha il compito di convocare le riunioni generali esterne, ossia le riunioni a cui sono convocati tutti i membri del gruppo e il Proponente e/o Committente. Se un componente qualsiasi ritiene necessaria la convocazione di una riunione, deve inoltrare la richiesta al *Responsabile di progetto* il quale decide se respingere o accettare tale richiesta.

Il *Responsabile di progetto* deve prima di tutto concordare con il Proponente e/o Committente le date e i luoghi possibili in cui svolgere la riunione attraverso l'invio di un'email contenente:

- Oggetto: Richiesta riunione;
- Corpo:
  - Motivazione;
  - Eventuali date e/o luoghi possibili.

Dopodiché, il responsabile deve inviare un post ai membri del gruppo nel canale dedicato su Slack in cui sono specificate:

- Oggetto: Richiesta riunione col Proponente/Committente

- Corpo:
  - Date e/o luoghi possibili.

Ogni membro del gruppo deve rispondere alla comunicazione confermando la presenza o in caso contrario motivando l'eventuale essenza. In caso di mancata risposta, il *Responsabile di progetto* deve contattare direttamente il componente che non ha fornito una risposta. Una volta ricevute tutte le risposte, il responsabile può decidere se confermare, rinviare, annullare la riunione con il Proponente/Committente basandosi sul numero di presenze e/o assenze. Il *Responsabile di progetto* deve poi, in caso di:

- Conferma: comunicare a tutti i membri e al Proponente e/o Committente la data e il luogo definitivi;
- Rinvio: comunicare a tutti i membri la decisione presa e ripetere il procedimento dall'inizio;
- Annullamento: comunicare a tutti i membri la decisione presa e ripetere il procedimento dall'inizio.

#### **4.9.3 Svolgimento riunione**

##### **Esterne**

All'inizio di ogni riunione, verificata la presenza dei membri previsti, viene scelto un segretario che avrà il compito di annotare gli argomenti trattati e di redigere il verbale della riunione, che dovrà poi essere caricato su Google Drive. Tutti i partecipanti devono tenere un comportamento consono al miglior svolgimento dell'assemblea e al raggiungimento degli obiettivi della stessa. Il segretario deve inoltre assicurarsi che venga seguito l'ordine del giorno in modo da affrontare ogni argomento previsto.

## A Lista di controllo

Lista di controllo degli errori comuni rilevati durante le verifiche, da sfruttare per le verifiche successive.

### A.1 Documentazione

- Uso del futuro al posto del presente indicativo;
- Punti e virgola finali in elenchi puntati non sempre presenti;
- Minuscole iniziali negli elenchi non sempre rispettate;
- Parole inserite nei documenti con comando \gl, ma non inserite nel glossario;
- Uso del comando \pedG invece di \gl;
- Alcuni sinonimi non uniformati (e.g. "Qualifica" o "Qualificazione");
- Spesso non viene inserito (o viene inserito erroneamente) un carattere di terminazione di comando L<sup>A</sup>T<sub>E</sub>X;
- Troppe ripetizioni di sostantivi o aggettivi in alcune frasi;
- Troppe virgolette in alcune frasi che spezzano inutilmente il ritmo di lettura;
- Liste nelle quali ogni elemento non termina con un segno di punteggiatura appropriato;
- Identificare i diagrammi UML come "grafici".

### A.2 Diagrammi UML

- Le relazioni inserite tra componenti o classi hanno direzione sbagliata;
- Erroneo uso di interfacce da sostituire, piuttosto, con classi astratte.