TheFellowshipOfTheCode

Norme di Progetto

Informazioni sul documento

Nome Documento	NormeDiProgetto_v3_0_0.pdf
Versione	3
Data di Creazione	8 Dicembre 2015
Data ultima modifica	8 Aprile 2016
Stato	Approvato
Redazione	Franco Berton Alberto Ferrara Simone Magagna Mattia Varotto Matteo Granzotto Marco Prelaz Matteo Gnoato
Verifica	Mattia Varotto
Approvazione	Franco Berton
Uso	Interno
Distribuzione	TheFellowshipOfTheCode
Destinato a	Prof. Tullio Vardanega, Prof. Riccardo Cardin, TheFellowshipOfTheCode
Email di riferimento	thefellowshipofthecode@gmail.com

Sommario

Documento contenente le norme di progetto che il gruppo TheFellowshipOfTheCode seguirà durante tutte le fasi di realizzazione del prodotto QuizziPedia.

Registro delle modifiche

Versione	Descrizione	Autore e Ruolo	Data
3.0.0	Approvazione documento	Franco Berton Responsabile	2016-04-08
2.1.0	Verifica documento	Mattia Varotto Verificatore	2016-03-24
2.0.2	Aggiornata lista degli errori frequenti	Matteo Gnoato Amministratore	2016-03-21
2.0.1	Inserita modalità d'uso del comando nel documento	Matteo Gnoato Amministratore	2016-03-21
2.0.0	Approvazione del documento	Simone Magagna Responsabile	2016-02-21
1.1.0	Verifica del documento	Alberto Ferrara Verificatore	2016-02-21
1.0.4	Associazione degli strumenti selezionati alle attività nelle quali verranno utilizzati	Alberto Ferrara Amministratore	2016-02-19
1.0.3	Spostato studio di fattibilità come figlio del processo di fornitura	Alberto Ferrara Amministratore	2016-02-17
1.0.2	Eliminata sezione tecnologie utilizzate perché superflua	Alberto Ferrara Amministratore	2016-02-16
1.0.1	Modifica processo di sviluppo come appartenente ai processi primari	Alberto Ferrara Amministratore	2016-02-16
1.0.0	Approvazione del documento	Matteo Granzotto Responsabile	2016-01-20
0.2.0	Verifica del documento	Mattia Varotto Verificatore	2015-12-30
0.1.1	Modifica paragrafi Scopo del prodotto e Glossario	Franco Berton Amministratore	2015-12-30
0.1.0	Verifica del documento	Simone Magagna Verificatore	2015-12-11
0.0.4	Stesura sezione Processi organizzativi	Alberto Ferrara Amministratore	2015-12-10
0.0.3	Stesura sezione Processi di supporto	Alberto Ferrara Amministratore	2015-12-10
0.0.2	Stesura sezione Processi di sviluppo	Franco Berton Amministratore	2015-12-09
0.0.2	Stesura sezione Introduzione	Franco Berton Amministratore	2015-12-08
0.0.1	Creato template	Alberto Ferrara Amministratore	2015-12-08

Indice

1 Introduzione **1**
1.1 Scopo del documento . . . 1
1.2 Scopo del prodotto . . . 1
1.3 Glossario . . . 1
1.4 Riferimenti . . . 1
1.4.1 Normativi . . . 1
1.4.2 Informativi . . . 2

2 Processi primari **2**
2.1 Processo di fornitura . . . 2
2.1.1 Studio di Fattibilità . . . 2
2.2 Processo di sviluppo . . . 2
2.2.1 Analisi dei Requisiti . . . 2
2.2.1.1 Classificazione dei requisiti . . . 2
2.2.1.2 Classificazione dei casi d'uso . . . 3
2.2.2 Progettazione . . . 4
2.2.2.1 Descrizione . . . 4
2.2.2.2 Diagrammi . . . 4
2.2.2.3 Requisiti per i progettisti . . . 4
2.2.2.4 Obiettivi della progettazione . . . 4
2.2.3 Codifica . . . 4
2.2.3.1 Descrizione . . . 4
2.2.3.2 Standard di Codifica . . . 5
2.2.3.2.1 Descrizione . . . 5
2.2.3.2.2 Tecniche di Scrittura . . . 5
2.2.3.2.3 Nomi . . . 5
2.2.3.2.4 Commenti . . . 6
2.2.3.2.5 Formattazione . . . 7
2.2.3.3 Ricorsione . . . 7
2.2.4 Strumenti . . . 7
2.2.4.1 DocumentsDB . . . 7
2.2.4.2 Tracciabilità . . . 7
2.2.4.3 TexMaker . . . 8
2.2.4.4 Astah . . . 8
2.2.4.5 WebStorm . . . 8

3 Processi di supporto **8**
3.1 Processo di documentazione . . . 8
3.1.1 Descrizione . . . 8
3.1.2 Strumenti . . . 9
3.1.3 Ciclo di vita di un documento . . . 9
3.1.4 Documenti finali . . . 9
3.1.4.1 Studio di Fattibilità . . . 9
3.1.4.2 Norme di Progetto . . . 9
3.1.4.3 Piano di Progetto . . . 9
3.1.4.4 Piano di Qualifica . . . 10
3.1.4.5 Analisi dei Requisiti . . . 10
3.1.4.6 Specifica Tecnica . . . 10
3.1.4.7 Definizione di Prodotto . . . 10

3.1.4.8 Glossario . . . 10
3.1.4.9 Manuale Utente . . . 10
3.1.4.10 Verbali . . . 10
3.1.5 Struttura del documento . . . 11
3.1.5.1 Prima pagina . . . 11
3.1.5.2 Diario delle modifiche . . . 12
3.1.5.3 Indice . . . 12
3.1.5.4 Formattazione generale delle pagine . . . 12
3.1.6 Norme tipografiche . . . 12
3.1.6.1 Stili di testo . . . 12
3.1.6.2 Punteggiatura . . . 13
3.1.6.3 Composizione del testo . . . 13
3.1.6.4 Formati . . . 13
3.1.7 Composizione email . . . 14
3.1.7.1 Destinatario . . . 14
3.1.7.2 Mittente . . . 14
3.1.7.3 Oggetto . . . 14
3.1.7.4 Corpo . . . 14
3.1.7.5 Allegati . . . 14
3.1.8 Componenti grafiche . . . 14
3.1.8.1 Tabelle . . . 14
3.1.8.2 Immagini . . . 14
3.1.9 Versionamento . . . 14
3.2 Processo di verifica . . . 15
3.2.1 Descrizione . . . 15
3.2.2 Analisi . . . 15
3.2.2.1 Analisi statica . . . 15
3.2.2.2 Analisi dinamica . . . 15
3.2.3 Test . . . 15
3.2.3.1 Test di unità . . . 15
3.2.3.2 Test di integrazione . . . 16
3.2.3.3 Test di sistema . . . 16
3.2.3.4 Test di regressione . . . 16
3.2.3.5 Test di validazione . . . 16
3.3 Strumenti . . . 16
3.3.1 Strumenti per l'analisi statica . . . 16
3.3.2 Strumenti per l'analisi dinamica . . . 17

4 Processi organizzativi **17**
4.1 Processo di coordinamento . . . 17
4.1.1 Comunicazioni . . . 17
4.1.1.1 Responsabilità . . . 17
4.1.1.2 Comunicazioni interne . . . 17
4.1.1.2.1 Descrizione . . . 17
4.1.1.3 Comunicazioni esterne . . . 18
4.1.1.3.1 Descrizione . . . 18
4.1.2 Riunioni . . . 18
4.1.2.1 Finalità . . . 18
4.1.2.2 Riunioni interne . . . 18
4.1.2.2.1 Ruoli . . . 18
4.1.2.2.1.1 Moderatore . . . 18
4.1.2.2.1.2 Segretario . . . 18

4.1.2.2.1.3 Partecipanti 19
4.1.2.2.2 Descrizione 19
4.1.2.3 Riunioni esterne 19
4.1.2.3.1 Descrizione 19
4.2 Processo di pianificazione 20
4.2.1 Descrizione 20
4.2.1.1 Responsabile di Progetto 20
4.2.1.2 Amministratore 20
4.2.1.3 Progettista 20
4.2.1.4 Analista 21
4.2.1.5 Verificatore 21
4.2.1.6 Programmatore 21
4.3 Strumenti 21
4.3.1 Google Drive 21
4.3.2 GitHub 21
4.3.3 Gestione delle attività del progetto 22
4.3.4 Task management 22
4.3.4.1 Creazione di un elenco di task 22
4.3.4.2 Creazione di un task 22
4.3.4.3 Modifica di un task 23
4.3.4.4 Completamento di un task 23
4.4 Lista di controllo 23

1 Introduzione

1.1 Scopo del documento

Questo documento definisce le norme interne che i membri di TheFellowshipOfTheCode dovranno seguire nello svolgimento del capitolato d'appalto QuizziPedia. Ogni membro del gruppo ha l'obbligo di visionare il documento e seguire rigorosamente le norme in esso contenute, al fine di garantire uniformità e coesione nel materiale prodotto e assicurando efficacia ed efficienza al lavoro svolto.
In questo documento verranno definite norme riguardanti:

- La definizione dei ruoli e l'identificazione delle relative mansioni;
- Le modalità di lavoro durante le varie fasi del progetto;
- Le modalità di stesura dei documenti e le convenzioni utilizzate;
- La definizione degli ambienti di sviluppo;
- L'organizzazione della comunicazione e della cooperazione.

1.2 Scopo del prodotto

Lo scopo del prodotto è di permettere la creazione e gestione di questionari in grado di identificare le lacune dei candidati prima, durante e al termine di un corso di formazione.
Il sistema dovrà offrire le seguenti funzionalità:

- Archiviare questionari in un server suddivisi per argomento;
- Somministrare all'utente, tramite un'interfaccia, questionari specifici per argomento scelto;
- Verificare e valutare i questionari scelti dagli utenti in base alle risposte date.

La parte destinata ai creatori di questionari dovrà essere fruibile attraverso un *browser*$_G$ desktop, abilitato all'utilizzo delle tecnologie *HTML5*$_G$, *CSS3*$_G$ e *JavaScript*$_G$. La parte destinata agli esaminandi sarà utilizzabile su qualunque dispositivo: dal personal computer ai tablet e smartphone.

1.3 Glossario

Al fine di evitare ogni ambiguità i termini tecnici del dominio del progetto, gli acronimi e le parole che necessitano di ulteriori spiegazioni saranno nei vari documenti marcate con il pedice $_G$ e quindi presenti nel documento *Glossario*.

1.4 Riferimenti

1.4.1 Normativi

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

1.4.2 Informativi

- *PianoDiProgetto_v_4_0_0*;
- *PianoDiQualifica_v_4_0_0*;
- **Amministrazione di progetto**: `http://www.math.unipd.it/~tullio/IS-1/2014/Dispense/P06.pdf`;
- **Specifiche *UTF-8*$_G$**: `http://www.unicode.org/versions/Unicode6.1.0/ch03.pdf`.

2 Processi primari

2.1 Processo di fornitura

2.1.1 Studio di Fattibilità

Alla pubblicazione dei capitolati d'appalto il *Responsabile di Progetto* dovrà fissare un numero di riunioni volte alla discussione e al confronto tra i membri del *team*$_G$. In seguito, gli *Analisti* dovranno redigere lo *Studio di Fattibilità* in base a quanto emerso nelle riunioni.
Lo *Studio di Fattibilità* sarà articolato nei seguenti punti:

- **Descrizione:** descrizione generale di ciò che viene richiesto dal capitolato;
- **Dominio Applicativo:** descrizione dell'ambito di utilizzo del prodotto richiesto;
- **Dominio Tecnologico:** descrizione delle tecnologie impiegate nello sviluppo del progetto richiesto;
- **Criticità:** elenco delle possibili problematiche che potrebbero sorgere durante lo sviluppo del prodotto richiesto, individuando quindi punti critici ed eventuali rischi;
- **Valutazione Finale:** piccolo riassunto finale nel quale verranno spiegate le motivazioni per cui è stato scelto o scartato il suddetto capitolato.

2.2 Processo di sviluppo

2.2.1 Analisi dei Requisiti

Ultimato lo *Studio di Fattibilità* gli *Analisti* dovranno redigere l'*Analisi dei Requisiti* che dovrà obbligatoriamente essere strutturata nel seguente modo:

2.2.1.1 Classificazione dei requisiti

Dovrà essere redatto un elenco di requisiti, emersi durante le riunioni interne e/o esterne. Questo compito spetta agli *Analisti*. I requisiti dovranno essere classificati secondo la seguente codifica:

R[Tipo][Importanza][Codice]

- **Tipo:** può assumere questi valori:
  - **F:** indica un requisito funzionale;
  - **Q:** indica un requisito di qualità;
  - **P:** indica un requisito prestazionale;
  - **V:** indica un requisito di vincolo.
- **Importanza:** può assumere questi valori:

  - **O:** indica un requisito obbligatorio;
  - **D:** indica un requisito desiderabile;
  - **F:** indica un requisito facoltativo.

- **Codice:** indica il codice identificativo del requisito, è univoco e deve essere indicato in forma gerarchica.

Per ogni requisito si dovrà inoltre indicare:

- **Descrizione:** una breve descrizione, deve essere meno ambigua possibile;
- **Fonte:** la fonte può essere una delle seguenti:
  - *Capitolato:* deriva direttamente dal testo del capitolato;
  - *Verbale:* deriva da un incontro verbalizzato;
  - *Interno:* deriva da discussioni interne al $team_G$;
  - *Casi d'uso:* deriva da uno o più casi d'uso.

2.2.1.2 Classificazione dei casi d'uso

I casi d'uso identificati devono essere descritti nel seguente modo:

UC[Codice padre].[Codice identificativo]

dove:

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

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

- **Nome**: indica il nome del caso d'uso;
- **Attori**: indica gli attori coinvolti nel caso d'uso;
- **Descrizione**: chiara, precisa e concisa descrizione del caso d'uso;
- **Precondizione**: indica la situazione che deve essere vera prima dell'esecuzione del caso d'uso;
- **Postcondizione**: indica la situazione che deve essere vera dopo l'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.

2.2.2 Progettazione

2.2.2.1 Descrizione

L'attività di progettazione definisce come deve essere realizzata la struttura dell'architettura software. I requisiti delineati, all'interno del documento *Analisi dei Requisiti*, devono essere utili a realizzare la documentazione specifica e a determinare le linee guida da seguire durante l'attività di codifica. Tale attività deve essere svolta in maniera ottimale e precisa dai *Progettisti*.

2.2.2.2 Diagrammi

La progettazione deve utilizzare le seguenti tipologie di diagrammi UML_G:

- **Diagrammi di classe**: illustrano una collezione di elementi dichiarativi di un modello come classi e tipi, assieme ai loro contenuti e alle loro relazioni;
- **Diagrammi dei *package*$_G$**: raggruppamenti di classi in una unità di livello più alto;
- **Diagrammi di attività**: illustrano il flusso di operazioni relativo ad un'attività; utilizzati soprattutto per descrivere la logica di un algoritmo;
- **Diagrammi di sequenza**: descrivono una determinata sequenza di azioni dove tutte le scelte sono già state effettuate; in pratica nel diagramma non compaiono scelte, né flussi alternativi.

2.2.2.3 Requisiti per i progettisti

I *Progettisti* sono responsabili delle attività di progettazione. Essi sono tenuti ad avere:

- Profonda conoscenza di tutto ciò che riguarda il processo di sviluppo del software;
- Capacità di saper anticipare i cambiamenti;
- Notevole inventiva per riuscire a trovare una soluzione progettuale accettabile anche in mancanza di una metodologia che sia sufficientemente espressiva;
- Capacità di individuare con rapidità e sicurezza le soluzioni più opportune.

2.2.2.4 Obiettivi della progettazione

La fase di progettazione si pone i seguenti obiettivi:

- Progettare un software con le caratteristiche di qualità che sono state dettagliate nella fase di analisi e specifica dei requisiti;
- Capacità di poter far fronte a modifiche da effettuare senza che l'intera struttura del software già costruita debba essere messa nuovamente in discussione ed elaborata;
- Soddisfare i requisiti di qualità fissati dal committente.

2.2.3 Codifica

2.2.3.1 Descrizione

L'attività di codifica ha come obiettivo quello di passare dalla descrizione della soluzione in termini di architettura alla descrizione della soluzione in formato eseguibile da un calcolatore. I *Programmatori*, responsabili di questa attività, sono tenuti a seguire le linee guida, delineate nell'attività di progettazione, con lo scopo di produrre in output il software designato.

2.2.3.2 Standard di Codifica

2.2.3.2.1 Descrizione

Affinché questa attività sia svolta al meglio, è opportuno che i *Programmatori* rispettino uno standard di codifica. Esso rappresenta uno strumento fondamentale per garantire sintonia tra gli sviluppatori e la creazione di un codice sorgente uniforme e valido. All'interno dello standard di codifica è necessario definire le modalità di scrittura e gestione del codice sorgente, al fine di facilitare la modifica del sistema software. L'utilizzo di tecniche di scrittura del codice affidabili e di regole di programmazione valide per la creazione di codice di alta qualità è di fondamentale importanza per le prestazioni e la qualità del software.

2.2.3.2.2 Tecniche di Scrittura

La definizione di tecniche di scrittura è importante per avere una maggiore comprensione del codice sorgente. Le tecniche di scrittura del codice sono suddivise in tre categorie:

- Nomi;
- Commenti;
- Formattazione.

2.2.3.2.3 Nomi

Lo schema di denominazione rappresenta uno dei supporti più determinanti per la comprensione del flusso logico del software. Di seguito sono riportate le tecniche di denominazione raccomandate:

- Assegnare ad ogni elemento un nome univoco e consono alla funzione svolta;
- Evitare nomi poco chiari e suscettibili di interpretazioni soggettive; questo tipo di nome può contribuire a creare ambiguità piuttosto che astrazione;
- Utilizzare la struttura verbo - nome per la denominazione di routine che consentono di eseguire operazioni specifiche su determinati oggetti;
- Poiché la maggior parte dei nomi viene creata concatenando più parole, utilizzare una combinazione di caratteri maiuscoli e minuscoli per semplificarne la lettura;
- Utilizzare un nome significativo anche per le variabili che vengono visualizzate solo in poche righe di codice;
- Utilizzare nomi di variabili composti da una singola lettera, come i o j, esclusivamente per gli indici a ciclo breve;
- Ridurre l'utilizzo delle abbreviazioni ma utilizzare quelle create in maniera coerente. È opportuno che a ogni abbreviazione corrisponda un solo significato e che a ciascuna parola abbreviata sia associata una sola abbreviazione. Se, ad esempio, si utilizza min come abbreviazione di minimum, in altri contesti non è possibile utilizzare la stessa abbreviazione per minute;
- Nella denominazione di funzioni inserire una descrizione del valore restituito;
- Evitare di utilizzare gli stessi nomi per elementi diversi, ad esempio una routine e una variabile denominate rispettivamente Getname() e iGetname;

- Nella denominazione degli elementi non utilizzare omonimi per evitare ambiguità durante le revisioni del codice;
- Per la denominazione degli elementi evitare l'uso di parole ortograficamente errate;
- È opportuno che nei nomi di file e cartelle sia contenuta una descrizione precisa della relativa funzione.

2.2.3.2.4 Commenti

All'interno del codice sorgente è di fondamentale importanza l'inserimento di commenti al fine di facilitare la comprensione del flusso logico. Di seguito sono suggeriti alcuni metodi di inserimento dei commenti:

- Quando si modifica il codice mantenere sempre aggiornati i relativi commenti;
- All'inizio di ogni routine è utile fornire commenti predefiniti standard in cui siano indicati le limitazioni, i presupposti e lo scopo della routine. Per commento predefinito si intende una breve introduzione in cui siano illustrate le funzionalità della routine;
- Evitare l'aggiunta di commenti alla fine di una riga di codice; la presenza di commenti a fine riga può rendere più difficoltosa la lettura del codice. Tuttavia questo tipo di commento è valido per l'annotazione di dichiarazioni di variabili. In tal caso è necessario allineare tutti i commenti di fine riga a una tabulazione comune;
- Evitare commenti confusi come righe intere di asterischi. Utilizzare invece spazi vuoti per separare i commenti dal codice;
- Evitare di racchiudere blocchi di commenti in cornici grafiche. Si tratta di un espediente interessante ma di difficile gestione;
- Quando si scrivono commenti, utilizzare frasi di senso compiuto. La funzione dei commenti consiste nel chiarire il significato del codice senza aggiungere alcun tipo di ambiguità;
- Inserire commenti in fase di scrittura del codice, in quanto ciò potrebbe non essere possibile in un secondo momento. Inoltre, se dovesse presentarsi l'opportunità di rivedere il codice scritto, tenere presente che ciò che può essere evidente al momento della stesura potrebbe non esserlo più in futuro;
- Evitare commenti superflui o inappropriati, come annotazioni umoristiche;
- Commentare tutto ciò che non è chiaro nel codice;
- Per evitare problemi ricorrenti, è opportuno utilizzare sempre i commenti nel caso di codice relativo a correzioni di errori e a potenziali soluzioni;
- Aggiungere commenti al codice costituito da cicli e diramazioni logiche. Si tratta di aree di fondamentale importanza che facilitano la lettura del codice sorgente;
- Creare commenti adottando uno stile uniforme e una struttura e una punteggiatura coerenti;
- Separare i commenti dai delimitatori di commento con spazi vuoti. In questo modo i commenti saranno chiari e facili da individuare.

2.2.3.2.5 Formattazione

La formattazione facilita la comprensione dell'organizzazione logica del codice. Per consentire agli sviluppatori la decifrazione del codice sorgente è fondamentale una formattazione logica e coerente.
Di seguito sono suggeriti alcuni metodi di formattazione:

- Definire una dimensione standard per i rientri e utilizzarla in maniera coerente. Allineare le sezioni di codice utilizzando il rientro predefinito;
- Allineare in verticale le parentesi di apertura e chiusura in cui esistono coppie di parentesi allineate. È possibile inoltre utilizzare uno stile trasversale, laddove le parentesi di apertura si trovano alla fine della riga mentre quelle di chiusura all'inizio. Qualunque sia lo stile scelto, utilizzarlo in tutto il codice sorgente;
- Rientrare le righe di codice secondo la relativa costruzione logica. Se non viene utilizzato il rientro, il codice risulterà di difficile comprensione;
- Utilizzare spazi prima e dopo la maggior parte degli operatori, se ciò non altera la funzione del codice;
- Utilizzare spazi vuoti per definire la struttura del codice sorgente. In questo modo sarà possibile creare paragrafi di codice, che consentono di semplificare la comprensione della segmentazione logica del software da parte del lettore;
- Suddividere logicamente il codice sorgente tra diversi file fisici;
- Suddividere le sezioni complesse ed estese di codice in moduli comprensibili di dimensioni minori.

2.2.3.3 Ricorsione

La ricorsione va evitata quando possibile. Per ogni funzione ricorsiva sarà necessario fornire una prova di terminazione e sarà necessario valutare il costo in termini di occupazione della memoria. Nel caso l'utilizzo di memoria risulti troppo elevato la ricorsione verrà rimossa.

2.2.4 Strumenti

2.2.4.1 DocumentsDB

Il *team*$_G$ ha creato un applicativo web, *DocumentsDB*, al fine di gestire in maniera veloce, automatizzata e sicura tutti i dati e il loro tracciamento. *DocumentsDB* è stato realizzato sulla base di un software esistente, *PragmaDB*, realizzato da studenti di anni passati e disponibile online. Esso permette la gestione di casi d'uso, attori, fonti, requisiti e termini del *Glossario*. Ogni membro del *team*$_G$ può accedervi via *browser*$_G$ previa autenticazione. L'applicazione si occupa di mantenere ordinata la gerarchia dei requisiti e dei casi d'uso in modo automatico durante tutto il loro ciclo di vita (creazione, modifica, eliminazione). **DocumentsDB** consente di esportare l'elenco degli attori, delle fonti, del glossario, dei casi d'uso, dei requisiti e delle loro relazioni come codice $\LaTeX_G$. Questo rende il lavoro più ordinato e automatizzato. Questo software verrà utilizzato durante l'intero progetto, principalmente nelle attività di analisi e progettazione.

2.2.4.2 Tracciabilità

La Tracciabilità dei requisiti e dei casi d'uso avviene tramite l'utilizzo del software *DocumentsDB*, il quale dà la possibilità ad ogni membro del *team*$_G$ di inserire, modificare o eliminare requisiti o casi d'uso. Inoltre è utilizzato per tenere traccia dello stato dei requisiti, se quindi sono stati soddisfatti o meno.

2.2.4.3 TexMaker

L'editor scelto dal *team*$_G$ per sviluppare i documenti in *LaTeX*$_G$ è *TexMaker*$_G$. *TexMaker*$_G$ è un editor gratuito, moderno e multi-piattaforma per *Linux*$_G$, sistemi *Mac OS*$_G$ e *Microsoft Windows*$_G$ che integra molti strumenti utili per sviluppare documenti in *LaTeX*$_G$. *TexMaker*$_G$ include il supporto *Unicode*$_G$, il controllo ortografico, il completamento automatico, il raggruppamento del codice e un visore incorporato *PDF*$_G$ con il supporto *SyncTex*$_G$ e modalità di visualizzazione continua. *TexMaker*$_G$ è uno strumento facile da usare e da configurare. Questo software verrà utilizzato durante l'attività di analisi e progettazione per redigere i documenti necessari.

2.2.4.4 Astah

Astah$_G$ è l'editor scelto dal *team*$_G$ per la modellazione nel linguaggio *UML*$_G$. *Astah*$_G$ è un editor gratuito, scaricabile online sia per *Microsoft Windows*$_G$, sia per *Linux*$_G$ che per *Mac OS*$_G$. E' un editor abbastanza semplice e user-fiendly. Infatti ad un primo utilizzo, è facile capire come muoversi nell'ambiente di sviluppo, come aggiungere classi, diagrammi e relazioni. Strumento molto utile che l'editor mette a disposizione è quello di poter esportare il diagramma realizzato in un file immagine per una rapida visualizzazione. *Astah*$_G$ supporta la funzionalità per la codifica dei diagrammi delle classi in un linguaggio di programmazione. Dunque, è possibile disegnare un diagramma delle classi, generare il codice e poi sistemare a mano le varie molteplicità delle relazioni. Questo software verrà utilizzato nelle attività di analisi e progettazione per la creazione dei diagrammi *UML*$_G$ necessari.

2.2.4.5 WebStorm

L'ambiente di sviluppo integrato (*IDE*$_G$) utilizzato è *WebStorm*$_G$. Sono stati testati anche altri *IDE*$_G$, ma nessuno si è dimostrato all'altezza. Esso presenta le seguenti funzionalità:

- Autocompletamento del codice *JavaScript*$_G$, *HTML*$_G$ e *CSS*$_G$;
- Autocompletamento per metodi, funzioni e *frameworks*$_G$ esterni, utili per il progetto;
- *Debugger*$_G$ *JavaScript*$_G$;
- Consente di effettuare test di unità per *JavaScript*$_G$ mediante il *framework*$_G$ *Karma*$_G$;
- Compila automaticamente i file *Sass*$_G$ in *CSS*$_G$;
- Tiene traccia dei cambiamenti effettuati sui file, consentendo di visualizzare lo storico delle modifiche locali e ritornare a versioni precedenti in caso di modifiche o perdite accidentali.

Questo software verrà utilizzato durante l'attività di codifica.

3 Processi di supporto

3.1 Processo di documentazione

3.1.1 Descrizione

In questo capitolo verranno trattate tutte le convenzioni adottate dal gruppo *TheFellowshipOfTheCode* riguardo la stesura, verifica e approvazione dei documenti. I documenti sono classificati come:

- *Interni*: utilizzo interno al *team*$_G$;

- *Esterni*: distribuzione esterna al committente e/o proponente.

Questa classificazione è dovuta dal tipo di distribuzione che andrà effettuata di tali documenti. I documenti *approvati* dal *Responsabile di Progetto* devono avere un nome strutturato nel seguente modo:

- La prima lettera del documento deve essere maiuscola;
- Il nome non deve contenere spazi;
- Va indicata la versione del documento nella parte finale del nome, in forma numerica, corrispondente a quanto indicato nel diario delle modifiche.

NomeDelDocumento_v_1_0_0

3.1.2 Strumenti

Per la stesura dell'intera documentazione è stato scelto di utilizzare *LaTeX*$_G$. E' stato scelto questo *linguaggio di markup*$_G$ per avere uno standard comune e per evitare possibili conflitti e incompatibilità derivanti dall'utilizzo di software differente.

3.1.3 Ciclo di vita di un documento

I documenti possono essere in uno dei seguenti stati:

- Documenti *in lavorazione*;
- Documenti *da verificare*;
- Documenti *approvati*.

I documenti *in lavorazione* sono quelli in fase di stesura da parte del relativo redattore. Ultimata la loro realizzazione questi documenti vengono segnati come *da verificare* e passano quindi in mano al relativo *Verificatore*. Infine i documenti *verificati* vengono consegnati al *Responsabile di Progetto* che avrà il compito di approvarli in via definitiva.

3.1.4 Documenti finali

3.1.4.1 Studio di Fattibilità

L'intento del suddetto documento è riportare lo studio effettuato dall'intero *team*$_G$ che ha portato all'accettazione dello sviluppo del progetto scelto. Questo documento è utilizzato internamente dal *team*$_G$ e la lista di distribuzione comprende solo i committenti.

3.1.4.2 Norme di Progetto

L'intento del suddetto documento è riportare tutte le convenzioni, strumenti e norme che il *team*$_G$ dovrà adottare durante tutto lo sviluppo del progetto. Questo documento è utilizzato internamente al *team*$_G$ e la lista di distribuzione comprende solo i committenti.

3.1.4.3 Piano di Progetto

L'intento del suddetto documento è descrivere come il *team*$_G$ gestisce le risorse temporali e umane. Tratta inoltre la gestione dei rischi. Questo documento è utilizzato anche esternamente e la lista di distribuzione comprende committenti e proponenti.

3.1.4.4 Piano di Qualifica

L'intento del suddetto documento è descrivere il modo in cui l'intero $team_G$ punta a soddisfare gli obiettivi di qualità. Questo documento è utilizzato anche esternamente e la lista di distribuzione comprende committenti e proponenti.

3.1.4.5 Analisi dei Requisiti

L'intento di tale documento è dare una visione generale dei requisiti e dei casi d'uso del progetto. Esso conterrà quindi la lista di tutti i casi d'uso, i diagrammi delle attività di interazione tra utente e sistema sviluppato e i servizi offerti dal prodotto finale. Questo documento è utilizzato anche esternamente e la lista di distribuzione comprende committenti e proponenti.

3.1.4.6 Specifica Tecnica

L'intento del suddetto documento è dare una visione generale del prodotto e delle sue richieste. Più nello specifico verrà mostrata una progettazione ad alto livello, basata su diagrammi dei $package_G$ per la descrizione delle componenti, diagrammi di sequenza per descrivere gli scenari d'uso e diagrammi di attività. Verranno inoltre elencati i software che saranno utilizzati per la realizzazione del progetto.

3.1.4.7 Definizione di Prodotto

L'intento di tale documento è fornire una progettazione dettagliata del prodotto. In esso verranno forniti tutti i dettagli implementativi del prodotto, fondamentali in fase di codifica, che comprenderanno diagrammi UML_G delle classi e i relativi metodi.

3.1.4.8 Glossario

L'intento di tale documento è fornire una definizione dettagliata di tutti i termini tecnici e acronimi utilizzati nell'intera documentazione. Questo documento è utilizzato anche esternamente e la lista di distribuzione comprende committenti e proponenti.

3.1.4.9 Manuale Utente

L'intento di tale documento è fornire all'utente una guida dettagliata di tutte le funzionalità offerte dal prodotto realizzato.

3.1.4.10 Verbali

Questo documento ha lo scopo di riassumere in modo formale le discussioni effettuate e le decisioni prese durante le riunioni. I verbali, come le riunioni, sono classificati in: **interni** ed **esterni**. In particolare i verbali esterni, essendo documenti ufficiali, devono essere redatti dal *Responsabile di Progetto*.
Ogni verbale dovrà essere denominato nel seguente modo:

Verbale_ Numero del verbale_ Tipo di verbale_ Data del verbale

dove:

- **Numero del verbale**: numero identificativo univoco del verbale;
- **Tipo di verbale**: identifica se si tratta di un *Verbale Interno* oppure *Verbale Esterno*;
- **Data del verbale:** identifica la data nella quale si è svolta la riunione corrispondente al verbale. Il formato è il seguente :

YYYY-MM-DD

Nella parte introduttiva del verbale devono essere specificate le seguenti informazioni:

- **Data incontro**: data in cui si è svolta la riunione;
- **Ora inizio incontro**: orario di inizio della riunione;
- **Ora termine incontro**: orario di terminazione della riunione;
- **Luogo incontro**: luogo in cui si è svolta la riunione;
- **Durata**: durata della riunione;
- **Oggetto**: argomento della riunione;
- **Segretario**: cognome e nome del membro incaricato a redigere il verbale;
- **Partecipanti**: cognome e nome di tutti i membri partecipanti alla riunione.

3.1.5 Struttura del documento

3.1.5.1 Prima pagina

Ogni documento deve avere nella prima pagina le seguenti informazioni:

- Nome del gruppo;
- Logo del progetto;
- Nome del progetto;
- Nome del documento;
- Versione del documento;
- Data di creazione del documento;
- Data di ultima modifica del documento;
- Stato del documento;
- Nome e cognome del redattore del documento;
- Nome e cognome del verificatore del documento;
- Nome e cognome del responsabile approvatore del documento;
- Uso del documento;
- Lista di distribuzione del documento;
- Destinatari del documento;
- Email di riferimento;
- Un sommario contenente una breve descrizione del documento.

3.1.5.2 Diario delle modifiche

La seconda pagina contiene il diario delle modifiche. In questa tabella vengono inserite tutte le modifiche effettuate dai vari redattori del documento. Ogni riga della tabella deve contenere le seguenti informazioni:

- **Versione**: versione del documento dopo la modifica;
- **Descrizione**: descrizione della modifica apportata;
- **Autore e Ruolo**: autore della modifica e ruolo che esso ricopre;
- **Data**: data della modifica apportata.

3.1.5.3 Indice

In ogni documento, dopo il diario delle modifiche, deve essere presente un indice di tutte le sezioni. In presenza di tabelle e/o immagini queste devono essere indicate con i relativi indici.

3.1.5.4 Formattazione generale delle pagine

La formattazione della pagina, oltre al contenuto, prevede un'intestazione e un piè di pagina. L'intestazione della pagina contiene:

- Nome del progetto;
- Nome del documento;
- Logo del progetto.

Il piè di pagina contiene:

- Il nome del gruppo;
- Email del gruppo;
- Numero della pagina corrente.

3.1.6 Norme tipografiche

Le seguenti norme tipografiche indicano i criteri riguardanti l'ortografia e la tipografia di tutti i documenti.

3.1.6.1 Stili di testo

- **Grassetto**: il grassetto deve essere utilizzato per evidenziare parole particolarmente importanti, negli elenchi puntati o nelle frasi;
- **Corsivo**: il corsivo deve essere utilizzato nelle seguenti situazioni:
  - Ruoli: ogni riferimento a ruoli di progetto va scritto in corsivo;
  - Documenti: ogni riferimento a un documento va scritto in corsivo;
  - Stati del documento: ogni stato del documento va scritto in corsivo;
  - Citazioni: ogni citazione va scritta in corsivo;
  - Glossario: ogni parola presente nel glossario, oltre ad avere un pedice, deve essere scritta in corsivo.
- **Font codice**: il font per il codice deve essere utilizzato (tramite il comando *texttt{}*) per indicare qualsiasi parte riguardi porzioni di codice.

3.1.6.2 Punteggiatura

- **Punteggiatura**: ogni simbolo di punteggiatura non può seguire un carattere di spazio;
- **Lettere maiuscole**: le lettere maiuscole vanno utilizzate dopo il punto, il punto interrogativo, il punto esclamativo e all'inizio di ogni elemento di un elenco puntato.

3.1.6.3 Composizione del testo

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

3.1.6.4 Formati

- **Date**: le date presenti nei documenti devono seguire lo standard *ISO$_G$ 8601:2004$_G$*:

  YYYY-MM-DD

  dove:

  - YYYY: rappresenta l'anno;
  - MM: rappresenta il mese;
  - DD: rappresenta il giorno.

- **Ore**: le ore presenti nei documenti devono seguire lo standard *ISOG$_G$ 8601:2004$_G$* con il sistema a 24 ore:

  hh:mm

  dove:

  - hh: rappresentano le ore;
  - mm: rappresentano i minuti.

- **Nome del documento**: per riferirsi al nome del documento si dovrà utilizzare il comando LaTeX$_G$ `\documento{Nome del documento}` garantendo in questo modo la corretta sintassi;
- **Nome del gruppo**: per riferirsi al nome del gruppo si dovrà utilizzare il comando LaTeX$_G$ `\gruppo` garantendo in questo modo la corretta sintassi;
- **Nome del progetto**: per riferirsi al nome del progetto si dovrà utilizzare il comando LaTeX$_G$ `\progetto` garantendo in questo modo la corretta sintassi;
- **Link sito del gruppo**: per riferirsi al link del sito del gruppo si dovrà utilizzare il comando LaTeX$_G$ `\gruppoLink` garantendo in questo modo la corretta sintassi;
- **Email del gruppo**: per riferirsi all'indirizzo email del gruppo si dovrà utilizzare il comando LaTeX$_G$ `\email` garantendo in questo modo la corretta sintassi;
- **Nome del proponente**: per riferirsi al nome del proponente, ovvero Zucchetti S.P.A., si dovrà utilizzare il comando LaTeX$_G$ `\proponente` garantendo in questo modo la corretta sintassi.

3.1.7 Composizione email

In questo paragrafo verranno descritte le norme da applicare nella composizione delle email.

3.1.7.1 Destinatario

- Interno: l'indirizzo da utilizzare è *thefellowshipofthecode@gmail.com*;
- Esterno: l'indirizzo del destinatario varia a seconda si tratti del Prof. Tullio Vardanega, Prof. Riccardo Cardin o i proponenti del progetto.

3.1.7.2 Mittente

- Interno: l'indirizzo è di colui che scrive e spedisce la email;
- Esterno: l'indirizzo da utilizzare è *thefellowshipofthecode@gmail.com* ed è utilizzabile unicamente dal *Responsabile di Progetto*.

3.1.7.3 Oggetto

L'oggetto della mail deve essere chiaro, preciso e conciso in modo da rendere semplice il riconoscimento di una mail tra le altre.

3.1.7.4 Corpo

Il testo del corpo della mail deve essere chiaro ed esaustivo. All'interno del testo verrà utilizzata la sintassi *@Ruolo* per riferirsi ad uno specifico ruolo del $team_G$, mentre verrà utilizzata la sintassi *@Destinatario* per riferirsi ad una o più persona del $team_G$.

3.1.7.5 Allegati

È permesso, anche se sconsigliato, l'invio di allegati tramite mail. È preferibile infatti condividere file all'interno del gruppo tramite strumenti più consoni, come *Google Drive*$_G$.

3.1.8 Componenti grafiche

3.1.8.1 Tabelle

Tutte le tabelle presenti all'interno del documento devono avere una didascalia ed un indice identificativo univoco per il loro tracciamento all'interno del documento stesso.

3.1.8.2 Immagini

Le immagini inserite nel documento sono nel formato PNG_G. In alternativa, giustificando il motivo, possono essere inserite in formato PDF_G.

3.1.9 Versionamento

Ogni documento prodotto deve essere identificato, oltre che dal nome, dal numero di versione nel seguente modo:

_vX_Y_Z

dove:

- **X**: indica il numero di uscite formali del documento e viene incrementato in seguito all'approvazione finale da parte del *Responsabile di Progetto*. L'incremento dell'indice **X** comporta l'azzeramento degli indici **Y** e **Z**;

- **Y**: indica il numero crescente delle verifiche. L'incremento viene eseguito dal *Verificatore* e comporta l'azzeramento dell'indice **Z**;
- **Z**: indica il numero di modifiche minori apportate al documento prima della sua verifica. Viene aumentato in modo incrementale.

A ogni modifica del documento anche il nome del file fisico deve essere modificato, seguendo il seguente schema:

nomeDocumento_vX_Y_Z.pdf

3.2 Processo di verifica

3.2.1 Descrizione

Il processo di verifica ha il compito di controllare che ogni documento prodotto rispecchi quanto previsto dai requisiti.

3.2.2 Analisi

3.2.2.1 Analisi statica

L'analisi statica è una tecnica che permette di trovare eventuali anomalie nella documentazione o nel software prodotto. Ci sono due modi con la quale essa viene impiegata:

- ***Walkthrough*$_G$**: questa tecnica di analisi statica consiste nella lettura a largo spettro del documento o del codice, al fine di trovare anomalie, senza avere un'idea precisa degli errori da cercare. Questa tecnica risulta molto utile nella fase iniziale dello sviluppo del prodotto data la scarsa esperienza dei membri del *team*$_G$. Questa tecnica è utilizzata dai *Verificatori* che avranno il compito di stilare una lista contenente gli errori rilevati più spesso. Una volta raggiunta la quasi completezza di questa lista, essa dovrà essere allegata a questo documento, in questo modo sarà possibile l'utilizzo della tecnica di *inspection*$_G$;
- ***Inspection*$_G$**: questa tecnica di analisi statica consiste in una lettura molto più dettagliata e più mirata dei documenti o del codice, utilizzando come supporto fondamentale la lista di controllo contenente gli errori più frequenti. Questa tecnica diventa sempre più efficace dato che la lista verrà sempre ampliata con esperienza nella verifica.

3.2.2.2 Analisi dinamica

L'analisi dinamica viene applicata solamente al software prodotto, in quanto essa consiste nell'esecuzione di test su di esso. Grazie a questi test è possibile verificare la correttezza del software.

3.2.3 Test

3.2.3.1 Test di unità

I test di unità verificano che ogni singola componente del software funzioni correttamente. Effettuando questi test di riduce al minimo la presenza di errori di tutte le componenti di base. I test di unità sono identificati dalla seguente sintassi:

TU[Codice Test]

3.2.3.2 Test di integrazione

I test di integrazione verificano che più unità, validate singolarmente, funzionino correttamente una volta assemblate. I test di integrazione sono identificati dalla seguente sintassi:

TI[Codice Test]

3.2.3.3 Test di sistema

I test di sistema vengono eseguiti sul prodotto che si ritiene essere giunto ad una versione definitiva, serve a verificare che tutti i requisiti siano soddisfatti. I test di sistema sono identificati dalla seguente sintassi:

TS[Codice Requisito]

3.2.3.4 Test di regressione

Il test di regressione consiste nel rieseguire tutti i test relativi ad una componente del software prodotto che ha subito delle modifiche. I test di regressione sono identificati dalla seguente sintassi:

TR[Codice Test]

3.2.3.5 Test di validazione

Il test di validazione coincide con il collaudo del software in presenza del *proponente*. In caso di esito positivo si procede col rilascio del software. I test di validazione sono identificati dalla seguente sintassi:

TV[Codice requisito]

3.3 Strumenti

3.3.1 Strumenti per l'analisi statica

- ***JSHint$_G$***: è uno strumento che aiuta a rilevare possibili errori nel codice *JavaScript$_G$*. Verrà installato come modulo per *Node.js$_G$*;
- ***CSSHint$_G$***: è uno strumento che aiuta a rilevare possibili errori nel codice *CSS$_G$*. Verrà installato come modulo per *Node.js$_G$*;
- ***W3C Markup Validator Service$_G$***: validatore che segnala errori di sintassi nel codice *HTML$_G$*. L'indirizzo web di riferimento è il seguente: `validator.w3.org/`validator.w3.org;
- ***Complexity-report$_G$***: è un'applicazione che verrà installata come modulo per *Node.js$_G$* e serve a misurare metriche riguardanti codice *JavaScript$_G$*, in particolare:
  - *Complessità ciclomatica$_G$*: misura la complessità delle classi, dei metodi e delle funzioni del programma;
  - Rapporto linee di commento su linee di codice: calcola il rapporto tra le righe di codice e le righe di commento scritte;
  - Dipendenze: restituisce le dipendenze interne o esterne con altre classi;
  - Indice di manutenibilità: calcola un valore che indica quanto il codice risulta manutenibile.

3.3.2 Strumenti per l'analisi dinamica

- ***Google Chrome DevTools***$_G$: gli strumenti offerti da *Google Chrome*$_G$ agli sviluppatori servono a tenere sotto controllo l'utilizzo di CPU e di memoria da parte degli oggetti e funzioni *JavaScript*$_G$;
- ***Karma***$_G$: è uno strumento per eseguire test di unità e sarà utilizzato per eseguire test sugli script realizzati. Verrà installato come modulo per *Node.js*$_G$.

4 Processi organizzativi

4.1 Processo di coordinamento

4.1.1 Comunicazioni

4.1.1.1 Responsabilità

Al fine di ottenere comunicazioni chiare e coincise il *Responsabile di Progetto* deve gestirle in modo strutturato, utilizzando la forma che meglio si adatta alla situazione. Le comunicazioni possono essere interne o esterne al *team*$_G$.

4.1.1.2 Comunicazioni interne

4.1.1.2.1 Descrizione

In funzione della tipologia di comunicazione interna si deve decidere quale modalità di presentazione delle informazioni deve essere adottata: formale o informale; scritta o orale. In funzione di questi due criteri, le principali forme di comunicazione che vengono adottate sono:

- **Comunicazione formale scritta**: questa modalità di comunicazione viene utilizzata quando si vuole che esse assumano un carattere di ufficialità e per tutte quelle comunicazioni che sono di fondamentale importanza per la corretta gestione del progetto;
- **Comunicazione verbale formale**: le comunicazioni a carattere divulgativo o di presentazione;
- **Comunicazione informale scritta**: quella scambiata tra due o più membri del *team*$_G$ senza carattere di ufficialità ma utile a chiarire o puntualizzare alcuni aspetti del progetto;
- **Comunicazione informale orale**: questo tipo di comunicazione avviene in tutti quei contesti in cui vengono scambiate opinioni e informazioni sulla attività di progetto.

Per ogni tipologia di comunicazione si è deciso di utilizzare i seguenti strumenti:

- *Piano di Progetto*, convocazioni di riunione e verbali di riunione per le comunicazioni formali scritte;
- Presentazioni e discorsi per le comunicazioni verbali formali;
- E-mail, *instant messaging*$_G$ e *web storage*$_G$ per le comunicazioni informali scritte;
- Riunioni e conversazioni per le comunicazioni informali orali.

Il *team*$_G$ ritiene fondamentale la comunicazione interna per:

- Trasmettere informazioni sulle esigenze tecniche ed operative;
- Informare i membri del *team*$_G$ su principi, politiche, strategie, obiettivi che regolano e definiscono il comportamento dell'organizzazione;
- Motivare i membri del *team*$_G$, realmente e concretamente.

4.1.1.3 Comunicazioni esterne

4.1.1.3.1 Descrizione

Il *team*$_G$ ha ritenuto idoneo creare una casella di posta elettronica per le comunicazioni esterne, denominata:

`thefellowshipofthecode@gmail.com`

La gestione delle relazioni con i proponenti e con i committenti è affidata al *Responsabile di Progetto*, il quale è tenuto, quando ritiene necessario, aggiornare i membri del *team*$_G$ riguardo le informazioni ottenute.

4.1.2 Riunioni

4.1.2.1 Finalità

Le riunioni sono vitali per gestire un progetto. Vanno gestite con attenzione per non sprecare tempo, per aumentare la produttività e la motivazione del *team*$_G$ e per risolvere eventuali problemi. Una riunione è produttiva quando genera nuove idee o consolida una iniziativa, senza trasformarsi in un boomerang. Una riunione ben organizzata, oltre a perseguire uno specifico obiettivo, motiva le persone generando fiducia ed armonia tra i membri del gruppo.

4.1.2.2 Riunioni interne

4.1.2.2.1 Ruoli

4.1.2.2.1.1 Moderatore

Il *Responsabile di Progetto* ha il compito di convocare le riunioni interne, ovvero gli incontri generali a cui partecipano tutti i membri. Ad ogni riunione il *Responsabile di Progetto* verificherà il numero dei presenti per decidere se confermare o meno l'incontro. Il *Responsabile di Progetto* ha l'obbligo di rendere le riunioni produttive, a tal fine esso deve fungere da guida nella discussione degli argomenti, essere fonte di ispirazione e avere l'autorità per seguire l'ordine del giorno e per trovare il consenso sulle decisioni da prendere. Il *Responsabile di Progetto*, essendo il coordinatore della riunione, è tenuto ad avere un atteggiamento:

- **Flessibile**: saper essere fermo quando occorre, senza dimostrarsi prepotente e meno severo a seconda delle situazioni;
- **Autorevole**: avere sempre il controllo della situazione e il rispetto dei partecipanti;
- **Assertivo**: mantenere un'adeguata comunicazione e saper ascoltare chi parla. Il maggior contributo deve essere quello di introdurre gli argomenti, fare domande per stimolare la discussione e trarre le conclusioni.

4.1.2.2.1.2 Segretario

Il *Segretario* ha il compito di tenere la minuta dell'incontro, controllare che siano stati discussi tutti i punti previsti dalla riunione e di redigere il verbale. Alla fine di ogni riunione, il *Segretario* deve inviare ai partecipanti copia del verbale dell'incontro, un rendiconto delle decisioni prese e delle responsabilità assunte e le raccomandazioni in previsione del prossimo incontro. Ciò consentirà di mantenere informato anche chi non ha potuto partecipare al meeting.

4.1.2.2.1.3 Partecipanti

Ad ogni incontro i membri del gruppo devono sentirsi responsabilizzati ad arrivare preparati e mantenere un comportamento consono volto al miglioramento degli obiettivi di essa. Ogni membro del gruppo può richiedere al *Responsabile di Progetto* di indire una riunione interna, in surplus a quella settimanale, motivando la richiesta. Spetterà al responsabile del progetto decidere se approvare o meno la richiesta.

4.1.2.2.2 Descrizione

La convocazione delle riunioni avranno cadenza settimanale, il *Responsabile di Progetto* invierà una email ad ogni membro del gruppo informandolo della riunione. Ogni membro del gruppo è tenuto ad inviare una mail di risposta confermando la presenza o motivando l'assenza. Affinché una riunione abbia validità, è necessario che siano presenti almeno la metà più uno dei membri del $team_G$, in caso contrario la riunione sarà annullata e spostata a data da destinarsi.
Le riunioni interne sono ritenute uno strumento fondamentale per:

- Scambiare delle informazioni e relazioni tra i membri;
- Riconoscere i meriti ai migliori membri;
- Rafforzare lo spirito di appartenenza all'organizzazione da parte di tutto il gruppo;
- Stimolare e motivare a fare sempre meglio.

Le riunioni, generalmente, si dividono in quattro tipi:

- Riunioni per informare;
- Riunioni per valutare;
- Riunioni per decidere;
- Riunioni per progettare.

Le decisioni dovranno riguardare:

- La definizione degli obiettivi da raggiungere;
- La designazione dei responsabili;
- La scelta dei mezzi;
- La determinazione dei tempi di esecuzione;
- Il fissare la data di una nuova ed eventuale riunione.

Importante è la gestione del dopo riunione, attraverso la compilazione di un verbale che metta a conoscenza tutti i membri del $team_G$ delle decisioni prese.

4.1.2.3 Riunioni esterne

4.1.2.3.1 Descrizione

Le riunioni con proponente/i o con committente/i sono ritenute di fondamentale importanza per instaurare un rapporto di fiducia tra le parti, condividere gli obiettivi e i bisogni. L'organizzazione delle riunioni esterne sarà affidata al *Responsabile di Progetto*, o in caso straordinario a qualunque membro del gruppo, qualora quest'ultimo abbia ricevuto il consenso da parte del *Responsabile di Progetto*. Al termine di ogni riunione, il *Responsabile di Progetto* o un suo delegato dovrà verbalizzare quanto emerso ed eventuali richieste di cambiamenti riguardanti elementi che possono avere degli impatti negativi sulla realizzazione del progetto.

4.2 Processo di pianificazione

4.2.1 Descrizione

Durante l'intero sviluppo del progetto didattico ogni componente del gruppo dovrà obbligatoriamente cimentarsi in tutti i ruoli elencati di seguito.
Inoltre non potrà mai accadere che un membro del gruppo risulti redattore e verificatore di un medesimo documento. In questo modo si tende ad evitare il conflitto di interessi che potrebbe sorgere se la responsabilità della stesura e della verifica di un documento fosse affidata ad un'unica persona. Un membro può inoltre ricoprire più ruoli contemporaneamente.

4.2.1.1 Responsabile di Progetto

Il *Responsabile di Progetto* è colui che detiene la responsabilità del lavoro svolto dall'intero $team_G$. Rappresenta inoltre colui che mantiene i contatti diretti presso il fornitore e il cliente, ovvero gli enti esterni. Più in dettaglio, ha responsabilità su:

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

Il *Responsabile di Progetto* redige l'$organigramma_G$, si assicura che tutte le attività vengano svolte seguendo rigorosamente le *Norme di Progetto*, si assicura che vengano rispettati i ruoli assegnati nel *Piano di Progetto* e che non si presentino conflitti di interesse tra redattori e verificatori. Ha inoltre l'incarico di creare, assegnare ad ogni membro e gestire i singoli task. Redige il *Piano di Progetto* e collabora alla stesura del *Piano di Qualifica*. Il *Responsabile di Progetto* è l'unica persona in grado di approvare in modo definitivo un documento.

4.2.1.2 Amministratore

L'*Amministratore* è il responsabile di tutto ciò che riguarda l'ambiente di lavoro. Più in dettaglio, egli si occupa di:

- Controllo dell'ambiente di lavoro;
- Gestione del versionamento della documentazione tramite l'uso di database;
- Controllo delle versioni e delle configurazioni del prodotto;
- Risoluzione dei problemi legati alla gestione dei processi.

L'*Amministratore* redige le *Norme di Progetto* e collabora alla stesura del *Piano di Progetto*.

4.2.1.3 Progettista

Il *Progettista* è il responsabile di tutto ciò che riguarda la progettazione. Più in dettaglio, egli si occupa di:

- Produrre una soluzione attuabile, robusta e semplice entro i limiti di tempo stabiliti;
- Effettuare scelte progettuali volte a garantire la manutenibilità e la modularità del prodotto software.

Il *Progettista* redige la *Specifica Tecnica*, la *Definizione di Prodotto* e la parte programmatica del *Piano di Qualifica*.

4.2.1.4 Analista

L'*Analista* si occupa di tutto ciò che riguarda l'analisi del problema da affrontare. Le mansioni principali sono quelle di:

- Studiare a fondo e capire le problematiche del prodotto da realizzare;
- Produrre una specifica di progetto compresibile per il *proponente*, per il *committente* e per il *Progettista*.

L'*Analista* redige lo *Studio di Fattibilità*, l'*Analisi dei Requisiti* e parte del *Piano di Qualifica*.

4.2.1.5 Verificatore

Il *Verificatore* è responsabile di tutto ciò che riguarda l'attività di verifica. Effettua la verifica dei documenti utilizzando gli strumenti e i metodi proposti nel *Piano di Qualifica* e seguendo rigorosamente quanto descritto nelle *Norme di Progetto*. Egli ha il compito di garantire la conformità rispetto le *Norme di Progetto* dei documenti da lui verificati. Si occupa di redigere la sezione del *Piano di Qualifica* che illustra l'esito delle verifiche effettuate sui documenti.

4.2.1.6 Programmatore

Il *Programmatore* si occuperà di implementare le soluzioni del *Progettista*, è quindi responsabile dell'attività di codifica. In dettaglio, i suoi compiti sono:

- implementare le soluzioni descritte dal *Progettista* in maniera rigorosa;
- Scrivere il codice rispettando le convenzioni prese nel presente documento;
- Implementare i test per il codice scritto da utilizzare per l'attività di verifica.

Il *Programmatore* redige il *Manuale Utente* e produce la documentazione del codice.

4.3 Strumenti

4.3.1 Google Drive

Google Drive$_G$ è un servizio di storage online utilizzato dal *team*$_G$ per condividere file e documenti che non necessitano di tracciamento. *Google Drive*$_G$ consente anche il lavoro concorrente su uno stesso file.

4.3.2 GitHub

Gli strumenti software per il controllo versione sono ritenuti molto spesso necessari per la maggior parte dei progetti di sviluppo software. Per questo motivo, il *team*$_G$ ha deciso di creare un *repository*$_G$, ospitato all'interno di un server, per gestire la continua evoluzione dei documenti digitali come il codice sorgente del software, la documentazione testuale e altre informazioni importanti. Il software di versionamento scelto è *Git*$_G$ perchè presenta molti più aspetti positivi rispetto ai *repository*$_G$ centralizzati. *Git*$_G$ permette di lavorare anche in assenza di connettività con il server centrale. I membri del *team*$_G$ possono lavorare sulla propria copia locale del *repository*$_G$ e rendere pubbliche le modifiche caricandole nel *repository*$_G$ centrale. Questo approccio risulta molto semplice e veloce per poter collaborare con i membri del *team*$_G$ simultaneamente. Inoltre online è possibile reperire moltissima documentazione a riguardo per un rapido apprendimento. Il servizio scelto per il mantenimento dei dai è *GitHub*$_G$.
La struttura base del *repository*$_G$:

- RR;

  - Interni;
  - Esterni.

- RQ;
- RP;
- RA;
- Template.

4.3.3 Gestione delle attività del progetto

Per gestire nella maniera più opportuna la divisione del lavoro, si è scelto di utilizzare il sistema di pianificazione delle attività *Zoho*$_G$.

4.3.4 Task management

Vengono creati dal *Responsabile di Progetto* e sono assegnati a due singoli membri del gruppo, il primo in qualità di redattore e il secondo in qualità di *Verificatore*. Per una gestione più chiara di questa divisione delle attività, sono stati creati su *Zoho*$_G$ due insiemi di *task*$_G$ per ogni documento, definiti:

- *Nome del documento*;
- *Nome del documento - da verificare*.

Una volta che il proprietario ritiene il suo *task*$_G$ completato, deve spostarlo nella relativa sezione *da verificare* in modo tale che il verificatore interessato possa controllarlo e segnarlo come completato, se risulta idoneo.

4.3.4.1 Creazione di un elenco di task

Un elenco di *task*$_G$ rappresenta l'insieme dei *task*$_G$ necessari per la realizzazione di un intero documento. Per la creazione di un nuovo insieme di *task*$_G$ bisogna seguire le seguenti istruzioni:

1. Dalla HomePage di *Zoho*$_G$ selezionare il progetto interessato (QuizziPedia);
2. Selezionare la voce **Compiti e Pietre miliari** dal menù laterale;
3. Selezionare la voce **Nuovo elenco di compiti** e compilare l'elenco di *task*$_G$ nel seguente modo:
   - **Elenco dei compiti:** assegnare il nome del documento che si vuole rappresentare;
   - **Pietra miliare collegata:** selezionare a quale pietra miliare si vuole collegare la realizzazione di questo insieme di *task*$_G$. Corrispondono alla versione finale del documento.

4.3.4.2 Creazione di un task

Per la creazione di un nuovo singolo *task*$_G$ bisogna seguire le seguenti istruzioni:

1. Dalla HomePage di *Zoho*$_G$ selezionare il progetto interessato (QuizziPedia);
2. Selezionare la voce **Compiti e pietre miliari** dal menù laterale;
3. Selezionare la voce **Nuovo compito** e compilare il *task*$_G$ nel seguente modo:

- **Nome task:** assegnare un nome identificativo al $task_G$ seguito dalla $milestone_G$ corrispondente;
- **Aggiungi descrizione:** inserire una descrizione breve ma concisa del $task_G$, deve avere la seguente forma:
  - Redattore: nome del redattore;
  - Verificatore: nome del verificatore.
- **Elenco dei compiti:** selezionare uno degli elenchi di $task_G$, creati precedentemente, che hanno come fine ultimo la realizzazione di un documento;
- **Chi è il responsabile?:** inserire l'intestatario del $task_G$ come primo utente, nel secondo invece inserire il verificatore assegnato;
- **Data di conclusione:** Inserire la data ultima per il completamento del $task_G$;
- **Priorità:** inserire, opzionalmente, una priorità al $task_G$.

4.3.4.3 Modifica di un task

Per modificare un $task_G$ seguire le seguenti istruzioni:

- Selezionare il $task_G$ da modificare;
- Dalla pagina proposta selezionare il campo che si vuole modificare;
- Completata la modifica premere il pulsante **Invio**.

4.3.4.4 Completamento di un task

Dopo che il verificatore ha appurato che il $task_G$ soddisfa i requisiti, e quindi è stato redatto secondo le *Norme di Progetto*, può procedere con il suo completamento seguendo le seguenti istruzioni:

- Selezionare il $task_G$ da segnare come completato;
- Nella parte inferiore della pagina proposta selezionare la voce **Contrassegna come completato**.

Dopo questa operazione, il $task_G$ viene spostato automaticamente nella lista dei $task_G$ completati. Da qui, in casi eccezionali, può essere rispostato nei $task_G$ *da verificare*. Questi casi eccezionali possono essere:

- Il *Responsabile di Progetto* non ha approvato il documento e quindi deve essere rivisto;
- Il *Verificatore*, dopo un secondo controllo sui $task_G$ a lui assegnati, può accorgersi di errori e/o incompletezze. Deve quindi essere rivisto.

4.4 Lista di controllo

Durante l'applicazione della tecnica del $Walkthrough_G$ ai documenti sono stati riportati più frequentemente i seguenti errori:

- **Norme stilistiche**:
  - La prima parola di una voce dell'elenco puntato non inizia con una lettera maiuscola;
  - La voce dell'elenco puntato termina con un punto anziché con un punto e virgola o viceversa;
  - I due punti in grassetto;

  - Errori di battitura;
  - Comando nuova riga non deve essere utilizzato per le descrizioni di descrizione ed utilizzo;
  - Le immagini e il testo devono stare dentro il range della pagina;
  - Dopo il nome di un metodo e di un parametro non ci vanno i :;
  - Parametri di un metodo non devono avere l'accessibilità;
  - Gli array vanno indicati così: Array<Tipo>;
  - Tutti i metodi hanno tipo di ritorno, se qualcosa non ritorna niente è void;
  - I costruttori non hanno tipo di ritorno.

- **Italiano**:
  - Maiuscole usate impropriamente.
- **LaTeX**:
  - Mancato utilizzo dei comandi LaTeX personalizzati;
  - Scambiato il comando `\textit{...}` con `\textbf{...}`;
- **Casi d'Uso**:
  - Mancato rispetto del template stabilito per i punti trattati nei casi d'uso.
- **Glossario**:
  - Sono stati evidenziati dei termini che non andavano nel *Glossario*;
  - Non sono stati evidenziati dei termini che sono presenti nel *Glossario*.