Zeus Code - Progetto "P2PCS"

Norme di Progetto

Versione	1.0.0
Approvazione	Marco Dalla Bà
Redazione	Andrea Pigatto
	Riccardo Basso
	Riccardo Dario
Verifica	Irina Hornoiu
	Diba Meysami
Stato	Approvato
Uso	Interno
Destinato a	Zeus Code
	Prof. Tullio Vardanega
	Prof. Riccardo Cardin

Descrizione
Studio di Fattibilità dei capitolati proposti.

zeuscode17@gmail.com

Diario delle modifiche

Versione	Data	Nominativo	Ruolo	Descrizione
1.0.0	2018-11-30	Sara Feltrin	*Responsabile del progetto*	Approvazione del documento per RR.
0.5.0	2018-11-27	Samuele Giuliano Piazzetta	*Verificatore*	Quinta revisione generale.
0.4.1	2018-11-25	Giacomo Greggio	*Amministratore*	Aggiornamento § 3.
0.4.0	2018-11-24	Mattia Bolzonella	*Verificatore*	Quarta revisione generale.
0.3.2	2018-11-24	Federico Bicciato	*Amministratore*	Aggiornato § 4.
0.3.1	2018-11-24	Federico Bicciato	*Amministratore*	Aggiunta § 3.5.
0.3.0	2018-11-24	Mattia Bolzonella	*Verificatore*	Terza revisione generale.
0.2.5	2018-11-24	Giacomo Greggio	*Amministratore*	Aggiustamento delle tabelle e indici.
0.2.4	2018-11-24	Federico Bicciato	*Amministratore*	Aggiornamento § 3.3.
0.2.3	2018-11-23	Sara Feltrin	*Amministratore*	Aggiornamento § 2.2.
0.2.2	2018-11-23	Paolo Pozzan	*Amministratore*	Completata stesura § 3.3.
0.2.1	2018-11-23	Giacomo Greggio	*Amministratore*	Iniziata stesura § 3.3.
0.2.0	2018-11-22	Mattia Bolzonella	*Verificatore*	Seconda revisione generale.
0.1.4	2018-11-22	Giacomo Greggio	*Amministratore*	Modifiche a § 4.
0.1.3	2018-11-21	Giacomo Greggio	*Amministratore*	Stesura § 3.
0.1.2	2018-11-21	Giacomo Greggio	*Amministratore*	Sistemate sottosezioni § 2.

Versione	Data	Nominativo	Ruolo	Descrizione
0.1.1	2018-11-21	Federico Bicciato	*Amministratore*	Iniziata stesura § 3.
0.1.0	2018-11-20	Mattia Bolzonella	*Verificatore*	Prima revisione generale.
0.0.5	2018-11-20	Giacomo Greggio	*Amministratore*	Aggiornata struttura documento con integrazione di alcuni paragrafi.
0.0.4	2019-11-20	Paolo Pozzan	*Amministratore*	Iniziata stesura § 4.
0.0.3	2018-11-19	Giacomo Greggio	*Amministratore*	Continuazione § 2.
0.0.2	2018-11-19	Paolo Pozzan	*Amministratore*	Stesura § 1 e iniziato stesura § 2.
0.0.1	2018-11-17	Mattia Bolzonella	*Amministratore*	Creata struttura documento LaTeX.

Indice

Elenco delle figure

Elenco delle tabelle

1 Introduzione

1.1 Scopo del documento

Questo documento ha lo scopo di definire le regole di base che tutti i membri di Zeus Code devono rispettare nello svolgimento del progetto, così da garantire uniformità in tutto il materiale. Verrà utilizzato un approccio incrementale, volto a normare passo passo ogni decisione descussa e concordata tra tutti i membri del gruppo. Ciascun componente è obbligato a prendere visione di tale documento e a rispettare le norme in esso descritte allo scopo di perseguire la coesione all'interno del team.

1.2 Scopo del prodotto

Il capitolato C5 ha per obiettivo l'arricchimento delle funzionalità dell'app GaiaGo già esistente, inserendo un nuovo servizio di Car Sharing Peer to Peer per la piattaforma Android. Il servizio, dunque, intende offrire la possibilità di condividere la propria macchina con altre persone amiche o meno sfrutttando almeno 5 core drive del framework Octalysis (G) allo scopo di motivare l'utente a condividere la propria auto con altri.

1.3 Glosssario

Al fine di evitare ambiguità e facilitare la comprensione dei documenti formali viene incluso il Glossario v1.0.0, in cui saranno presenti acronimi, abbreviazioni e termini tecnici. Ogni termine presente nel glossario sarà marcato con una G a pedice.

1.4 Riferimenti

1.4.1 Riferimenti normativi

- **Standard ISO/IEC 12207:1995:**https://www.math.unipd.it/ tullio/IS-1/2009/Approfondimenti/ISO_1220
  1995.pdf;
- **Capitolato d'appalto C5 - Piattaforma peer-to-peer car sharing:** https://www.math.unipd.it/ tullio/IS-
  1/2018/Progetto/C6.pdf

1.4.2 Riferimenti informativi

- **Piano di Progetto:**
- **Piano di Qualifica:**
- **Software Engineering - Ian Sommerville - 10th Edition:**
  (formato cartaceo);
- **Octalysis - Gamification framework**
  https://yukaichou.com/gamification-examples/octalysis-complete-gamification-framework/

2 Processi primari

2.1 Fornitura

2.1.1 Scopo

Lo scopo del processo di fornitura è di determinare le procedure e le risorse necessarie allo svolgimento del progetto. Una volta comprese le richieste del proponente e aver stilato uno *Studio di Fattibilità*, il processo può essere avviato con fine di soddisfare ognuna di queste richieste. Inoltre si deve stipulare e concordare con il proponente un contratto per la consegna del prodotto. Si passa dunque a determinare le procedure, le risorse necessarie, e si sviluppa un *Piano di Progetto* che getterà le basi da perseguire fino alla consegna del materiale prodotto. Il processo di fornitura è composto dalle seguenti fasi:

- avvio;
- approntamento di risposte alle richieste;
- contrattazione;
- pianificazione;
- esecuzione e controllo;
- revisione e valutazione;
- consegna e completamento.

2.1.2 Aspettative

Il gruppo intende mantenere un costante dialogo con il proponente, per avere un riscontro efficace sul lavoro svolto e instaurare un rapporto di collaborazione per quanto riguarda:

- determinare aspetti chiave per far fronte ai bisogni del proponente;
- stilare requisiti e vincoli sui processi;
- stimare le tempistiche di lavoro;
- promuovere una verifica continua;
- chiarire eventuali dubbi emersi;
- accordarsi sulla qualifica del prodotto.

2.1.3 Gestione della Qualità

Il fornitore si impegna ad utilizzare strategie di verifica e validazione con l'obiettivo di garantire efficacia e qualità nei processi e nei prodotti. I progettisti hanno a carico la scelta di queste strategie e le descrivono nel Piano di qualifica v1.0.0. I verificatori invece si occupano di documentare l'esito delle verifiche e prove effettuate seguendo quanto deciso nel Piano di qualifica v1.0.0.

2.1.4 Descrizione

Questa sezione tratta le norme che i membri del gruppo *Zeus Code* devono rispettare in tutte le fasi di progettazione, sviluppo e consegna del prodotto *P2PCS*, al fine di diventare fornitori nei confronti del proponente *GaiaGo* e dei committenti Prof. Tullio Vardanega e Prof. Riccardo Cardin.

2.1.5 Attività

2.1.5.1 Studio di Fattibilità

É compito del responsabile di progetto organizzare riunioni tra i membri del gruppo al fine di permettere lo scambio di opinioni sui capitolati proposti. Lo *Studio di Fattibilità*, redatto dagli analisti, indica per ogni capitolato$_G$:

- **Informazioni generali**: vengono elencate le informazioni di base, come il nome del progetto, il proponente e il committente;
- **Descrizione e finalità del progetto**: viene fatta una presentazione del capitolato$_G$ in generale, una descrizione delle caratteristiche principali richieste per il prodotto e viene definito l'obiettivo che si vuole raggiungere;
- **Tecnologie interessate**: viene fatto un elenco delle tecnologie richieste per lo svolgimento, che rientrano nel dominio tecnologico;
- **Aspetti positivi, criticità e fattori di rischio**: vengono esposte le considerazione fatte dal gruppo sugli aspetti positivi e sui fattori di rischio del capitolato$_G$;
- **Conclusioni**: vengono esposte le ragioni per la quale il gruppo ha deciso di accettare o scartare il capitolato$_G$.

2.1.5.2 Piano di Progetto

Il responsabile, con l'aiuto degli amministratori, redige un *Piano di Progetto* da seguire durante il corso del progetto. Questo documento contiene:

- **Analisi dei rischi**: vengono analizzati nel dettaglio i rischi che potranno presentarsi e vengono esposte le misure e le modalità attraverso le quali i rischi vengono contenuti o mitigati. Viene anche fornita la probabilità con la quale questi possono presentarsi e il livello di gravità per ciascuno;
- **Modello di sviluppo$_G$**: viene descritto il modello di sviluppo$_G$ che è stato scelto, indispensabile per la pianificazione;
- **Pianificazione**: vengono pianificate le attività da eseguire nelle diverse fasi del progetto e vengono stabilite le loro scadenze temporali;
- **Preventivo e consuntivo**: viene data una stima di lavoro necessaria per ciascuna fase proponendo così un preventivo per il costo totale del progetto. Viene anche tracciato, un consuntivo di periodo relativo all'andamento rispetto a ciò che è stato preventivato.

2.1.5.3 Piano di Qualifica

I verificatori dovranno redigere un documento, detto *Piano di Qualifica* contenente le strategie da adottare per garantire la qualità del materiale prodotto dal gruppo, e dei processi attuati. Il piano è così suddiviso:

- **Qualità di processo**: vengono identificati dei processi dagli standard, stabiliti degli obiettivi, escogitate delle strategie per attuarli e individuate le metriche per misurarli e controllarli;
- **Qualità di prodotto**: vengono identificati gli attributi più rilevanti per il prodotto, definiti degli obiettivi per raggiungerli e delle metriche per misurarli;
- **Specifiche dei test**: definiscono una serie di test attraverso i quali il prodotto passa per garantire che soddisfi i requisiti;
- **Standard di qualità**: vengono esposti gli standard di qualità scelti;
- **Valutazioni per il miglioramento:** vengono riportati i problemi e le relative soluzioni nel ricoprire un determinato ruolo e nell'uso degli strumenti scelti;
- **Resoconto delle attività di verifica:** per ogni attività si riportano i risultati delle metriche calcolate in forma di resoconto.

2.1.6 Strumenti

Di seguito sono elencati gli strumenti utilizzati durante il processo di fornitura.

2.1.6.1 Microsoft Excel

Software della suite Microsoft Office per realizzare fogli elettronici. Usato per fare calcoli, produrre diagrammi, istogrammi e areogrammi, creare tabelle e grafici.

2.1.6.2 Microsoft Project

Per assistere i responsabili di progetto nella pianificazione, nell'assegnazione delle risorse, nella verifica del rispetto dei tempi, nella gestione dei budget e nell'analisi dei carichi di lavoro attraverso la creazioni di diagrammi di Gantt$_G$, è stato utilizzato Microsoft Project.
https://products.office.com/it-it/project/

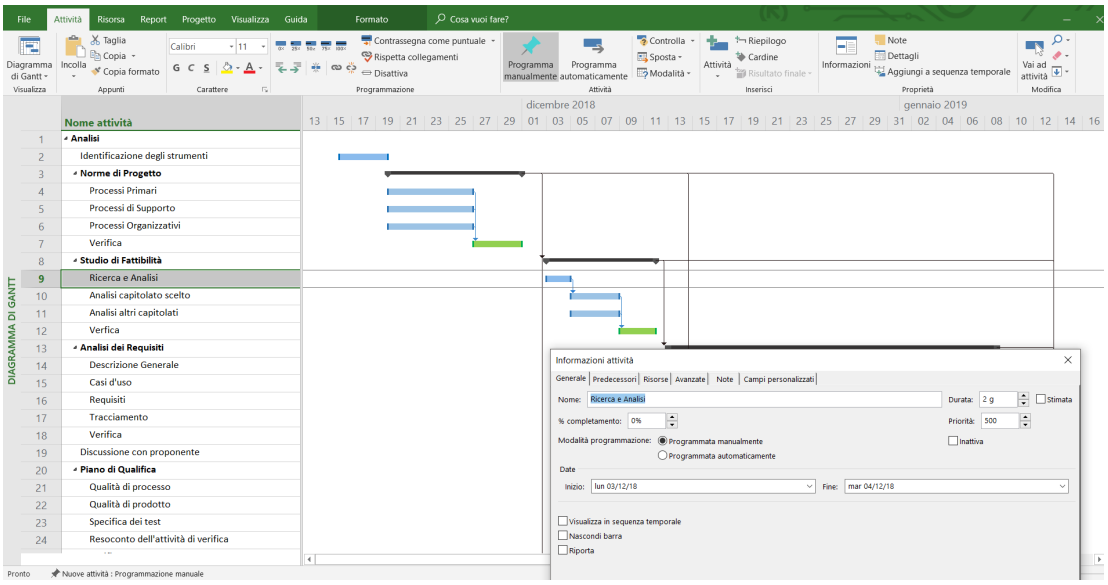

Figura 2.1.1: Microsoft Project: Gantt

2.2 Sviluppo

2.2.1 Scopo

Il processo contiene le attività e i compiti da svolgere, al fine di realizzare il prodotto finale richiesto dal proponente.

2.2.2 Aspettative

Le aspettative sono le seguenti:

- fissare gli obiettivi di sviluppo;
- fissare i vincoli tecnologici;
- fissare i vincoli di design;
- realizzare un prodotto finale che supera i test, che soddisfa i requisiti e le richieste del proponente.

2.2.3 Descrizione

Il processo di sviluppo si articola in:

- *Analisi dei Requisiti*;
- Progettazione;
- Codifica.

2.2.4 Attività

2.2.4.1 Analisi dei Requisiti

Scopo

Gli analisti hanno il compito di redigere il documento di *Analisi dei Requisiti* che individua ed elenca dunque, i requisiti. Lo scopo dei requisiti è quello di:

- definire lo scopo del lavoro;
- fornire ai progettisti riferimenti precisi ed affidabili;
- fissare le funzionalità e i requisiti concordati col cliente;
- fornire una base per raffinamenti successivi al fine di garantire un miglioramento continuo del prodotto e del processo di sviluppo;
- fornire ai verificatori riferimenti per l'attività di controllo dei test;
- calcolare la mole di lavoro per tracciare dei riferimenti per un stima dei costi.

Aspettative

Obiettivo dell'attività è la creazione della documentazione formale contenente tutti i requisiti richiesti dal proponente.

Descrizione

I requisiti si raccolgono secondo modalità predefinite:

- lettura del capitolato$_G$, analisi e approfondimento dello stesso;
- confronto con il proponente;
- confronto tra membri del team di progetto;
- analisi di uno o più casi d'uso.

Casi d'uso

Rappresenta un diagramma che esprime un comportamento, offerto o desiderato, sulla base di risultati osservabili. La struttura dei casi d'uso è così suddivisa:

- codice identificativo;
- titolo;
- diagramma UML$_G$;
- attori primari;
- attori secondari;
- descrizione;

- scenario principale;
- scenario alternativo (se presente);
- inclusioni(se presenti);
- estensioni(se presenti);
- specializzazioni(se presenti);
- precondizione;
- postcondizione.

Codice identificativo dei casi d'uso

Il codice di ogni caso d'uso seguirà questo formalismo:

UC[codice_padre].[codice_figlio]

Dove:

- **codice_padre**: numero che identifica univocamente i casi d'uso;
- **codice_figlio**: numero progressivo che identifica i sottocasi. Può a sua volta includere altri livelli.

Requisiti

Ogni requisito è composto dalla seguente struttura:

- **codice identificativo**: ogni codice identificativo è univoco e conforme alla seguente codifica:

  R[Importanza][Tipologia][Codice]

  Il significato delle cui voci è:

  - **Importanza**: ogni requisito può assumere uno dei seguenti valori:
    - *1*: requisito obbligatorio, ovvero irrinunciabile per gli stakeholder;
    - *2*: requisito desiderabile, ovvero non strettamente necessari ma a valore aggiunto riconoscibile;
    - *3*: requisito opzionale, ovvero relativamente utile oppure contrattabile più avanti nel progetto;
  - **Tipologia**: ogni requisito può assumere uno dei seguenti valori:
    - *F*: funzionale;
    - *Q*: prestazionale;
    - *P*: qualitativo;
    - *V*: vincolo.
  - **Codice**: è un identificatore univoco del requisito in forma gerarchica padre/figlio.

- **classificazione**: viene riportata l'importanza del requisito. Sebbene questa sia un'informazione ridondante ne facilita la lettura;
- **descrizione**: descrizione breve ma completa del requisito, meno ambigua possibile;
- **fonti**: ogni requisito può derivare da una o più tra le seguenti opzioni:
  - *capitolato*$_G$: si tratta di un requisito individuato dalla lettura del capitolato$_G$;
  - *interno*: si tratta di un requisito che gli analisti hanno ritenuto opportuno aggiungere;
  - *caso d'uso*: il requisito è estrapolato da uno o più casi d'uso. In questo caso deve essere riportato il codice univoco del caso d'uso;
  - *verbale*: si tratta di un requisito individuato in seguito ad una richiesta di chiarimento con il proponente. Tali informazioni sono riportate nei verbali in cui ogni requisito individuato è segnato da un codice presente nella tabella dei tracciamenti.

UML

I diagrammi UML$_G$ devono essere realizzati usando la versione del linguaggio v2.0.

2.2.4.2 Progettazione

Scopo

L'attività di progettazione definisce, in funzione dei requisiti specificati nel documento *Analisi dei Requisiti*, le caratteristiche del prodotto software richiesto. Il compito di questa fase è di definire una soluzione del problema che sia soddisfacente per tutti gli stakeholder. La progettazione segue il procedimento inverso rispetto all'*Analisi dei Requisiti* che divide il problema in parti per capirne completamente il dominio applicativo. La progettazione, infatti, rimette insieme le parti specificando le funzionalità dei sottosistemi in modo da ricondurre ad un'unica possibile soluzione.

Aspettative

Il processo ha come risultato la realizzazione dell'architettura del sistema.

Descrizione

Le parti principali sono due:

- **Technology baseline**: contiene le specifiche della progettazione ad alto livello del prodotto e delle sue componenti, l'elenco dei diagrammi UML$_G$ che saranno utilizzati per la realizzazione dell'architettura e i test di verifica;
- **Product baseline**: dettaglia ulteriormente l'attività di progettazione, integrando ciò che è riportato nella Technology baseline. Inoltre definisce i test necessari alla verifica.

Technology baseline

Redatta dal progettista, dovrà includere:

- **Diagrammi UML$_G$**:
  - diagrammi delle classi;
  - diagrammi dei package;
  - diagrammi di attività;
  - diagrammi di sequenza.
- **Tecnologie utilizzate**: devono essere descritte le tecnologie adottate specificandone l'utilizzo nel progetto, i vantaggi e gli svantaggi;
- **Design pattern$_G$**: devono essere descritti i design pattern$_G$ utilizzati per realizzare l'architettura. Ogni design pattern$_G$ deve essere accompagnato da una descrizione ed un diagramma, che ne esponga il significato e la struttura;
- **Tracciamento delle componenti**: ogni requisito deve riferirsi al componente che lo soddisfa;
- **Test di integrazione**: l'unione delle parti, intese come classi di verifica, permette di verificare che ogni componente del sistema funzioni nella maniera voluta.

Product baseline

A carico del progettista c'è anche la Product baseline che si sofferma su diversi aspetti tra i quali:

- **definizione delle classi**: ogni classe deve essere descritta in modo da spiegarne in maniera esaustiva lo scopo e le funzionalità, evitando ridondanze;
- **tracciamento delle classi**: ogni requisito deve essere tracciato in modo da garantire che per ognuno esista una classe che lo soddisfi. Questa operazione è fondamentale per permettere di risalire alle classi ad esso associate;
- **test di unità**: devono essere definiti al fine di verificare che le parti funzionino individualmente nel modo stabilito.

2.2.4.3 Codifica

Scopo

Questa attività ha come scopo quello di normare l'effettiva realizzazione del prodotto software richiesto. In questa fase si concretizza la soluzione attraverso la programmazione. I programmatori dovranno attenersi a queste norme durante la fase di programmazione ed implementazione.

Aspettative

Obiettivo dell'attività è la creazione di un prodotto software conforme alle richieste prefissate con il proponente. L'uso di norme e convenzioni in questa fase, è fondamentale per permettere la generazione di codice leggibile ed uniforme, agevolare le fasi di manutenzione, verifica e validazione e migliorare la qualità di prodotto.

Descrizione

La scrittura del codice dovrà rispettare quanto stabilito nella documentazione di prodotto. Dovrà perseguire gli obiettivi di qualità definiti all'interno del documento *Piano di Qualifica v1.0.0* per poter garantire una buona qualità del codice.

Stile di codifica

Al fine di garantire uniformità nel codice del progetto, ciascun membro del gruppo è tenuto a rispettare le seguenti norme:

- **Indentazione**: i blocchi innestati devono essere correttamente indentati, usando per ciascun livello di indentazione quattro (4) spazi (fanno eccezione i commenti). Al fine di assicurare il rispetto di questa regola si consiglia di configurare adeguatamente il proprio editor o IDE;
- **Parentesizzazione**: è richiesto di inserire le parentesi di delimitazione dei costrutti in linea e non al di sotto di essi;
- **Scrittura dei metodi**: è desiderabile, ove possibile, mantenere i metodi brevi(poche righe di codice);
- **Univocità dei nomi**: classi, metodi, variabili devono avere un nome univoco ed esplicativo al fine di evitare ambiguità e incomprensione;
- **Classi**: i nomi delle classi devono iniziare sempre con una lettera maiuscola;
- **Costanti**: i nomi delle costanti devono essere scritte usando solo maiuscole;
- **Metodi**: i nomi dei metodi devono iniziare con una lettera minuscola. Nel caso siano composti da più parole, quelle successive devono iniziare con una lettera maiuscola (CamelCase$_G$);
- **Lingua**: il codice, come anche i commenti, deve essere scritto in lingua inglese.

Una parte del front end$_G$ è normata dalla "Airbnb JavaScript style guide", il cui uso è implementato attraverso ESLint$_G$.

Ricorsione

L'uso della ricorsione va evitato quanto più possibile in quanto potrebbe indurre ad una maggiore occupazione di memoria rispetto a soluzioni iterative.

2.2.5 Strumenti

Di seguito sono elencati gli strumenti utilizzati dal gruppo durante il progetto per il processo di sviluppo.

2.2.5.1 PragmaDB

Programma usato per il tracciamento dei requisiti, fondamentale dunque per la stesura del documento *Analisi dei Requisiti v1.0.0*.

https://github.com/StefanoMunari/PragmaDB

2.2.5.2 Draw.io

Per la produzione di diagrammi UML$_G$ viene utilizzato Draw.io in quanto offre molte agevolazioni per la produzione veloce dei diagrammi e risulta semplice da usare.

https://www.draw.io/

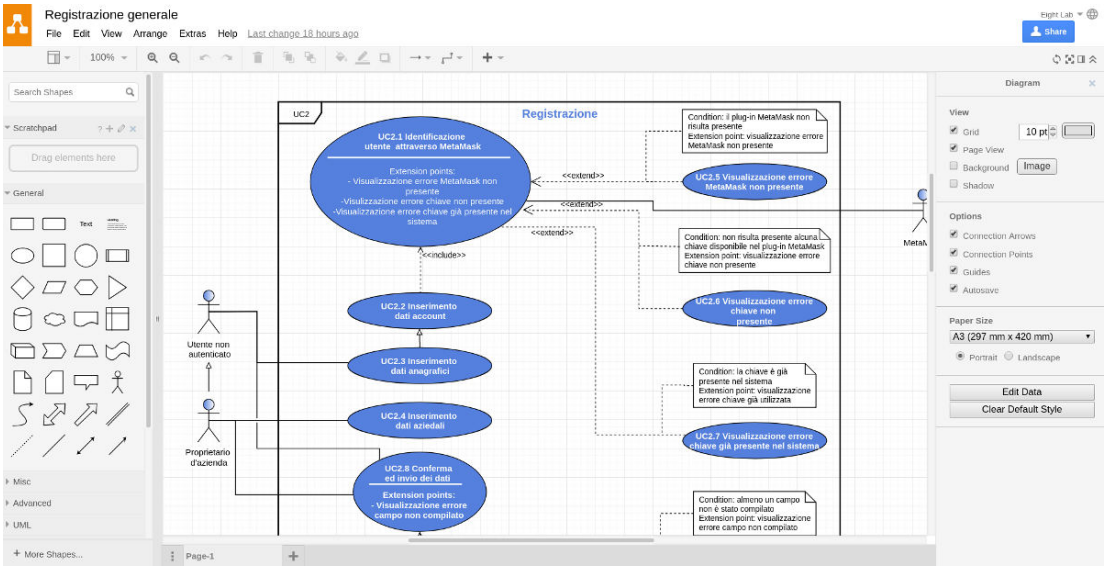

Figura 2.2.1: Software per la creazione di diagrammi online

https://www.jetbrains.com/idea/

2.2.5.3 Android Studio

Android Studio è un ambiente di sviluppo integrato per lo sviluppo per la piattaforma Android. I linguaggio utilizzato in questo ambiente è Kotlin.

https://developer.android.com/studio

2.2.5.4 Adobe Photoshop

Adobe Photoshop è un software per l'elaborazione di immagini, sarà utilizzato per progettare la GUI.

https://www.adobe.com/it/products/photoshop.html

2.2.5.5 Movens

Movens è una piattaforma pensata per la gestione delle smart cities. Verrà utilizzata per la gestione degli utenti e dell'applicazione.

https://www.adobe.com/it/products/photoshop.html

3 Processi di Supporto

3.1 Documentazione

3.1.1 Scopo

Ogni processo e attività significativi volti allo sviluppo del progetto sono documentati. Lo scopo di questa sezione è definire gli standard che riguardano i documenti prodotti durante il ciclo di vita del software. I documenti sono consultabili nelle apposite sezioni della repository$_G$: https://github.com/8LabSolutions/Soldino.

3.1.2 Aspettative

Le aspettative su questo processo riguardano:

- un'idea precisa relativa alla struttura della documentazione che deve essere prodotta durante il ciclo di vita del software;
- l'individuazione di una serie di norme per la stesura di documenti coerenti e validi.

3.1.3 Descrizione

Questo capitolo contiene le decisioni e le norme che sono state scelte per la stesura, verifica e approvazione della documentazione ufficiale. Tali norme sono tassative per tutti i documenti formali.

3.1.4 Ciclo di vita del documento

Ogni documento segue le fasi del seguente ciclo di vita:

- **Creazione del documento**: il documento viene creato, in conformità alle norme e adeguandosi a documenti precedenti dello stesso tipo. Viene usato un template reperibile dalla cartella "latex" della repository$_G$ remota;
- **Creazione della struttura**: viene creato un registro delle modifiche, un indice che traccia gli argomenti trattati nel documento, e a seconda delle necessità, un indice relativo alle figure e uno relativo alle tabelle presenti nel documento;
- **Realizzazione**: il documento viene scritto progressivamente in modo incrementale;
- **In revisione**: ogni voce del documento è prodotta interamente, ma è soggetta a verifiche per correggere, ampliare, semplificare quanto presente. La verifica di un frammento è svolta da almeno una persona, diversa da quella che ha scritto quel frammento;
- **Approvato**: il responsabile di progetto sancisce che il documento è stato completato, ed è quindi pronto per il rilascio.

3.1.5 Template

Il gruppo ha creato un template LaTeX per uniformare velocemente la struttura grafica e lo stile di formattazione dei documenti, in modo che i membri del team possano concentrarsi maggiormente, nella stesura del contenuto degli stessi. Lo scopo dei template è quello di permettere, a colui che redige il documento, di adottare automaticamente le conformità previste dalle *Norme di Progetto*. Permette inoltre di agevolare la procedura di adeguamento alle nuove norme per la redazione, nel caso esse cambiassero.

3.1.6 Struttura dei documenti

Un file "main.tex" (il cui termine "main" verrà sostituito dal nome del documento) raccoglie tramite comandi di input le sezioni di cui è composto il documento. Tra i file in input ci sono:

- "package.tex", che contiene i pacchetti necessari alla compilazione;
- "config.tex", contenente comandi LaTeX creati dal team.

3.1.6.1 Prima pagina

Il frontespizio è la prima pagina del documento ed è così strutturata:

- **Logo del gruppo**: logo di *8Lab Solutions* visibile come primo elemento centrato orizzontalmente in alto;
- **Gruppo e progetto**: nome del gruppo e del progetto *Soldino*, visibile centralmente subito sotto il logo;
- **Titolo**: nome del documento, posizionato centralmente in grassetto;
- **Tabella**: presente sotto il titolo del documento, centrale e contenente le seguenti informazioni:
  - **Versione**: versione del documento;
  - **Approvazione**: nome e cognome dei membri del gruppo incaricati dell'approvazione del documento;
  - **Redazione**: nome e cognome dei membri del gruppo incaricati della redazione del documento;
  - **Verifica**: nome e cognome dei membri del gruppo incaricati della verifica del documento;
  - **Stato**: stadio corrente del ciclo di vita del documento;
  - **Uso**: tipo d'uso che può essere "interno" o "esterno";
  - **Destinato a**: destinatari del documento.
- **Descrizione**: descrizione sintetica relativa al documento, centrale, posta sotto la tabella descrittiva;
- **Recapito**: indirizzo di posta elettronica del gruppo, posizionato centralmente in fondo alla pagina.

3.1.6.2 Registro modifiche

Ogni documento dispone di un changelog$_G$: una tabella posta a seguito della prima pagina che contiene le modifiche apportate al documento. Nel changelog$_G$sono indicati:

- versione del documento dopo la modifica;
- data della modifica;
- nominativo di chi ha modificato;
- ruolo di chi ha modificato;
- descrizione sintetica della modifica.

3.1.6.3 Indice

Gli indici hanno lo scopo di riepilogare e dare una visione macroscopica della struttura del documento, mostrando le parti gerarchiche di cui è composto.
Ogni documento è corredato dall'indice dei contenuti, posizionato dopo il changelog$_G$. Se sono presenti tabelle o immagini all'interno del documento, l'indice dei contenuti è seguito prima dalla lista delle figure, poi dalla lista delle tabelle.

3.1.6.4 Contenuto principale

La struttura delle pagine di contenuto è così strutturata:

- in alto a sinistra è presente il logo del gruppo;
- in alto a destra è riportato il nome del documento;
- una riga divide l'intestazione dal contenuto;
- il contenuto della pagina è posto tra l'intestazione e il piè di pagina;
- una riga divide il contenuto dal piè pagina;
- in basso a destra è posto il numero della pagina corrente.

3.1.6.5 Verbali

I verbali vengono prodotti dal/i soggetto/i incaricato/i alla loro stesura in occasione di incontri tra i membri del team con o senza la presenza di esterni. Per i verbali è prevista un'unica stesura. Tale scelta è motivata dal fatto che apportare modifiche implica una modifica delle decisioni prese in modo retroattivo. I verbali seguono la struttura degli altri documenti, ma il corpo del verbale è suddiviso in introduzione e contenuto. L'introduzione contiene:

- **Luogo**: luogo di svolgimento dell'incontro;
- **Data**: data dell'incontro(in formato YYYY-MM-DD);
- **Ora di inizio**: l'orario di inizio dell'incontro;
- **Ora di fine**: l'orario di fine dell'incontro;
- **Partecipanti**: l'elenco dei membri del gruppo che erano presenti all'incontro e, se presenti, i nominativi di persone esterne al gruppo che hanno partecipato. Un esempio, sono i nomi dei componenti di *Red Babel*, a fianco al quale deve essere indicato "(proponente)";
- **Argomenti affrontati**: ciò di cui si è discusso durante l'incontro in forma riassuntiva e facendo risaltare gli argomenti principali.

Il contenuto è composto da:

- una descrizione più approfondita in merito agli argomenti trattati durante l'incontro;
- un riepilogo dei tracciamenti in forma tabellare che elenca le decisioni emerse: assegna ad ognuna di loro un codice e le descrive. Sarà ripresa successivamente.

Ogni *Verbale* dovrà essere denominato secondo il seguente formato:

TipologiaYYYY-MM-DD

dove per "Tipologia" si intende il tipo di verbale:

- **Interno**: concentrato sul riassunto dell'incontro dei membri del team;
- **Esterno**: concentrato sulla trattazione di argomenti con partecipanti esterni al gruppo, in particolare domande e risposte riguardanti il progetto in sè.

La sezione "Riepilogo tracciamenti" avrà la funzione di tenere traccia delle decisioni emerse da ogni incontro, sotto forma di tabella riassuntiva a fine di ogni verbale. Il formalismo utilizzato dalla tabella dovrà essere il seguente:

Tipologia_X.Y

Dove:

- per "Tipologia" basterà indicare l'iniziale della tipologia del verbale(I=Interno/E=Esterno);
- "X.Y": X si riferisce al numero dell'incontro mentre per Y si intende il numero progressivo della decisione presa dal gruppo(partendo da 1).

3.1.6.6 Note a piè di pagina

In caso di presenza di note da esplicare, esse vanno indicate nella pagina corrente, in basso a sinistra. Ogni nota deve riportare un numero e una descrizione.

3.1.7 Norme tipografiche

3.1.7.1 Convenzioni sui nomi dei file

I nomi di file (estensione esclusa) e cartelle utilizzano la convenzione "Snake case$_G$" e alcune regole aggiuntive elencate di seguito:

1. i nomi dei file composti da più parole usano il carattere underscore come carattere separatore;
2. i nomi sono scritti interamente in minuscolo;
3. le preposizioni non si omettono.

Alcuni esempi **corretti** sono:

- studio_di_fattibilità;
- analisi_dei_requisiti.

Alcuni esempi **non corretti** sono:

- Norme_di_progetto (usa maiuscole);
- norme-di-progetto (carattere separatore errato);
- norme_progetto (omette "di").

3.1.7.2 Glossario

- ogni termine del *Glossario* è marcato con una **G** maiuscola a pedice in ogni sua occorrenza;
- se la voce è presente ripetutamente nello stesso paragrafo non è necessario marcarla in ogni sua occorrenza, ma è possibile indicare la **G** a pedice solo nella prima occorrenza;
- se nel *Glossario v1.0.0* un termina presenta una descrizione che utilizza termini da glossario, è necessario trattare questi termini come tali, segnando la **G** a pedice e aggiungendoli al documento con la relativa descrizione;
- non vengono segnate con la **G** a pedice le parole da *Glossario* presenti nei titoli e nelle didascalie di immagini e tabelle.

3.1.7.3 Stile del testo

- **Grassetto**: viene applicato se necessario alle voci di un elenco puntato, a titoli o a termini di frasi su cui si vuol far ricadere l'attenzione del lettore;
- **Corsivo**: vengono scritti in corsivo il nome del progetto *Soldino*, il nome del gruppo *8Lab Solutions* ed il nome del gruppo dei proponenti, ovvero *Red Babel*;
- **Maiuscolo**: vengono scritti con sole lettere maiuscole tutti gli acronimi. Nel caso di nomi o titoli composti da più parole verrà indicato con la lettera maiuscola solamente la prima lettera della prima parola, lasciando in minuscolo il restante (esclusi nomi dei documenti che sono normati secondo quanto segue);
- **Nomi dei documenti**:
  - ogni volta che si cita un documento, si deve indicare con la lettera maiuscola le iniziali dei nomi di cui è composto (*e.g.* Analisi dei Requisiti), ma senza specificare la versione, riportando il tutto in corsivo;
  - quando si fa riferimento al documento vero e proprio o a qualcosa in esso contenuto si segue la prima convenzione ma si aggiunge la versione del documento v*.0.0 (separata da uno spazio), anch'essa in corsivo;
  - ogni volta che si utilizza il documento come titolo o in una voce di elenco seguita da una descrizione, si deve seguire la prima convenzione ma senza utilizzare il corsivo.

3.1.7.4 Elenchi puntati

Ogni voce di un elenco comincia per lettera **minuscola**, a patto che non subentrino altre norme, e termina per **";"**, eccetto l'ultima che termina per **"."**. I sottoelenchi innestati dentro una voce di elenco, rispettano le medesime regole, poiché la loro funzione è analoga.
Se le voci dell'elenco sono della forma *termine - descrizione*, si pongono i termini in grassetto.

3.1.7.5 Formati comuni

In conformità allo standard ISO 8601, le date devono essere scritte secondo il formato:

YYYY-MM-DD

- **YYYY**: rappresentazione dell'anno con quattro (4) cifre;

- **MM**: rappresentazione del mese con due (2) cifre;
- **DD**: rappresentazione del giorno con due (2) cifre.

3.1.7.6 Sigle

Il progetto prevede la redazione di un insieme di documenti, suddivisi in documenti interni e documenti esterni. Essi sono elencati di seguito con le rispettive sigle.
I documenti esterni sono:

- **Analisi dei Requisiti - AdR**: stabilisce le caratteristiche che il software deve rispettare;
- **Manuale Utente - MU**: ad uso degli utilizzatori del software;
- **Manuale Sviluppatore - MS**: per gli sviluppatori e manutentori;
- **Piano di Progetto - PdP**: concerne la gestione del progetto, evidenziandone la fattibilità e le criticità; tratta di tempi, costi, obiettivi, rischi, vincoli;
- **Piano di Qualifica - PdQ**: : descrive la qualità del software e dei processi, e come la si intende raggiungere mediante l'uso di strumenti, metriche e processi stessi.

I documenti interni sono:

- **Glossario - G**: raccoglie i termini di interesse per il team di sviluppo e sui quali è necessaria una descrizione che ne chiarisca il significato;
- **Norme di Progetto - NdP**: sono un riferimento normativo per lo svolgimento delle attività di progetto;
- **Studio di Fattibilità - SdF**: descrive sommariamente i capitolati e spiega la loro scelta o esclusione.

Un caso particolare di documenti, sono i verbali, che possono essere esterni o interni:

- **Verbale - V**: descrive le interazioni avvenute durante un incontro con il proponente (verbale esterno) o tra i membri del team (verbale interno); sono orientati a dare informazioni semplici, di veloce lettura e complete.

Le diverse fasi del progetto sono le seguenti, accompagnate dalle relative sigle:

- **Revisione dei Requisiti - RR**: studio iniziale del capitolato$_G$, se ben fatto permette al gruppo di aggiudicarselo;
- **Revisione di Progettazione - RP**: riguarda la definizione dell'architettura del software e di una Proof of Concept$_G$ per mostrare la fattibilità;
- **Revisione di Qualifica - RQ**: riguarda la definizione dettagliata e la codifica del prodotto;
- **Revisione di Accettazione - RA**: se il prodotto soddisfa i requisiti del proponente, viene accettato e rilasciato.

Altre sigle presenti nei documenti interessano i ruoli assunti dai componenti del team:

- **responsabile di progetto - Re**;

- **amministratore - Ad**;
- **analista - An**;
- **progettista - Pt**;
- **programmatore - Pr**;
- **verificatore - Ve**;

3.1.8 Elementi grafici

3.1.8.1 Tabelle

In tutti i documenti LaTeX le tabelle sono scritte allo stesso modo: fare riferimento alla Wiki su GitHub per i comandi necessari.
Ogni tabella deve essere accompagnata dalla propria didascalia descrittiva, la "caption", da posizionare subito sopra la tabella a cui si riferisce. É stata scelta questa convenzione per adattarsi agli standard usati dalla maggioranza della community di LaTeX; nella didascalia deve comparire il numero della sezione a cui si riferisce, seguita in modo incrementale dal numero progressivo delle tabelle di quella sezione.

- **X.Y**: rappresenta la sezione;
- **Z**: rappresenta il numero progressivo della tabella nella sezione.

Fanno eccezione le tabelle dei changelog$_G$ che non hanno didascalia e le tabelle dei casi d'uso presenti nel documento *Analisi dei Requisiti*.

3.1.8.2 Immagini

Le immagini sono centrate e hanno una didascalia descrittiva.

3.1.8.3 Diagrammi UML

Tutti i diagrammi UML$_G$ vengono inseriti nei documenti sotto forma di immagine.

3.1.9 Strumenti

3.1.9.1 LaTeX

Lo strumento scelto per la scrittura di documenti è LaTeX, un linguaggio basato sul programma di composizione tipografica TeX che permette di scrivere documenti in modo ordinato, modulare, collaborativo e scalabile.

3.1.9.2 TeXstudio

Per la stesura del codice LaTeX è stato utilizzato l'editor TeXstudio. Questo strumento, oltre ad integrare un compilatore e un visualizzatore PDF, fornisce suggerimenti di completamento per comandi LaTeX.

https://www.texstudio.org/

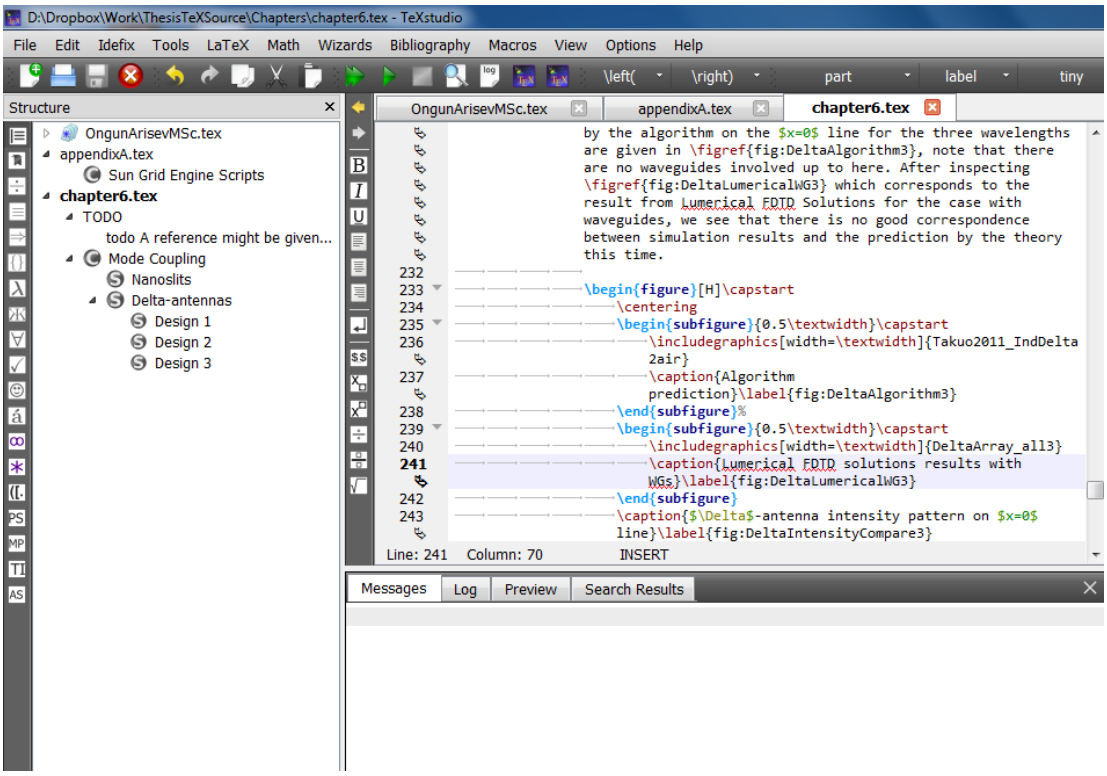

Figura 3.1.1: TEXstudio - per la stesura dei documenti

3.1.9.3 Draw.io

Draw.io, già citato in precedenza, viene utilizzato per la produzione degli UML$_G$.
https://www.draw.io/

3.2 Gestione della configurazione

3.2.1 Scopo

Lo scopo della configurazione è di creare ordine tra software e documenti. Tutto ciò che è configurato si trova in un posto preciso, ha uno stato identificativo, è modificato secondo procedure e posto sotto versionamento.

3.2.2 Versionamento

3.2.2.1 Codice di versione del documento

Ogni documento, deve avere una storia, ricostruibile attraverso le sue versioni. Ogni versione deve corrispondere a una riga della tabella delle modifiche. Ogni numero di versione è composto da tre cifre:

X.Y.Z

- **X**: rappresenta una versione stabile del documento, resa tale dopo l' approvazione del responsabile di progetto:
  - inizia da 0;
  - viene incrementato dal responsabile di progetto all'approvazione del documento.
- **Y**: rappresenta una versione parzialmente stabile del documento che è stata soggetta a verifica da parte di un verificatore:
  - inizia da 0;
  - viene incrementato dal verificatore ad ogni verifica;
  - quando viene incrementato X, viene riportato a 0.
- **Z**: rappresenta una versione instabile del documento in fase di lavorazione da parte dei redattori:
  - inizia da 0;
  - viene incrementato dal redattore del documento ad ogni modifica;
  - quando viene incrementato Y, viene riportato a 0.

3.2.2.2 Tecnologie

Per le parti del progetto da versionare si è scelto di usare il sistema di versionamento distribuito Git, usando il servizio di GitHub per ospitare la repository$_G$ remota.

3.2.2.3 Repository

I membri del team possono interagire con il VCS (Version Control System) sia da linea di comando, sia attraverso software che ne migliorano l'usabilità, come GitFlow e GitKraken. La versione comune e ufficiale del progetto è ospitata in una repository$_G$ remota su Github all'indirizzo:

https://github.com/8LabSolutions/Soldino

3.2.2.4 Struttura del repository

Ci sono due tipi di repository$_G$, la cui struttura è identica:

- **locale**: ogni membro del gruppo lavora sui file clonati dal repository$_G$ remoto nel proprio computer;
- **remoto**: pubblicato su GitHub, contiene il lavoro svolto da ogni componente e che viene condiviso con il team.

Entrambe i tipi di repository$_G$ sono organizzati in cartelle:

- **latex**: contiene tutti i file che definiscono il template LaTeX per la creazione di un nuovo documento. Tra questi vi sono i file che definiscono i comandi personalizzati utilizzati dal gruppo e i package che devono essere inclusi per poter compilare i documenti;
- **Guide**: cartella che contiene delle brevi guide di carattere pratico, come brevi spiegazioni per l'utilizzo dei comandi LaTeX poco conosciuti;

- **RR**: raccoglie i file sorgenti per la compilazione dei documenti, suddivisi tra esterni ed interni, realizzati per la Revisione dei Requisiti;
- Saranno aggiunte in futuro cartelle distinte nominate **RP**, **RQ** e **RA** contenenti i file delle rispettive consegne.

La suddivisione dei file per revisione aiuta ad evidenziare il lavoro svolto per ogni consegna e migliora la tracciabilità dei file all'interno di ognuna di esse.

3.2.2.5 Tipi di file e .gitignore

I file utilizzati per la documentazione del progetto sono:

- file con estensione .tex di LaTeX;
- file con estensione .pdf che sono oggetto di consegna;
- alcuni file testuali e immagini di supporto ai precedenti.

Il file ".gitignore" è presente al livello più esterno della repository$_G$ ed elenca tutti i file da escludere dal versionamento.

3.2.2.6 Utilizzo di Git

Il repository$_G$ di Git è composto da vari branch, per lavorare in modo modulare e collaborativo. Quindi viene consiglia questa procedura:

1. scegliere il branch su cui si vuole lavorare in base al compito assegnato;
2. eseguire il pull dal repository$_G$ remoto che effettua l'aggiornamento del proprio repository$_G$ locale;
3. svolgere il lavoro assegnato, che può consiste nell'aggiunta di nuovo materiale o nella modifica di file già presenti;
4. eseguire il comando di aggiunta dei file nuovi o modificati da condividere all'area di staging$_G$;
5. eseguire il comando di commit dei file aggiunti, corredato da un messaggio che identifica il task eseguito;
6. eseguire il push del commit sul repository$_G$ remoto.

3.2.2.7 Gestione delle modifiche

Tutti i membri del team possono modificare i file in ogni branch, ad eccezione di quelli presenti nel branch master, per il quale occorre fare una pull request$_G$ e ottenere l'approvazione di un altro membro .
Per effettuare modifiche maggiori sui contenuti o sulla struttura dei file è previsto di:

- contattare il responsabile del file da modificare;
- suggerire la modifica da effettuare;

- se il responsabile valuta positivamente la modifica, allora la applica.

Per modifiche minori, come correzioni grammaticali o miglioramenti sintattici, si consiglia di modificare indipendentemente.
In ambo i casi è opportuno commentare i propri commit con chiarezza per risalire facilmente alle modifiche.

3.3 Gestione della Qualità

3.3.1 Scopo

Lo scopo è di garantire che il prodotto e i servizi offerti rispettino gli obiettivi di qualità e che i bisogni del proponente siano soddisfatti.

3.3.2 Aspettative

Attraverso la gestione della qualità si intende ottenere:

- qualità nell'organizzazione e nei suoi processi;
- qualità nel prodotto;
- qualità provata oggettivamente;
- soddisfazione finale di cliente e proponente.

3.3.3 Descrizione

Per la trattazione approfondita della gestione della qualità si fa riferimento al *Piano di Qualifica v1.0.0*, dove sono descritte le modalità utilizzate per garantire la qualità nello sviluppo del progetto. In particolare, in tale documento:

- sono presentati gli standard utilizzati;
- sono individuati i processi di interesse negli standard;
- sono individuati gli attributi del software più significativi per il progetto.

Per ogni processo vengono descritti:

- gli obiettivi da perseguire;
- le strategie da applicare;
- le metriche da utilizzare.

L'obiettivo è quello di controllare e migliorare i processi e i loro risultati. Per ogni prodotto vengono associati:

- gli obiettivi da perseguire;
- le metriche da utilizzare.

In questo caso si mira ad ottenere software e documentazione di qualità soddisfacente.

3.3.4 Attività

Le tre attività principali del processo sono:

- **Pianificazione**: porsi degli obiettivi di qualità, definire strategie per raggiungerli e disporre di conseguenza le persone e le risorse nel modo migliore;
- **Valutazione**: mettere in atto la pianificazione, applicando i criteri, misurando e monitorando i risultati;
- **Reazione**: sulla base dei risultati, adattare le proprie strategie, criteri e piani.

3.3.5 Strumenti

Gli strumenti predefiniti per la qualità sono:

- forniti dallo standard **ISO-12207**;
- le metriche.

3.4 Verifica

3.4.1 Scopo

Il processo di verifica ha per scopo la realizzazione di prodotti corretti, coesi e completi. Sono soggetti a verifica il software e i documenti.

3.4.2 Aspettative

Il corretto svolgimento del processo di verifica rispetta i punti seguenti:

- viene effettuata seguendo procedure definite;
- vi sono criteri chiari e affidabili da seguire per verificare;
- i prodotti passano attraverso fasi successive, ognuna delle quali è verificata;
- dopo la verifica il prodotto è in uno stato stabile;
- rende possibile la validazione.

3.4.3 Descrizione

Il processo di verifica prende in input ciò che è già stato prodotto e lo restituisce in uno stato conforme alle aspettative. Per ottenere tale risultato ci si affida a processi di analisi e test.

3.4.4 Attività

3.4.4.1 Analisi

Il processo di analisi si suddivide in statica e dinamica.

Analisi statica

L'analisi statica effettua controlli su documenti e codice, di cui valuta e applica la correttezza

(intesa come assenza di errori e difetti), la conformità a regole e la coesione$_G$ dei componenti. Per effettuare analisi statica esistono metodi manuali di lettura (attuati da persone) e metodi formali (attuati da macchine). Quelli manuali sono due:

- **Walkthrough**: i vari componenti del team analizzano gli oggetti nella loro totalità per cercare anomalie, senza sapere inizialmente se vi siano difetti, quali e dove siano;
- **Inspection**: i verificatori usano liste di controllo per fare ispezione cercando errori specifici in parti specifiche.

A seguire sono descritte le liste di controllo utilizzabili per le ispezioni. Si prevede l'ampliamento delle viste al progredire delle stesse ispezioni e degli errori trovati.

Tabella 3.4.1: Errori frequenti nei documenti

Oggetto	Controllo
Formato data	Deve seguire il formato americano *YYYY-MM-DD*
Sintassi	La frase è troppo complessa e può essere semplificata
Parte mancante	Controllare titoli vuoti o sezioni di grandezza insolita
Forma dei verbi	In generale è preferibile il presente indicativo
Punteggiatura degli elenchi	Ogni voce termina in ";" eccetto l'ultima che termina in "."

Analisi dinamica

L'analisi dinamica è una tecnica di analisi del prodotto software che richiede la sua esecuzione. Viene effettuata mediante dei test che verificano se il prodotto funziona e se ci sono anomalie.

3.4.4.2 Test

I test sono l'attività fondamentale dell'analisi dinamica: il loro scopo è verificare che il codice scritto funzioni correttamente. Per ogni test devono essere definiti i seguenti parametri:

- **ambiente**: sistema hardware e software sul quale verrà eseguito il test;
- **stato iniziale**: lo stato iniziale dal quale il test viene eseguito;
- **input**: input inserito;
- **output**: output atteso;
- **istruzioni aggiuntive**: ulteriori istruzioni su come va eseguito il test e su come vanno interpretati i risultati ottenuti.

Test ben scritti devono:

- essere ripetibili;
- specificare l'ambiente di esecuzione;
- identificare input e output richiesti;

- avvertire di possibili effetti indesiderati;
- fornire informazioni sui risultati dell'esecuzione in forma di file di log.

Ci sono vari tipi di test del software, ognuno dei quali ha un diverso oggetto di verifica e scopo.

Test di unità

I test di unità si eseguono su unità di software. Questi test si concentrano sul funzionamento delle unità individuali. Dati gli input possibili in un unità, si suddividono gli input in partizioni di equivalenza e si prova se gli input attesi danno gli output previsti. Le singole unità possono essere testate con l'ausilio di driver$_G$ e stub$_G$.

Test di integrazione

Dopo aver superato i test di unità, le stesse unità vengono assemblate in agglomerati progressivamente più grandi. Il test di integrazione si concentra sulle interfacce tra i componenti. Un agglomerato che supera il test di integrazione costituisce quindi una nuova unità per un agglomerato di grandezza maggiore. Questa procedura si ripete fino a raggiungere la dimensione totale del sistema.

Test di sistema

Il sistema viene testato nella sua interezza, una volta integrati tutti i suoi componenti. Ci si concentra sulle interazioni tra le parti, sul comportamento emergente$_G$ delle caratteristiche del sistema e sulla copertura$_G$ di tutte le funzionalità. Questo tipo di test porta alla luce le ipotesi errate che i diversi sviluppatori fanno su parti del software sviluppate da altri. In questa fase ci si assicura che il sistema rispetti tutte le specifiche definite nell'*Analisi dei Requisiti*.

Test di regressione

Si effettua di seguito ad una modifica del sistema e consiste nella riesecuzione dei test esistenti: si combina bene con l'automazione dei test.

Test di accettazione

Anche detto "test di collaudo", è simile al test di sistema per l'oggetto testato, ma viene eseguito con la collaborazione dei committenti. Si occupa di verificare il prodotto e, in particolare, il soddisfacimento del cliente. Il superamento del test di collaudo garantisce che il software sia pronto per essere rilasciato.

3.4.5 Strumenti

3.4.5.1 Verifica ortografica

Viene utilizzata la verifica dell'ortografia in tempo reale, strumento integrato in TEXstudio che sottolinea in rosso le parole errate e in verde le ripetizioni a breve distanza secondo la lingua italiana.

3.4.5.2 Validazione W3C

Per la validazione delle pagine di markup HTML viene utilizzato lo strumento offerto dal W3C, raggiungibile al seguente indirizzo:

https://validator.w3.org/

Per la validazione dei fogli di stile CSS viene utilizzato lo strumento offerto dal W3C, raggiungibile al seguente indirizzo:

https://jigsaw.w3.org/css-validator/

3.5 Validazione

3.5.1 Scopo

Lo scopo della validazione è stabilire se il prodotto soddisfa il compito per cui è stato creato. Dopo la validazione, è garantito che il software rispetti i requisiti e che soddisfa i bisogni del committente.

3.5.2 Attività

Per validare il prodotto si devono rispettare i seguenti punti:

- identificare gli oggetti da validare;
- identificare una strategia con delle procedure di validazione in cui le procedure di verifica possono essere riutilizzare;
- applicare la strategia;
- valutare che i risultati rispettino le aspettative.

4 Processi Organizzativi

4.1 Gestione Organizzativa

4.1.1 Scopo

Lo scopo di questo processo è:

- creare un modello organizzativo volto a specificare i rischi che si possono verificare;
- definire un modello di sviluppo$_G$ da seguire;
- pianificare il lavoro in base alle scadenze;
- calcolare un prospetto economico suddiviso per ruoli;
- effettuare un bilancio finale sulle spese.

Tutte queste attività sono a carico del responsabile di progetto e devono essere raccolte nel documento *Piano di Progetto*.

4.1.2 Aspettative

Gli obiettivi di questo processo sono:

- ottenere una pianificazione ragionevole delle attività da seguire;
- coordinare i membri del team assegnando loro ruoli, compiti e facilitando la comunicazione tra loro;
- adoperare processi per regolare le attività e renderle economiche;
- mantenere il controllo sul progetto in modo efficace ma non invasivo, monitorando il team, i processi e i prodotti.

4.1.3 Descrizione

Le attività di gestione sono:

- inizio e definizione dello scopo;
- istanziazione dei processi;
- pianificazione e stima di tempi, ricorse e costi;
- assegnazione dei ruoli e dei compiti;
- esecuzione e controllo;
- revisione e valutazione periodica delle attività.

4.1.4 Ruoli di progetto

Ciascun membro del gruppo, a rotazione, deve ricoprire il ruolo che gli viene assegnato e che corrisponde all'omonima figura aziendale. Nel *Piano di Progetto* vengono organizzate e pianificate le attività assegnate agli specifici ruoli. I ruoli che ogni componente del gruppo è tenuto a rappresentare sono descritti di seguito.

4.1.4.1 Responsabile di progetto

Il responsabile di progetto è una figura importante in quanto ricadono su di lui le responsabilità di pianificazione, gestione, controllo e coordinamento. Un altro compito del responsabile è quello interfacciare il gruppo con le persone esterne, facendo da intermediario: sono quindi di sua competenza le comunicazioni con committente e proponente.
Riassumendo, si occupa di:

- gestire, controllare, coordinare risorse e attività del gruppo;
- gestire, controllare, coordinare gli altri componenti del gruppo;
- analizzare e gestire le criticità;
- approvare i documenti.

4.1.4.2 Amministratore di progetto

L'amministratore ha il compito di supporto e controllo dell'ambiente di lavoro.
Egli deve quindi:

- dirigere le infrastrutture di supporto;
- risolvere problemi legati alla gestione dei processi;
- gestire la documentazione;
- controllare versioni e configurazioni.

4.1.4.3 Analista

L'analista si occupa di analisi dei problemi e del dominio applicativo. Questa figura non sarà sempre presente durante il progetto.
Le sue responsabilità sono:

- studio del dominio del problema;
- definizione della complessità e dei requisiti dello stesso;
- redazione del documenti: *Analisi dei Requisiti* e *Studio di Fattibilità*.

4.1.4.4 Progettista

Il progettista gestisce gli aspetti tecnologici e tecnici del progetto.
Il progettista deve:

- effettuare scelte efficienti ed ottimizzate su aspetti tecnici del progetto;
- sviluppare un'architettura che sfrutti tecnologie note ed ottimizzate, su cui basare un prodotto stabile e mantenibile.

4.1.4.5 Programmatore

Il programmatore è responsabile della codifica del progetto e delle componenti di supporto che serviranno per effettuare le prove di verifica e validazione sul prodotto.
Il programmatore si occupa di:

- implementare le decisioni del progettista;
- creare o gestire componenti di supporto per la verifica e validazione del codice.

4.1.4.6 Verificatore

Il verificatore si occupa di controllare il prodotto del lavoro svolto dagli altri membri del team, sia esso codice o documentazione. Per le correzioni si affida agli standard definiti nelle *Norme di Progetto v1.0.0*, nonché alla propria esperienza e capacità di giudizio.
Il verificatore deve:

- ispezionare i prodotti in fase di revisione, avvalendosi delle tecniche e degli strumenti definiti nelle *Norme di Progetto*;
- evidenziare difetti ed errori del prodotto in esame;
- segnalare eventuali errori trovati all'autore dell'oggetto preso in esame o alla persona che ha responsabilità su di esso.

4.1.5 Procedure

Seguono le procedure che il gruppo adotterà durante la realizzazione del progetto. Le comunicazioni saranno interne, ossia coinvolgeranno i partecipanti del gruppo, o esterne, ossia includeranno anche proponente e committente.

4.1.5.1 Gestione delle comunicazioni

Comunicazioni interne

Le comunicazioni interne al gruppo avvengono utilizzando Slack$_G$. Slack$_G$ è un software di collaborazione aziendale adatto al lavoro di gruppo. Permette di separare i vari argomenti di discussione in canali e ogni team ha la possibilità di creare il proprio workspace. Inoltre permette di integrare servizi di terze parti utili al gruppo, tra cui bot e videochiamate. Le sue funzionalità e la sua conformità agli scopi del progetto ne hanno indotto la scelta.

Comunicazioni esterne

Le comunicazioni con soggetti esterni al gruppo sono di competenza del responsabile. Gli strumenti predefiniti sono la posta elettronica, dove viene utilizzato l'indirizzo 8labsolutions@gmail.com. Per comunicare con *Red Babel* si utilizza un canale Slack$_G$ per la chat testuale e il servizio Google Hangouts per le chiamate. Il responsabile ha il compito di tenere informati gli altri componenti del gruppo in caso di assenza.

4.1.5.2 Gestione degli incontri

Incontri interni del team

Le riunioni interne del team sono organizzate dal responsabile in accordo con i membri del gruppo. Per stabilire data e ora viene utilizzato Google Calendar, dove ogni membro segnala gli orari nei quali non è disponibile.

Verbali di riunioni interne

Ad ogni riunione interna corrisponde un *Verbale*. Questo sarà redatto da un segretario, persona nominata dal responsabile, che dovrà tenere nota delle discussioni fatte e delle decisioni prese.

Incontri esterni del team

Il responsabile ha il compito di comunicare e organizzare gli incontri con proponente e committente. Se un membro del gruppo, il proponente o il committente ritiene necessario organizzare un incontro allora il responsabile decide una data, in accordo tra le due parti, e la comunica tramite i canali sopra citati.

Verbali di riunioni esterne

Come per le riunioni interne, anche per le esterne viene redatto un *Verbale*. La struttura delle due tipologie è analoga, ma le riunioni esterne presentano una maggiore criticità per la presenza di persone esterne al gruppo, quali il committente e il proponente.

4.1.5.3 Gestione degli strumenti di coordinamento

Tickecting

Il ticketing consente ai membri di avere chiaro in ogni momento quali attività sono in corso; permette al responsabile di progetto di assegnare compiti ai membri del team e di controllare l'andamento delle attività. Inoltre permette ai membri del team di conoscere e gestire il proprio carico di lavoro.
Lo strumento di ticketing scelto è Trello: consiste in una lavagna virtuale online dove sono esposti i task. Ad ogni task sono associati una data di scadenza e un insieme di membri assegnatari. Ogni compito passa attraverso i seguenti stati:

- da fare (to do);
- in lavorazione (doing);
- in revisione;
- completato (done).

La scelta di Trello è stata determinata dalla sua facilità di apprendimento, di controllo, dall'usabilità e dalla trasparenza che fornisce. La gestione dei ticket dev'essere scrupolosa, perché la lavagna di Trello tende ad affollarsi velocemente.

4.1.5.4 Gestione dei rischi

Il responsabile di progetto ha il compito di rilevare i rischi e di renderli noti, documentando quest'attività nel *Piano di Progetto*. La procedura da seguire per la gestione dei rischi è la seguente:

- individuare nuovi problemi e monitorare i rischi già previsti;
- registrare ogni riscontro previsto dei rischi nel *Piano di Progetto*;
- aggiungere i nuovi rischi individuati nel *Piano di Progetto*;
- ridefinire, se necessario, le strategie di gestione dei rischi.

Codifica dei rischi

Le tipologie di rischi sono così codificate:

- **RT**: Rischi Tecnologici;
- **RO**: Rischi Organizzativi;
- **RI**: Rischi Interpersonali.

4.1.6 Strumenti

Il gruppo, nel corso del progetto, ha utilizzato o utilizzerà i seguenti strumenti:

- **Telegram**$_G$: strumento di messaggistica utilizzato inizialmente per la gestione del gruppo;
- **Slack**$_G$: per la comunicazione interna del team ed eventuali comunicazioni col proponente;
- **Trello**: per assegnare determinate attività e imporre una scadenza;
- **Git**: sistema di controllo di versionamento;
- **Gitflow, GitKraken**: interfacce per utilizzare Git più comodamente sul proprio desktop;
- **GitHub**: per il versionamento e il salvataggio in remoto di tutti i file riguardanti il progetto.
- **Google Drive**: utilizzato per la stesura di file che sono soggetti a molti cambiamenti e devono essere visibili a tutti nella loro versione più aggiornata, come ad esempio il *Glossario*;
- **Google Calendar**: per facilitare il lavoro al responsabile, ogni settimana ciascun membro indica quando non è disponibile, in modo da semplificare l'organizzazione degli incontri;
- **Skype**: servizio che offre possibilità di fare videoconferenze e chiamate VoIP, utilizzato per il primo contatto con il proponente;
- **Hangouts**: servizio che offre la possibilità di fare videoconferenze e chiamate VoIP, utilizzato per parlare con il proponente e per alcuni incontri interni;
- **Sistemi operativi**: i requisiti non indicano la necessità di usare un sistema operativo specifico, verranno quindi utilizzati Windows, Linux e Mac OS dai diversi membri del team.