Norme di progetto v2.0.0

WarMachine – Progetto IronWorks

warmachine.swe@gmail.com

Informazioni sul documento:

Versione	2.0.0
Data di creazione	03/03/2018
Redazione	Cisternino Nicola, Coletti Andrea, Bernucci Riccardo
Verifica	Zanon Elena, Fogarollo Stefano, Bragagnolo Leonardo
Approvazione	Fogarollo Stefano
Uso	Interno
Distribuzione	*WarMachine*

Diario delle modifiche

Versione	Data	Collaboratori	Ruolo	Descrizione
2.0.0	07/06/2018	Fogarollo Stefano	*Responsabile di progetto*	Approvazione documento.
1.7.0	06/06/2018	Fogarollo Stefano	*Verificatore*	Verificato intero documento. Niente da segnalare.
1.6.0	16/05/2018	Zanon Elena	*Verificatore*	Verificate §[2.2.6], §[3.2]. Niente da segnalare.
1.5.0	14/05/2018	Zanon Elena, Bragagnolo Leonardo	*Verificatore*	Verificata §[2.2.5]. Niente da segnalare.
1.4.1	05/05/2018	Bernucci Riccardo	*Amministratore*	Scritte §[2.2.6.6], §[2.2.6.9], §[2.2.7.2], §[2.2.7.3]. Correzioni grammaticali in §[3].
1.4.0	04/05/2018	Bragagnolo Leonardo	*Verificatore*	Verificate §[3.3.10], §[3.3.11] e §[3.3.12]. Da rivedere grammatica in §[3].
1.3.1	04/05/2018	Bernucci Riccardo	*Amministratore*	Scritte §[3.3.10], §[3.3.11] e §[3.3.12]. Corretta §[3.2.5.2].
1.3.0	03/05/2018	Zanon Elena	*Verificatore*	Verificate §[2.2.5], §[3.2.5.2], §[3.2.5.3]. Da rivedere §[3.2.5.2].

1.2.1	03/05/2018	Coletti Andrea	*Amministratore*	Scritte §[2.2.5], §[3.2.5.2] e §[3.2.5.3]. Corretta §[3.1.7.3].
1.2.0	02/05/2018	Zanon Elena	*Verificatore*	Verificate §[3.1.6.5], §[3.1.7.3] e §[4.2]. Da rivedere §[3.1.7.3].
1.1.1	02/05/2018	Cisternino Nicola	*Amministratore*	Aggiunta descrizioni verbi permessi all'interno del diario delle modifiche sezione §[3.1.6.5]. Aggiunto strumento W3schools a sezione §[4.2]. Aggiunto acronimo rischio in §[3.1.7.3].
1.1.0	01/05/2018	Fogarollo Stefano	*Verificatore*	Verificate §[3.2], §[3.4], §[4.2]. Da rivedere §[3.2].
1.0.1	27/04/2018	Cisternino Nicola	*Amministratore*	Scritta §[3.2],§[3.4]. Modificata §[4.2]. Aggiunte da §[4.2.1] a §[4.2.4].
1.0.0	09/03/2018	Bragagnolo Leonardo	*Responsabile di progetto*	Approvazione del documento.
0.3.0	09/03/2018	Coletti Andrea, Zanon Elena	*Verificatore*	Verificato intero documento.

0.2.2	09/03/2018	Cisternino Nicola	*Amministratore*	Aggiunta metodo di miglioramento continuo dei processi: ciclo di deming$_{G}$. Aggiunta descrizione Astah e PragmaDB.
0.2.1	09/03/2018	Cisternino Nicola	*Amministratore*	Aggiunti test unità, sistema$_{G}$, integrazione, regressione e accettazione. Modiche SPI e CPI con SV e CV.
0.1.1	08/03/2018	Cisternino Nicola	*Amministratore*	Correzioni grammaticali e di sintassi §[1], §[2], §[3], §[4]. Modificata e corretta §4.1.9.5.
0.2.0	08/03/2018	Coletti Andrea	*Verificatore*	Rivedere grammatica e sintassi, in particolare: §[1], §[2], §[3] e §[4].
0.0.8	07/03/2018	Zanetti Ilenia	*Amministratore*	Correzione §[4].
0.1.2	07/03/2018	Cisternino Nicola	*Amministratore*	Aggiunte sigle dei nominativi dell'analisi dei rischi$_{G}$ sul *Piano di progetto*.
0.1.1	06/03/2018	Zanon Elena	*Verificatore*	Verificate: §[1], §[2], §[3]. Da rivedere §[4], in particolare §[4.1.5.1] Comunicazioni interne.

0.1.0	06/03/2018	Cisternino Nicola	*Amministratore*	Aggiunta metrica per errori concettuali nel testo §[3.2]. Correzioni errori non conforme alle norme stabilite e errori di ortografia.
0.0.5	05/03/2018	Zanetti Ilenia	*Amministratore*	Modificati canali di coordinamento, aggiunta §[2.2.4] Analisi dei requisiti.
0.0.4	05/03/2018	Cisternino Nicola	*Amministratore*	Correzione incomprensioni processi di supporto.
0.0.3	04/03/2018	Cisternino Nicola, Zanetti Ilenia	*Amministratore*	Completate §[3], §[4] aggiunta di §[4.1.9.7], revisione attività$_G$ di analisi §[3.2].
0.0.2	04/03/2018	Cisternino Nicola	*Amministratore*	Scritte §[3] e §[4].
0.0.1	03/03/2018	Cisternino Nicola	*Amministratore*	Creazione documento. Scritte §[1] e §[2].

Indice

1 Introduzione **11**
1.1 Scopo del Documento . . . 11
1.2 Struttura del documento . . . 11
1.3 Scopo del prodotto$_G$ . . . 11
1.4 Glossario . . . 12
1.5 Riferimenti . . . 12
1.5.1 Normativi . . . 12
1.5.2 Informativi . . . 12

2 Processi primari **13**
2.1 Processo$_G$ di fornitura . . . 13
2.1.1 Scopo del processo . . . 13
2.1.2 Aspettative . . . 13
2.1.3 Descrizione del processo . . . 13
2.1.4 Accordo contrattuale . . . 13
2.1.5 Modalità di esecuzione del contratto . . . 13
2.1.6 Consegna e manutenzione del prodotto . . . 14
2.1.7 Studio di Fattibilità . . . 14
2.2 Processo di sviluppo . . . 14
2.2.1 Scopo del processo . . . 14
2.2.2 Aspettative . . . 14
2.2.3 Descrizione del processo . . . 15
2.2.4 Analisi dei requisiti . . . 15
2.2.4.1 Scopo dell'attività . . . 15
2.2.4.2 Aspettative . . . 15
2.2.4.3 Descrizione dell'attività . . . 15
2.2.4.4 Casi d'uso . . . 15
2.2.4.5 Codice identificativo . . . 16
2.2.4.6 Requisiti . . . 16
2.2.4.7 Codice identificativo . . . 16
2.2.4.8 Strumenti utili al tracciamento . . . 17
2.2.5 Progettazione . . . 17
2.2.5.1 Scopo dell'attività . . . 17
2.2.5.2 Aspettative . . . 17
2.2.5.3 Descrizione dell'attività . . . 17
2.2.5.4 Progettazione architetturale . . . 17
2.2.5.5 Progettazione di dettaglio . . . 18
2.2.5.6 Diagrammi . . . 18
2.2.5.7 Qualità per una buona architettura . . . 18
2.2.6 Codifica . . . 19
2.2.6.1 Scopo dell'attività . . . 19
2.2.6.2 Aspettative . . . 19
2.2.6.3 Descrizione dell'attività . . . 19
2.2.6.4 Manuale utente . . . 19
2.2.6.5 Formattazione . . . 19

2.2.6.6 Stile di codifica 20
2.2.6.7 Intestazione 20
2.2.6.8 Versionamento 20
2.2.6.9 Ricorsione 21
2.2.7 Strumenti 21
2.2.7.1 Expressjs 21
2.2.7.2 WebStorm 21
2.2.7.3 ESLlint 22
2.2.7.4 Jasmine 22

3 Processi di supporto **23**
3.1 Processo di documentazione 23
3.1.1 Scopo del processo 23
3.1.2 Aspettative 23
3.1.3 Descrizione del processo 23
3.1.4 Procedura di approvazione dei documenti 23
3.1.5 Template LaTeX 23
3.1.6 Struttura dei documenti 24
3.1.6.1 Costruzione di un documento 24
3.1.6.2 Prima pagina 24
3.1.6.3 Struttura comune delle pagine 24
3.1.6.4 Indice 24
3.1.6.5 Diario delle modifiche 25
3.1.7 Norme tipografiche 25
3.1.7.1 Elenchi puntati 25
3.1.7.2 Comandi personalizzati LaTeX 26
3.1.7.3 Sigle e abbreviazioni 27
3.1.7.4 Stile del testo 28
3.1.7.5 Formati 29
3.1.8 Componenti grafiche 29
3.1.8.1 Tabelle 29
3.1.8.2 Immagini 30
3.1.9 Classificazione dei documenti 30
3.1.9.1 Documenti informali 30
3.1.9.2 Documenti formali 30
3.1.9.3 Verbali 30
3.1.10 Strumenti 31
3.1.10.1 LaTeX 31
3.1.10.2 TexMaker 31
3.1.10.3 Astah 31
3.1.10.4 Lucidchart 31
3.1.10.5 Microsoft Excel 32
3.2 Processo di gestione delle configurazioni 32
3.2.1 Scopo del processo 32
3.2.2 Aspettative 32
3.2.3 Descrizione del processo 32

3.2.4 Versionamento della documentazione . . . 33
3.2.5 Gestione del versionamento . . . 33
3.2.5.1 Repository . . . 33
3.2.5.2 Glossario sintetico termini chiave Git . . . 34
3.2.5.3 Comandi principali . . . 35
3.2.5.4 Struttura del repository . . . 35
3.2.5.5 Tipi di file e .gitignore . . . 36
3.2.5.6 Norme sui commit . . . 36
3.2.6 Gestione dei rilasci e delle consegne . . . 36
3.3 Processo di verifica . . . 36
3.3.1 Scopo del processo . . . 36
3.3.2 Aspettative . . . 36
3.3.3 Descrizione del processo . . . 37
3.3.4 Linee guida per la verifica della documentazione . . . 37
3.3.5 Modifiche al diario modifiche . . . 37
3.3.6 Analisi . . . 38
3.3.6.1 Analisi statica . . . 38
3.3.6.2 Analisi dinamica . . . 38
3.3.7 Test di verifica . . . 39
3.3.7.1 Test di unità[TU] . . . 39
3.3.7.2 Test di integrazione[TI] . . . 39
3.3.7.3 Test di sistema[TS] . . . 39
3.3.7.4 Test di regressione[TR] . . . 40
3.3.7.5 Test di accettazione[TA] . . . 40
3.3.8 Strumenti . . . 40
3.3.8.1 Verifica ortografica . . . 40
3.3.8.2 GitHub Issues . . . 40
3.3.8.3 Travis CI . . . 40
3.3.9 Metriche . . . 41
3.3.10 Metriche per i processi . . . 41
3.3.10.1 Modello SPICE$_G$ . . . 41
3.3.10.2 Ciclo Di Deming . . . 43
3.3.10.3 SV - Schedule Variance$_G$ . . . 44
3.3.10.4 CV - Cost Variance$_G$ . . . 45
3.3.10.5 Process Efficiency (PE) . . . 45
3.3.10.6 Requirements Coverage (RC) . . . 45
3.3.10.7 Number of Parameters (NP) . . . 45
3.3.10.8 Response for Class (RFC) . . . 46
3.3.10.9 Complessità Ciclomatica (CC) . . . 46
3.3.10.10 Density of Comments (DC) . . . 46
3.3.10.11 Weighted Methods per Class (WMC) . . . 46
3.3.10.12 Average Interaction Density of Software Components (AIDC) . . . 46
3.3.10.13 Test Coverage (TC) . . . 47
3.3.11 Metriche per i documenti . . . 47
3.3.11.1 Indice di Gulpease . . . 47

3.3.11.2 Correttezza concettuale 48
3.3.12 Metriche per il prodotto 48
3.3.12.1 Copertura requisiti obbligatori (CRO) 48
3.3.12.2 Copertura requisiti desiderabili (CRD) 48
3.3.12.3 Passed Test Cases Percentage (PTCP) 48
3.3.12.4 Usability metrics for effectiveness 49
3.3.12.5 Tempo medio di risposta (TRISP) 49
3.4 Processo di gestione dei cambiamenti 49
3.4.1 Scopo del processo 49
3.4.2 Aspettative 49
3.4.3 Descrizione del processo 49
3.4.4 Metodi di segnalazione problemi nel codice 50
3.4.4.1 GitHub Issues 50
3.4.5 Metodi di segnalazione problemi universale 50

4 Processi organizzativi **51**
4.1 Processo di gestione di progetto 51
4.1.1 Scopo del processo 51
4.1.2 Aspettative 51
4.1.3 Descrizione del processo 51
4.1.4 Ruoli di progetto 51
4.1.4.1 Amministratore di progetto 52
4.1.4.2 Responsabile di progetto 52
4.1.4.3 Analista 52
4.1.4.4 Progettista 53
4.1.4.5 Verificatore 53
4.1.4.6 Programmatore 53
4.1.5 Gestione delle comunicazioni 53
4.1.5.1 Comunicazioni interne 53
4.1.5.2 Comunicazioni esterne 54
4.1.6 Gestione degli incontri 55
4.1.6.1 Incontri interni 55
4.1.6.2 Incontri esterni 57
4.1.7 Strumenti di coordinamento 58
4.1.7.1 Ticketing 58
4.1.8 Strumenti per il versionamento 60
4.1.8.1 GitHub 60
4.1.8.2 GitHub Desktop 60
4.1.9 Gestione dei rischi 60
4.1.10 Strumenti 61
4.1.10.1 Sistemi operativi 61
4.1.10.2 Slack 61
4.1.10.3 Telegram 62
4.1.10.4 Asana 62
4.1.10.5 GanttProject 62
4.2 Processo di formazione 62

4.2.1 Scopo del processo 62
4.2.2 Aspettative 63
4.2.3 Descrizione del processo 63
4.2.4 Scambio di materiale a fine formativo 63
4.2.5 Strumenti 63
4.2.5.1 W3schools 63

Elenco delle figure

1 Immagine del repository del sito web expressjs.com. 21
2 Immagine esempio TexMaker. 31
3 Immagine esempio Lucidchart. 32
4 Immagine che illustra una delle funzionalità di Travis CI. 41
5 Immagine esplicativa per ISO/IEC 15504. 43
6 Procedura per l'organizzazione di un ritrovo interno. 56
7 Procedura per l'organizzazione di un ritrovo esterno. 57
8 Procedura per l'assegnazione di un ticket. 59
9 GitHub Desktop su Windows. 60
10 Immagine rappresentativa del profilo Slack del team WarMachine. 61
11 Asana visualizzato sul web. 62

1 Introduzione

1.1 Scopo del Documento

Questo documento sancisce le norme di comportamento che verranno adottate dai membri del team$_G$ *WarMachine* nello svolgimento del progetto$_G$ *IronWorks*. All'interno del documento verranno trattati in particolare i seguenti punti:

- Regole d'interazione e comportamento tra membri del team;
- Stesura e revisione di documenti;
- Ambiente di lavoro.

Si è scelto di adottare un modello di sviluppo *Incrementale*. Il contenuto delle norme sarà aggiornato ad ogni incremento, dando priorità, nei primi incrementi, al soddisfacimento dei requisiti importanti sul piano strategico. I requisiti meno importanti avranno più tempo a disposizione per stabilizzarsi con lo stato del sistema.

1.2 Struttura del documento

Il documento è strutturato seguendo i passi del modello standard ISO/IEC 12207. Il documento si compone di un'introduzione seguita da tre sezioni principali:

- Sezione adibita alla trattazione dei processi primari:
  - *Fornitura*;
  - *Sviluppo*.
- Sezione adibita alla trattazione dei processi di supporto:
  - *Gestione della documentazione*;
  - *Gestione della configurazione*;
  - *Verifica*;
  - *Validazione*.
- Sezione adibita alla trattazione dei processi organizzativi:
  - *Gestione di progetto*;
  - *Formazione*.

1.3 Scopo del prodotto$_G$

Con lo sviluppo di questo progetto si intende creare un software capace di:

- Creare e modificare diagrammi di robustezza$_G$;
- auto-generare codice Java$_G$ e SQL$_G$ a partire dal diagramma di robustezza realizzato precedentemente.

1.4 Glossario

Per evitare ogni sorta di ambiguità relativa al linguaggio utilizzato è stato creato un *Glossario* contenente le parole ambigue e di difficile comprensione. Queste parole sono marcate con una G nel pedice$_G$.

1.5 Riferimenti

1.5.1 Normativi

- **ISO/IEC 12207**: https://it.wikipedia.org/wiki/Iso_12207;
- **ISO/IEC 15504**: https://en.wikipedia.org/wiki/ISO/IEC_15504;
- **Capitolato$_G$ trattato**: http://www.math.unipd.it/~tullio/IS-1/2017/Progetto/C5.pdf.

1.5.2 Informativi

- **Guida fornita dal corso di LaTeX2016-2017 tenuto dal prof. Bresolin** : https://github.com/R-and-LaTeX/GuidaGalatticaPerLaTeX/releases/;
- **Software Engineering - Ian Sommerville - 9th Edition** : in particolare il capitolo 4 - Requirements engineering;
- **Slide del corso**: http://www.math.unipd.it/~tullio/IS-1/2017/;
- **Use-Case$_G$ 2.0 The Guide to Succeeding with Use Cases - Ivar Jacobson**;
- **Use Case Driven Object modeling - Doug Rosenberg, Matt Stephens**.

2 Processi primari

2.1 Processo$_G$ di fornitura

2.1.1 Scopo del processo

Lo scopo del processo di fornitura è quello di stabilire un accordo tra proponente$_G$ e fornitore, in cui quest'ultimo si impegna a consegnare un prodotto che soddisfa i requisiti concordati.

2.1.2 Aspettative

Il fine di questo processo è lo stabilire un accordo contrattuale con il proponente, *Zucchetti s.p.a*, che definisca le modalità di esecuzione del contratto e la consegna di un prodotto finito.

2.1.3 Descrizione del processo

Facendo riferimento allo standard Iso/IEC 12207:1995 il fornitore deve svolgere le seguenti attività:

- Accordo Contrattuale;
- Studio di fattibilità;
- Modalità di esecuzione del contratto.

2.1.4 Accordo contrattuale

Il fornitore deve accordarsi con il proponente, *Zucchetti s.p.a*, per definire e contrattare le richieste presenti nel documento di presentazione del capitolato. La manutenzione del suddetto prodotto non rientra nei compiti del fornitore.

2.1.5 Modalità di esecuzione del contratto

Il fornitore si impegna a collaborare con il proponente *Zucchetti s.p.a* per l'intera durata del progetto per raggiungere i seguenti obbiettivi:

- Concordare e chiarire i vincoli sui requisiti;
- Concordare e chiarire i vincoli di progetto.

Il fornitore si impegna a fornire al proponente, *Zucchetti s.p.a*, e al committente$_G$, *Prof. Vardanega Tullio* e *Prof. Cardin Riccardo* , la seguente documentazione:

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

2.1.6 Consegna e manutenzione del prodotto

Il prodotto finito candidato all'accettazione dovrà essere consegnato su supporto CD-ROM / DVD, comprensivo di:

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

Il fornitore, *WarMachine*, come stabilito nell'accordo contrattuale non si occuperà della manutenzione del prodotto.

2.1.7 Studio di Fattibilità

È il primo documento da redigere per la revisione dei requisiti.
Il *Responsabile di progetto* deve organizzare delle riunioni con lo scopo di valutare i capitolati. Dalle riunioni deve emergere la stesura dello *Studio di fattibilità*, basato su:

- **Dominio tecnologico e applicativo**: Conoscenza delle tecnologie richieste dal capitolato;
- **Rapporto costi/benefici**: Prodotti simili già presenti sul mercato, quantità di requisiti obbligatori, costi in rapporto ai risultati stimati;
- **Individuazione dei rischi**: Comprensione dei punti critici della realizzazione quali mancanza di conoscenze o competenze e difficoltà nell'individuazione dei requisiti.

2.2 Processo di sviluppo

2.2.1 Scopo del processo

Il processo ha lo scopo di sviluppare un prodotto software che rispecchi la volontà del proponente. Di seguito verranno trattate le attività e i compiti svolti dal Team.

2.2.2 Aspettative

Le aspettative per la corretta esecuzione di questo processo sono:

- Realizzare un prodotto finale che sia in grado di soddisfare sia i test di validazione e verifica sia i requisiti concordati con il proponente;
- Fissare gli obbiettivi per lo sviluppo;
- Fissare i vincoli tecnologici.

2.2.3 Descrizione del processo

Il processo di sviluppo, in accordo con lo standard ISO/IEC 12207, è strutturato nel seguente modo:

- Analisi dei requisiti;
- Progettazione;
- Codifica.

2.2.4 Analisi dei requisiti

2.2.4.1 Scopo dell'attività

Lo scopo di questa attività è individuare requisiti del progetto descritti nel capitolato presentato e tramite incontri in sede del proponente per concordare requisiti derivati dal dominio tecnologico e applicativo.

2.2.4.2 Aspettative

Il risultato atteso dallo svolgimento di questa attività è la creazione di un documento formale contenente tutti i requisiti concordati e fissati insieme al proponente.

2.2.4.3 Descrizione dell'attività

I requisiti concordati e fissati vanno specificati nel documento *Analisi dei requisiti v2.0.0*. Per analizzare e trovare i requisiti si utilizzano gli use-case. I requisiti vengono tracciati tramite l'applicazione open-source$_{G}$ PragmaDB per garantire il controllo di conformità.

2.2.4.4 Casi d'uso

Ogni caso d'uso viene descritto secondo la seguente struttura:

- Codice identificativo;
- Titolo;
- Diagramma UML$_{G}$;
- Attori primari;
- Attori secondari (se presente);
- Scopo e descrizione;
- Precondizione;
- Postcondizione;
- Scenario principale;
- Scenario alternativo (se presente);
- Estensione (se presente).

2.2.4.5 Codice identificativo

Il codice identificativo descrittivo di ogni caso d'uso è univoco e conforme alla seguente struttura:
UC(Codice primo livello).(Codice secondo livello).(Codice terzo livello).(Codice quarto livello)
I campi sono rispettivamente:
Codice di primo livello: Numero che identifica i casi d'uso da sviluppare;
Codice di secondo livello: Numero progressivo che identifica tutti i possibili sotto-casi del caso d'uso enunciato precedentemente;
Codice di terzo livello: Numero progressivo che identifica tutti i possibili sotto-casi del sotto-caso in questione; **Codice di quarto livello:** Numero progressivo che identifica tutti i possibili sotto-casi del sotto-caso enunciato nel terzo livello.

2.2.4.6 Requisiti

Tutti i requisiti sono descritti secondo la seguente struttura:

- Codice identificativo;
- Tipologia;
- Descrizione;
- Fonti.

2.2.4.7 Codice identificativo

Il codice identificativo descrittivo di ogni requisito è univoco e conforme alla seguente struttura:
R{X}{Y}{Codice primo livello}.{Codice secondo livello}.{Codice terzo livello}.{Codice quarto livello}
I campi sono rispettivamente:

- **X**: Identifica il grado di richiesta del proponente
  - **0**: Requisito obbligatorio;
  - **1**: Requisito desiderabile;
  - **2**: Requisito opzionale.
- **Y**: Identifica il tipo di requisito
  - F: Requisito funzionale;
  - Q: Requisito qualitativo;
  - V: Vincolo progettuale.
- **Codice primo livello**: Numero che identifica il requisito da sviluppare;
- **Codice secondo livello**: Numero progressivo che identifica tutti i possibili sotto-casi del requisito enunciato precedentemente;

- **Codice terzo livello**: Numero progressivo che identifica tutti i possibili sotto-casi del sotto-caso enunciato precedentemente;
- **Codice quarto livello:** Numero progressivo che identifica tutti i possibili sotto-casi del sotto-caso enunciato nel terzo livello.

2.2.4.8 Strumenti utili al tracciamento

Per il tracciamento dei requisiti e dei casi d'uso$_G$ si è utilizzata un'applicazione open-source, l'applicazione consiste di un database relazionale$_G$ a cui è associata un'interfaccia grafica web per facilitarne l'utilizzo. PragmaDB è un applicativo sviluppato da Stefano Munari e Fabio Vedovato. Per maggiori informazioni viene condiviso il link della repository$_G$ GitHub pubblica dove è possibile consultare la documentazione.

https://github.com/StefanoMunari/PragmaDB

2.2.5 Progettazione

2.2.5.1 Scopo dell'attività

L'attività è fondamentale per il buon funzionamento del progetto. Inoltre pone le basi per l'attività di codifica.

2.2.5.2 Aspettative

Il risultato atteso dell'attività di Progettazione consiste di:

- Definizione architettura di sistema;
- Diagrammi delle classi;
- Diagrammi di sequenza;
- Definizione preliminare dei test a cui sottoporre il codice.

2.2.5.3 Descrizione dell'attività

L'attività viene svolta dai *Progettisti*. L'attività divide il sistema in componenti, i componenti vengono divisi a loro volta in unità, identificate come la minore parte di codice verificabile. La progettazione descrive come verranno organizzate e utilizzate queste unità . Inoltre vengono fissati dei test per verificare che le unità siano coerenti e conformi alla logica con cui le unità son state create.

2.2.5.4 Progettazione architetturale

Vengono definite le componenenti del sistema. La progettazione architetturale si articola in:

- Individuazione delle componenti;
- Definire ruoli e responsabilità per ogni componente;
- Definire le interazioni tra le componenti;

- Assicurarsi che ogni requisito sia soddisfatto da almeno una componente e viceversa;
- Creare e documentare i test di integrazione, al fine di verificare il corretto funzionamento di più componenti integrate assieme.

2.2.5.5 Progettazione di dettaglio

Per le componenti individuate durante la progettazione architetturale si procede attraverso i seguenti passi:

- Suddivisione delle componenti trovate in unità;
- Definizione dei ruoli e delle interazioni delle unità;
- Produrre la documentazione alla specifica di ogni unità;
- Specifica dell'unità come insieme di moduli (il modulo è la più piccola entità strutturale utilmente rappresentabile);
- Definizione dei ruoli di ogni modulo;
- Definire gli strumenti per le prove di unità.

2.2.5.6 Diagrammi

Come supporto alla descrizione dell'architettura e al comportamento delle parti di un sistema vengono utilizzati i seguenti diagrammi UML:

- **Diagrammi delle classi**: Descrizione degli oggetti facenti parte di un sistema;
- **Diagrammi di sequenza**: Descrivono la collaborazione di un gruppo di oggetti che devono implementare collettivamente un comportamento.

2.2.5.7 Qualità per una buona architettura

Di seguito sono elencate le qualità richieste per una buona architettura:

- **Sufficienza**: Capacità di soddisfacimento di tutti i requisiti;
- **Comprensibilità**: Facilmente comprensibile dagli stakeholder;
- **Modularità**: Suddivisa in parti chiare e ben distinte;
- **Robustezza**: Capace di gestire diversi input provenienti dall'utente, che siano giusti o sbagliati;
- **Flessibilità**: Permette modifiche a costo contenuto al variare dei requisiti;
- **Riusabilità**: Le sue parti possono essere riutilizzate in altri contesti;
- **Efficienza**: Svolge i compiti per cui l'applicazione è stata pensata e lo fa con il minimo spreco di risorse;
- **Affidabilità**: Svolge in maniera corretta i suoi compiti quando utilizzata;

- **Disponibiltà**: Necessità di tempi molto piccoli per la manutenzione;
- **Sicurezza rispetto a malfunzionamenti**: Resistente a malfunzionamenti gravi perché dotata di un buon sistema di ridondanza per rimanere operativa anche in presenza di malfunzionamenti;
- **Sicurezza rispetto a intrusioni**: I dati e le funzioni non sono vulnerabili a intrusioni esterne;
- **Semplicità**: Ogni parte dell'applicazione contiene solo il necessario;
- **Incapsulazione**: Le parti interne degli elementi non sono visibili dall'esterno;
- **Coesione**: Funzionalità correlate dovrebbero avere lo stesso obiettivo;
- **Basso accoppiamento**: Parti di applicazione distinte dipendono poco o niente le une dalle altre.

2.2.6 Codifica

2.2.6.1 Scopo dell'attività
Lo scopo di questa attività è l'implementazione del prodotto software richiesto. In questa fase di realizzazione del software si concretizza la soluzione attraverso la programmazione di un prodotto.

2.2.6.2 Aspettative
Il risultato atteso dallo svolgimento di questa attività è la creazione di un prodotto software conforme alle richieste prefissate con il *Zucchetti s.p.a.*

2.2.6.3 Descrizione dell'attività
La scrittura del codice dovrà rispettare gli obiettivi di qualità definiti all'interno del *Piano di qualifica v2.0.0* per poter garantire una buona qualità al codice e rispettare i compiti assegnati e gli strumenti indicati nel *Piano di progetto v2.0.0*.

2.2.6.4 Manuale utente
I Programmatori dovranno redigere il Manuale Utente nel quale vengono fornite, a chi utilizzerà il prodotto, le informazioni principali e una guida che fornisce una spiegazione delle funzionalità presenti nell'applicazione.

2.2.6.5 Formattazione
E' richiesto che tutti i membri del gruppo seguano le stesse norme stilistiche di codifica, così che tutto il codice prodotto sia uniforme e di facile comprensione:

- **Indentazione:** Utilizzo di esattamente 4 spazi;
- **Parentesi dei costrutti:** Inserire le parentesi di delimitazione dei costrutti al di sotto di essi e non in linea;

- **Nomi di variabili:** Le variabili vanno scritte in inglese, con nomi brevi e significativi in stile CamelCase$_G$;
- **Nomi di variabili globali:** Le variabili globali vanno scritte con lo stile UPPERCASE;
- **Documentazione codice:** La documentazione del codice va scritta sopra ad ogni metodo e classe. Deve essere aperta nella forma ** e chiusa in *\;
- **Commenti nel codice**: I commenti devono essere usati in parti di codice di difficile comprensione per spiegare, in maniera sintetica, la logica adottata dall'autore del file;
- **Caratteri totali su linea:** Per favorire la leggibilità del codice le linee di codice non devono superare gli 80 caratteri.

2.2.6.6 Stile di codifica

Lo stile di codifica JavaScript adottato è quello sviluppato da Airbnb, A mostly reasonable approach to JavaScript. Uno stile riconosciuto a livello mondiale e che ricopre quasi interamente ogni aspetto del linguaggio.

https://github.com/airbnb/javascript

2.2.6.7 Intestazione

Ogni file di codice dovrà seguire la seguente intestazione:

```
File: Nome file
Version: Versione file
Type: Tipo file
Date: Data di creazione
Author: Nome autore
Registro modifiche: Storico modifiche
```

2.2.6.8 Versionamento

Il versionamento del codice viene inserito nell'intestazione del file e rispetta la seguente struttura:

X.Y

- **X:** Indice di versione principale, un incremento di tale indice rappresenta un avanzamento della versione stabile, che porta il valore dell'indice Y ad essere azzerato;
- **Y:** Indice di modifica parziale, un incremento di tale indice rappresenta una verifica o una modifica rilevante, come per esempio la rimozione o l'aggiunta di una istruzione.

La versione 1.0 deve rappresentare la prima versione del file completo e stabile, cioè quando le sue funzionalità obbligatorie sono state definite e si considerano funzionanti.
Dalle versioni seguenti è possibile testare il file, con gli appositi test definiti nel documento *Piano di qualifica v2.0.0*, per verificarne l'effettivo funzionamento.

2.2.6.9 Ricorsione

E' stato stabilito di evitare la ricorsione ove possibile. Nel caso di utilizzo sarà necessario fornire una prova di terminazione e valutare il costo di occupazione in memoria.

2.2.7 Strumenti

2.2.7.1 Expressjs

Expressjs è un framework per applicazioni web Node.js flessibile e leggero, rilasciato come software libero e open-source con licenza MIT, che fornisce una serie di funzioni avanzate per le applicazioni web e per dispositivi mobili. Esso permette la creazione di un API affidabile attraverso un processo facile e veloce.

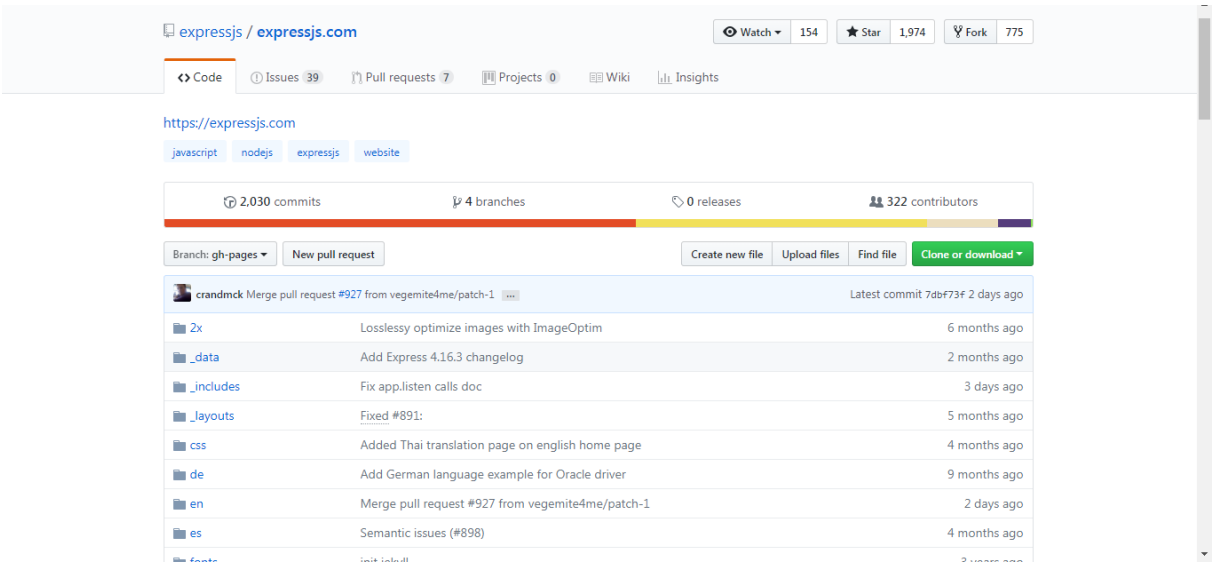

Figura 1: Immagine del repository del sito web `expressjs.com`.

2.2.7.2 WebStorm

L'ambiente di sviluppo (IDE$_G$) adottato è WebStorm$_G$. La scelta del programma è stata facilitata dalla conoscenza pregressa del programma di alcuni componenti del gruppo. Esso presenta le seguenti funzionalità:

- Autocompletamento per la sintassi di codice JavaScript, HTML e CSS;
- Storico locale dei cambiamenti;
- Possiede un terminale proprietario;
- Debugger client-side per JavaScript e Node.js;
- Consente di effettuare test di unità per il linguaggio JavaScript;
- Integrazione con ESLint$_G$;
- Integrazione con Git.

2.2.7.3 ESLlint

Per quanto riguarda l'analisi statica del codice sorgente per scovare errori, bugs, errori di stile, errori di sintassi o costrutti sospetti è stato adottato il software open-source ESLint.

https://eslint.org/

2.2.7.4 Jasmine

Jasmine è un framework di sviluppo behavior-driven$_G$ per fare test su codice JavaScript. Jasmine non richiede un DOM$_G$ e la sua sintassi è di facile comprensione.

https://jasmine.github.io/2.0/introduction.html

3 Processi di supporto

3.1 Processo di documentazione

3.1.1 Scopo del processo

Lo scopo di questo processo consiste nell'illustrare come deve essere stilata e gestita la documentazione durante l'intero ciclo di vita del software.

3.1.2 Aspettative

Le aspettative per il suddetto processo si identificano in:

- Fornire una documentazione chiara che venga mantenuta lungo tutto il ciclo di vita del software;
- Norme chiare e semplici per la corretta stesura dei documenti;
- Documentazione formale e coerente.

3.1.3 Descrizione del processo

Il documento contiene le regole per stilare e manutenere una documentazione corretta e coerente.

3.1.4 Procedura di approvazione dei documenti

Ogni documento non formale deve seguire i seguenti punti:

- Il documento viene stilato dai redattori;
- Il documento viene sottoposto ai *Verificatori* incaricati dal *Responsabile di progetto* per il controllo degli errori;
- Qualora i *Verificatori* trovassero degli errori nella documentazione, devono segnalare i problemi al *Responsabile di progetto* che a sua volta incarica i redattori di apportare le dovute modifiche;
- Se i *Verificatori* ritengono che il documento sia privo di errori, allora il documento viene passato al *Responsabile di progetto* che decide se approvarlo o meno;
- In caso il *Responsabile di progetto* ritenga che il documento non sia ancora in grado di essere approvato, questo deve comunicare una valida motivazione ai redattori con anche le modifiche da apportare al documento.

3.1.5 Template LaTeX

Per facilitare la creazione di documenti è stato creato un template LaTeX. Ogni documento costruito in questo modo sarà dotato di una struttura e uno stile di formattazione comune.

3.1.6 Struttura dei documenti

3.1.6.1 Costruzione di un documento

I documenti, eccetto i verbali, saranno costruiti a partire da un template LaTeX in cui verranno incluse, tramite il comando \include, le sezioni di cui si compone. Le sezioni verranno scritte in file a parte per facilitare il compito dei *Verificatori*.

3.1.6.2 Prima pagina

La prima pagina di ogni documento presenta i seguenti attributi:

- **Logo del gruppo**: Immediatamente visibile come elemento centrato;
- **Titolo del documento**: Immediatamente sottostante al logo;
- **Nome del gruppo e del progetto**;
- **Tabella informativa**: Tabella informativa per la creazione e il versionamento$_G$ del documento, si compone di:
  - *Versione*: Versione aggiornata del documento;
  - *Data Redazione*: Data di creazione del documento;
  - *Redazione*: Cognome e nome dei redattori del documento;
  - *Verifica*: Cognome e nome dei *Verificatori* del documento;
  - *Approvazione*: Cognome e nome del responsabile$_G$ di progetto;
  - *Uso*: Interno o esterno;
  - *Distribuzione*: Gruppo che fornisce il determinato documento.

3.1.6.3 Struttura comune delle pagine

Tutte le pagine, ad eccezioni della prima, devono essere costruite rispettando i vincoli adottati nella creazione del template. In ogni pagina devono essere presenti i seguenti elementi:

- Logo del gruppo in alto a sinistra;
- Numero e titolo della sezione corrente in alto a destra;
- Nome del documento e versione attuale in basso a sinistra;
- Pagina corrente su pagine totali indicate come "Pagina di X su Y" dove X indica la pagina corrente e Y la totalità delle pagine di cui si compone il documento.

3.1.6.4 Indice

Ogni documento formale, verbali esclusi, deve essere dotato di un indice, in modo da agevolare la consultazione e permettere una lettura non solo sequenziale. Qualsiasi indice deve partire da 1, per ciascuna sottosezione deve essere presente un punto di separazione dalla sezione padre. Le appendici sono catalogate tramite una lettera maiuscola che verrà incrementata a partire dalla lettera A seguendo poi l'ordine alfabetico internazionale.

3.1.6.5 Diario delle modifiche

Ogni documento formale deve contenere questo diario. Questo si compone di una tabella che contiene le modifiche apportate al documento stesso indicando per ognuna:

- Versione del documento;
- Data della modifica;
- Nome e cognome delle persone coinvolte nella sua redazione, verifica e approvazione;
- Ruolo delle persone che hanno redatto, verificato o approvato il documento;
- Descrizione breve e precisa della modifica apportata.

Per facilitare la compresione e la verifica delle modifiche si è stabilito di utilizzare i seguenti verbi all'interno della sezione descrizione del diario delle modifiche:

- **Scrivere**: Utilizzato per l'inserimento di sezioni all'interno del documento;
- **Aggiungere**: Utilizzato per l'inserimento di singole sotto-sezioni all'interno del documento;
- **Modificare**: Utilizzato per i cambiamenti effettuati su una parte di testo già inserita;
- **Correggere**: Utilizzato da chi stila il documento per segnalare l'avvenuta correzione degli errori segnalati nel testo;
- **Rivedere**: Utilizzato dai *Verificatori* per segnalare le parti del documento non corrette;
- **Creare**: Utilizzato per evidenziare la data di inizio redazione del documento;
- **Verificare**: Utilizzato dai *Verificatori* per evidenziare le parti del documento che sono state sottoposte al processo di verifica;
- **Approvare**: Utilizzato dal *Responsabile di progetto* per l'approvazione del documento al termine delle verifiche;
- **Riorganizzare**: Utilizzato per lo spostamento, all'interno del documento, di parti del testo;
- **Eliminare**: Utilizzato per la rimozione di sezioni ritenute superflue o sbagliate.

3.1.7 Norme tipografiche

3.1.7.1 Elenchi puntati

Gli elenchi puntati sono rappresentati con un pallino nero nel primo livello. Al secondo livello si usa un trattino e un asterisco nel terzo livello. Se è necessario un'ordine, ad ogni livello, gli elementi degli elenchi puntati devono essere rappresentati tramite numeri, che partendo da uno, si incrementano progressivamente. Ogni elenco puntato esprime un concetto, che viene diviso in più parti per facilitare la comprensione.
Ogni elemento di un elenco puntato deve cominciare con la prima lettera maiuscola e concludersi con un punto e virgola, eccetto l'ultimo che mette fine all'elenco con un punto.

3.1.7.2 Comandi personalizzati LaTeX

Per i riferimenti si utilizzano comandi personalizzati in LaTeX perché garantiscono una corretta scrittura. I comandi sono i seguenti:

- **Ruoli di progetto**:
  - \rdp = *Responsabile di progetto*;
  - \amm = *Amministratore*;
  - \anl = *Analista*;
  - \pro = *Progettista*;
  - \dev = *Programmatore*;
  - \ver = *Verificatore*;
  - \ammi = *Amministratori*;
  - \anli = *Analisti*;
  - \proi = *Progettisti*;
  - \prgi = *Programmatori*;
  - \veri = *Verificatori*.
- **Ruoli di verbale**:
  - \sgr = *Segretario*;
  - \sgri = *Segretari*;
- **Documentazione**:
  - \pdp = *Piano di progetto*;
  - \pdq = *Piano di qualifica*;
  - \np = *Norme di progetto*;
  - \sdf = *Studio di fattibilità*;
  - \adr = *Analisi dei requisiti*;
  - \glo = *Glossario*;
  - \mdu = *Manuale utente*.
- **Documentazione compresa di versione**:
  - \pdpv = *Piano di progetto v2.0.0*;
  - \pdqv = *Piano di qualifica v2.0.0*;
  - \npv = *Norme di progetto v2.0.0*;
  - \adrv = *Analisi dei requisiti v2.0.0*;
  - \sdfv = *Studio di fattibilità v1.0.0*;
  - \glov = *Glossario v2.0.0*.

- **Revisioni**:
  - \revisionedeirequisiti = **Revisione dei Requisiti**;
  - \revisionediprogettazione = **Revisione di Progettazione**;
  - \revisionediqualifica = **Revisione di Qualifica**;
  - \revisionediaccettazione = **Revisione di Accettazione**.
- **Nome del team**:
  - \wm = *WarMachine*.
- **Nome del proponente**:
  - \proponente = *Zucchetti s.p.a.*
- **Nome del committente**:
  - \committente = *Prof. Vardanega Tullio* e *Prof. Cardin Riccardo* .
- **Nome di progetto**:
  - \progetto = *IronWorks*.
- **Nuova riga di Diario delle modifiche**: Il comando\diariomodifichenew permette di aggiungere una riga alla tabella del diario modifiche di un documento.

3.1.7.3 Sigle e abbreviazioni

Nella documentazione si utilizzano sigle e abbreviazioni per l'uso di tabelle o altre strutture con spazio limitato. Le sigle sono le seguenti:

- Per i nominativi della documentazione:
  - **RR**: Revisione dei requisiti;
  - **RP**: Revisione di progettazione;
  - **RQ**: Revisione di qualifica;
  - **RA**: Revisione di accettazione;
  - **NP**: Norme di Progetto;
  - **PP**: Piano di progetto;
  - **PQ**: Piano di qualifica;
  - **AR**: Analisi dei requisiti;
  - **SF**: Studio di fattibilità;
  - **VB**: Verbale;
  - **GL**: Glossario.
- Per i nominativi dei rischi alla sezione analisi dei rischi sul *Piano di progetto v2.0.0*:
  - **ITT**: Inesperienza tecnologica del team;

  - **GHS**: Guasti e problemi hardware e software;
  - **PD**: Perdita dati;
  - **CTS**: Cambiamenti nelle tecnologie scelte;
  - **PC**: Problemi di calendario;
  - **IIM**: Indisponibilità improvvisa dei membri;
  - **ILT**: Inesperienza lavorativa del Team;
  - **ROR**: Rotazione ruoli;
  - **IUS**: Inesperienza del Team nell'uso degli strumenti;
  - **CR**: Comprensione requisiti;
  - **SCA**: Sottovalutazione costi attività.
- Per i nominativi dei ruoli di progetto:
  - **AM**: Amministratore;
  - **RE**: Responsabile di progetto;
  - **AN**: Analista;
  - **PT**: Progettista;
  - **PM**: Programmatore;
  - **VR**: Verificatore.

3.1.7.4 Stile del testo

- **Glossario**: Ogni parola contenuta nel Glossario viene marcata con una G maiuscola nel pedice: per esempio committente. Viene segnalata soltanto la prima occorrenza del termine presente nel glossario;
- **Grassetto**: Viene applicato ai titoli delle sezioni e alle parole chiave che compongono il primo livello di un elenco puntato;
- **Corsivo**: Il corsivo viene utilizzato nei seguenti casi:
  - Secondo livello degli elenchi puntati;
  - Abbreviazioni;
  - Parole facenti parte del glossario;
  - Nomi di documentazione;
  - Nomi di azienda o titoli professionali;
  - Ruoli di progetto;
  - Nomi delle parti coinvolte nel progetto.
- **Monospace**: Viene utilizzato per i nomi dei file;

- **Spaziatura**: Viene utilizzata nei seguenti modi:
  - *Dopo virgole e punti*: Deve essere lasciato uno spazio dopo l'inserimento nel testo, per la terminazione di un periodo, di virgole e punti;
  - *Parentesi, single quotes e double quotes*: Deve essere lasciato uno spazio tra l'apertura di una parentesi, una quote o una double quote e il testo che le precede. Allo stesso modo deve essere lasciato uno spazio tra ka chiusura di una parentesi, di una quote o di una double quote e il testo che ne segue.

3.1.7.5 Formati

I formati utilizzati nella documentazione sono:

- **Orario**: hh:mm
  Dove hh indica le ore e varia tra 0 e 23, mentre mm indica i minuti e varia tra 0 e 59;
- **Data**: gg/mm/yyyy
  Dove gg rappresenta il giorno utilizzando due cifre al massimo, mm rappresenta i mesi utilizzando al massimo due cifre e yyyy rappresenta gli anni utilizzando quattro cifre.
- **Nomi di rilievo**:
  - **Nome di ruolo di progetto**: Viene scritto con la prima lettera maiuscola e in stile corsivo;
  - **Nome delle parti coinvolte nel progetto**: Rientrano in questa categoria fornitore, proponente e committente. Questi nomi sono scritti con la prima lettera maiuscola e in stile corsivo;
  - **Nome di documentazione**: Viene scritto con la prima lettera maiuscola e in stile corsivo;
  - **Nome di file**: Viene scritto senza spaziatura e compreso di ultima versione aggiornata;
  - **Nome proprio di persona**: Viene scritto con la prima lettera di nome e cognome maiuscola, nell'ordine Cognome Nome.

3.1.8 Componenti grafiche

3.1.8.1 Tabelle

Ogni tabella deve essere centrata orizzontalmente e avere, sotto di essa, la propria didascalia associata in modo tale da renderla di più semplice comprensione. Nella didascalia andrà inserito il numero identificativo incrementale per il tracciamento associato a una breve descrizione del contenuto.

3.1.8.2 Immagini
Le immagini devono essere centrate orizzontalmente e avere una larghezza variabile in base alla dimensione originale dell'immagine.
Per agevolare la consultazione della documentazione le immagini verranno separate verticalmente dal testo. L'immagine dovrà essere associata a una didascalia, composta di un numero identificativo utile per il tracciamento all'interno del documento e una breve descrizione di cosa essa rappresenta. I diagrammi UML vengono inseriti nella documentazione come immagini.

3.1.9 Classificazione dei documenti

3.1.9.1 Documenti informali
I documenti che rientrano in questa categoria sono tutti quelli che non sono ancora stati approvati dal *Responsabile di progetto*, sono da considerarsi esclusivamente ad uso interno fino ad approvazione.

3.1.9.2 Documenti formali
I documenti che rientrano in questa categoria sono tutti quelli che hanno ricevuto approvazione dal *Responsabile di progetto*. Questi documenti sono da considerarsi per uso esterno, saranno i documenti condivisi con committente e proponente. Per arrivare a tale stato il documento deve aver già passato la fase di verifica e validazione.

3.1.9.3 Verbali
Il verbale è un documento non formale redatto da un *Segretario* in occasione di incontri con il proponente o ritrovi dei membri del gruppo per discutere problematiche relative al progetto.
Il verbale è un documento non soggetto a versionamento, viene redatto una sola volta e non subisce successive modifiche. Il verbale dovrà essere approvato dal *Responsabile di progetto*. Ogni verbale dovrà indicare nel seguente ordine le sezioni:

1. **"Informazioni generali"**:
    - **Luogo incontro**: Città (Provincia), Via, Sede;
    - **Data incontro**: gg-mm-yyyy;
    - **Ora inizio incontro**: hh-mm formato 24h;
    - **Ora fine incontro**: hh-mm formato 24h;
    - **Argomenti all'ordine del giorno**: Argomenti trattati;
    - **Segretario**: Cognome Nome;
    - **Partecipanti**: Cognome Nome.
2. **"Riassunto dell'incontro"**: contenente le decisioni prese, descritte come elementi di un elenco puntato;
3. **"Riepilogo"**: Contenente una tabella che riprende i codici identificativi e le decisioni relative ad essi.

3.1.10 Strumenti

3.1.10.1 LaTeX

Per la stesura della documentazione è stato scelto di usare il linguaggio LaTeX. La scelta è stata dettata da questioni di praticità, in quanto è utile per separare il contenuto dalla formattazione permettendo un versionamento agevolato e la creazione di template per l'impaginazione condivisa dei documenti. Inoltre il linguaggio permette una comoda gestione automatica degli indici e dei glossari.

3.1.10.2 TexMaker

Per la scrittura del codice LaTeX si è utilizzato TexMaker, di versione minima 4.5. L'applicazione offre un pacchetto con compilatore e visualizzatore PDF integrato e fornisce suggerimenti sul completamento dei comandi LaTeX.

http://www.xm1math.net/texmaker/

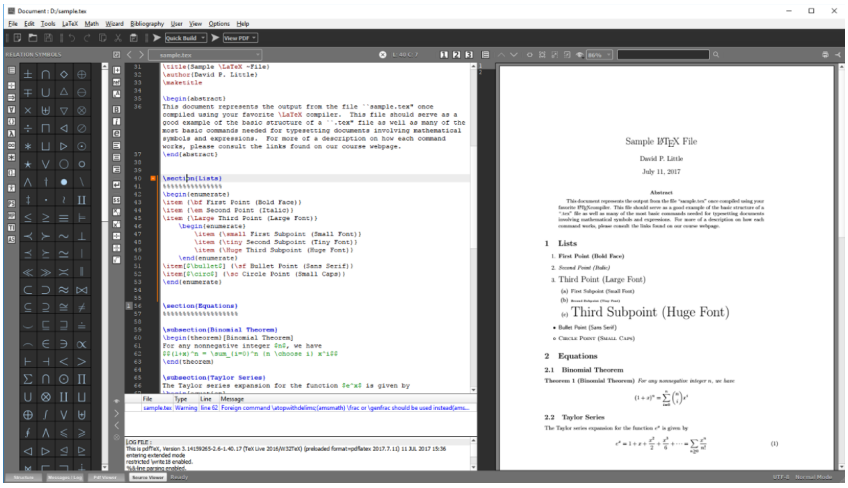

Figura 2: Immagine esempio TexMaker.

3.1.10.3 Astah

Per la realizzazione dei diagrammi degli Use-Case ricavati nel documento *Analisi dei requisiti v2.0.0* si è scelto di utilizzare Astah. Astah è un'applicazione che fornisce un'interfaccia utente che comprende un editor$_G$ per diagrammi UML e dei tool da utilizzare per la modellazione del diagramma. L'applicazione consiste in una versione gratuita e di una versione premium. Si è deciso di utilizzare la versione premium poiché offriva più possibilità per la modellazione.

http://astah.net/

3.1.10.4 Lucidchart

Per la realizzazione dei diagrammi a scopo illustrativo utilizzati nella documentazione si è utilizzato Lucidchart, un'applicazione web$_G$ che permette di realizzare diversi tipi di diagrammi.

https://www.lucidchart.com

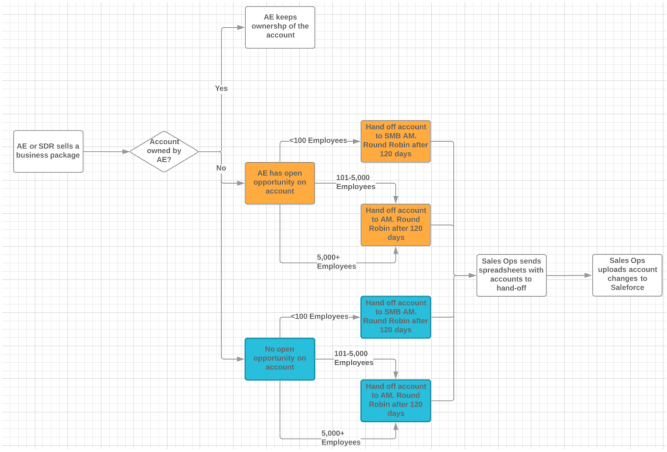

Figura 3: Immagine esempio Lucidchart.

3.1.10.5 Microsoft Excel

Per calcolare il preventivo da proporre, la realizzazione di grafici e tabelle per la gestione del tempo rendicontato e dell'impatto percentuale dei costi rispetto ai vari ruoli è stato usato Microsoft Excel.

3.2 Processo di gestione delle configurazioni

3.2.1 Scopo del processo

Lo scopo del processo è la gestione delle configurazioni utilizzate dal Team nello sviluppo del progetto.

3.2.2 Aspettative

Dall'instanziazione di questo processo il Team si aspetta di poter gestire, in maniera organizzata e sistematica, gli oggetti software che sono stati creati e devono essere utilizzati.

3.2.3 Descrizione del processo

Il processo consiste nell'applicare procedure tecniche e amministrative lungo tutto il ciclo di vita del software per:

- Identificare, definire e assegnare baseline ad oggetti software in un sistema;
- Modifiche di controllo e rilasci degli oggetti software;
- Registrare e segnalare lo stato degli oggetti e delle richieste di modifica;
- Assicurare completezza, consistenza e correttezza degli oggetti;
- Assicurare il controllo, il magazzinamento, lo sfruttamento e la consegna degli oggetti software.

3.2.4 Versionamento della documentazione

Ciascun documento deve essere soggetto a versionamento, eccetto i verbali, perciò chiunque lo utilizzi avrà una visione chiara e dettagliata della sua storia e delle sue modifiche. Ad ogni versione deve corrispondere una riga nel diario delle modifiche. Verrà applicato il seguente formalismo:

vX.Y.Z

dove:

- **v**: Versione del documento;
- **X**:
  - Inizia da 0;
  - Viene incrementato quando il Responsabile di Progetto approva il documento;
  - Limitato superiormente dal numero di revisioni.
- **Y**:
  - Inizia da 0;
  - Viene incrementato da parte del Verificatore ad ogni verifica;
  - Non è limitato superiormente;
  - Quando viene incrementato X, Y e Z vengono riportati a 0.
- **Z**:
  - Inizia da 0;
  - Viene incrementato da parte del Redattore del documento ad ogni modifica;
  - Non è limitato superiormente;
  - Quando viene incrementato Y, viene riportato a 0.

3.2.5 Gestione del versionamento

3.2.5.1 Repository

Per il versionamento e il salvataggio dei file prodotti durante il progetto è previsto l'utilizzo del repository creato su GitHub. L'*Amministratore* di Progetto si occupa di creare tali repository con un account generale, rappresentante il gruppo di progetto chiamato "WarMachineSwe". Successivamente, tutti gli altri membri del gruppo dovranno creare un account personale. L'*Amministratore* fornirà agli account creati i permessi di: scrittura, lettura ed eliminazione.

3.2.5.2 Glossario sintetico termini chiave Git

In seguito è riportata una breve introduzione ai termini fondamentali per l'uso di Git all'interno del progetto.

- **Working tree:** Un albero dei file già verficati. Il working tree contiene il contenuto dell'albero HEAD dei commit più qualsiasi modifica locale di cui ancora non è stato fatto commit (verbo).
- **Parents:** Lista di possibili predecessori di un certo commit;
- **Branch$_G$:** Ogni membro ha a sua disposizione un branch, oltre a quello di default (master). La divisione in branch permette ai membri di lavorarare su requisiti diversi senza la possibilità di generare conflitti sui file causati da altri membri del Team. L'ultimo commit effettuato su un determinato branch viene considerato come la cima di quel branch;
- **Index$_G$:** Una collezione di file con informazioni sullo stato del progetto, il cui contenuto è salvato come un oggetto. L'index è una versione salvata del working tree del progetto. Può contenere anche le versioni di altri working tree utili per possibili merge;
- **HEAD:** Identificato come il branch corrente. HEAD è un albero contenente i riferimenti agli heads nella repository utilizzata;
- **head:** Un riferimento nominale al commit che sta in cima a un determinato branch;
- **Pull requests:** Quando un requisito software è ultimato, viene incorporato sul master branch dal branch corrente attraverso una pull requests. Nell'apertura di una pull request, vengono visualizzate le differenze nei contenuti di entrambi i branch.
- **Commit:**
  - *Come nome:* Un singolo punto nello storico Git; l'intera storia del progetto è rappresentata da un insieme di commit indipendenti l'uno dall'altro. Ogni commit deve avere un messaggio associato che consiste di una descrizione che spiega il perchè una particolare modifica è stata fatta. I commit sono identificati univocamente dal sistema di versionamento Git, semplificando il tracciamento delle modifiche effettuate;
  - *Come verbo:* L'azione per salvare un nuovo stato di avanzamento del progetto nello storico Git, effettuata creando un nuovo commit rappresentante lo stato corrente dell'index e facendo avanzare l'HEAD facendolo puntare al nuovo commit.
- **Merge$_G$:**
  - *Come verbo:* L'azione usata per trasferire il contenuto di un altro branch sul branch corrente;
  - *Come nome:* Identificato come la creazione di un commit rappresentante il risultato dell'azione merge, che ha come parents le cime dei branch su cui viene fatto merge.

3.2.5.3 Comandi principali

- **git clone [url]:** Esegue la creazione della copia locale del repository;
- **git pull:** Aggiorna sovrascrivendo il repository locale con quello remoto;
- **git push:** Aggiorna sovrascrivendo il repository remoto con quello locale;
- **git status:** Mostra lo stato attuale del repository locale con i file modificati;
- **git add nomefile:** Aggiunge un nuovo file alla lista di quelli tracciati da Git;
- **git rm nomefile:** Rimuove un file dal repository;
- **git commit -m "Commento Modifica":** : Fa il commit delle modifiche effettuate salvandole sul repository locale sui file desiderati;
- **git branch:** Visualizza tutti i branch presenti nel repository locale;
- **git checkout:** Permette di cambiare il branch corrente nel repository locale;
- **git fetch:** Scarica la versione aggiornata dal repository senza eseguire il merge;
- **git merge:** Effettua l'unione del repository remoto scaricato con il comando fetch con quello locale.

3.2.5.4 Struttura del repository

La struttura del repository è composta da quattro cartelle, una per ogni tipo di revisione (RR, RQ, RP e RA). A loro volta, queste si sviluppano nelle cartelle seguenti:

- **interni**:
  - *Norme di progetto v2.0.0*;
  - *Studio di fattibilità*;
  - *Verbali*: Cartella per i verbali a scopo interno.
- **esterni**:
  - *Glossario v2.0.0*;
  - *Analisi dei requisiti v2.0.0*;
  - *Piano di progetto v2.0.0*;
  - *Piano di qualifica v2.0.0*;
  - *Verbali*: Cartella per i verbali a scopo esterno.
- **templates**: Contiene i file del template e il layout dei documenti LaTeX;
- **scripts**: Contiene gli scripts scritti nel linguaggio di programmazione Python creati per automatizzare determinati compiti.

3.2.5.5 Tipi di file e .gitignore

Nelle cartelle contenenti tutti i documenti saranno presenti solamente i file .tex, .jpg, .png. Le estensioni dei file generati automaticamente dalla compilazione sono stati aggiunti al file .gitignore nel repository, quindi vengono ignorati e resi invisibili a Git_{G}.

3.2.5.6 Norme sui commit

Ogni qual volta che vengono effettuate delle modifiche ai file del repository, e successivamente devono essere caricate su di esso, bisogna specificarne le motivazioni nel modo più chiaro possibile. Questo avviene utilizzando il comando "commit" accompagnato da un messaggio riassuntivo e una descrizione in cui va specificato:

- La lista dei file coinvolti;
- La liste delle modifiche effettuate, ordinate per ogni singolo file.

Prima di eseguire tale procedura, va aggiornato il diario delle modifiche, secondo le regole viste nella §[3.1.6.5].

3.2.6 Gestione dei rilasci e delle consegne

I rilasci e la consegna dei prodotti software e della documentazione allegata devono essere sottoposti a rigido controllo. La copia principale costantemente aggiornata della documentazione verrà mantenuta all'interno di un repository privato su GitHub mentre al *Prof. Vardanega Tullio* e *Prof. Cardin Riccardo* , prima di ogni revisione, verrà comunicato tramite email, secondo le tempistiche stabilite dal *Piano di progetto v2.0.0*, il link di un repository pubblico contenente tutta la documentazione tecnica in formato ".pdf". Ad ogni revisione, i documenti saranno accompagnati da una Lettera di Presentazione anch'essa inclusa nel repository.

3.3 Processo di verifica

3.3.1 Scopo del processo

Lo scopo di questo processo è quello di accertare che l'esecuzione di specifiche attività non abbia introdotto errori.

3.3.2 Aspettative

Dall'esecuzione di questo processo è possibile individuare:

- Una o più procedure di verifica;
- Regole per verificare la correttezza del prodotto;
- Imprecisioni, errori e imperfezioni del prodotto, i quali vengono annottati per essere corretti.

3.3.3 Descrizione del processo

Il processo è diviso in due attività:

- **Analisi**: Questa attività viene effettuata in due modi, l'analisi statica e l'analisi dinamica.
- **Test di verifica**: Consiste in test definiti dal team *WarMachine* per verificare la corretta esecuzione del prodotto. Dal momento che ancora non si sono trattate le tecnologie richieste questa parte sarà posticipata al completamento della revisione dei requisiti.

3.3.4 Linee guida per la verifica della documentazione

Per la verifica dei documenti sono fornite le seguenti linee guida che i *Verificatori* devono seguire per garantire un corretto processo di verifica. Le linee guida sono:

- Verificare che siano presenti tutte le sezioni indicate nell'indice del documento;
- Verificare la grammatica utilizzata e la sintassi del testo del documento;
- Verificare la grammatica nel testo associato alle figure e alle tabelle (captions, testo interno);
- Verificare che il testo del documento non contenga termini qualitativi ambigui. Es. Abbiamo svolto un'analisi approfondita, "APPROFONDITA" sarebbe meglio non usarlo a meno che non sia stata definita una misura precisa;
- Verificare che le frasi siano di senso compiuto;
- Verificare che i periodi non risultino inutilmente lunghi;
- Verificare che il soggetto e l'oggetto dell'azione siano sempre indicati;
- Verificare che i termini ambigui o di difficile comprensione siano trattati nel glossario.

3.3.5 Modifiche al diario modifiche

I *Verificatori* quando avranno finito il processo di verifica registreranno i risultati sul diario delle modifiche, considerando sia le parti verificate con successo sia le parti che hanno bisogno di essere riviste. Per quanto riguarda le parti verificate correttamente, per facilitare la scrittura e la comprensione, si scrivono solo le sezioni verificate segnalando gli errori nella maniera più precisa possibile. Le segnalazioni devono essere scritte nel seguente modo:

Da rivedere [W].[X].[Y].[Z]

Dove ogni lettera, rispettivamente, indica un diverso livello di annidamento:

- *W*: La sezione di riferimento;
- *X*: La sotto-sezione di riferimento;

- *Y*: La sotto-sotto-sezione di riferimento;
- *Z*: Il paragrafo di riferimento.

Qualora si presentino più errori nello stesso livello di annidamento si deve segnalare l'intero livello.

3.3.6 Analisi

3.3.6.1 Analisi statica

L'analisi statica è un metodo di verifica del codice sorgente e della documentazione. Non richiede l'esecuzione di parti del sistema software e viene eseguita tramite una delle seguenti tecniche:

- **Walkthrough**: Si tratta di una lettura a largo spettro. Il suo scopo è di esaminare l'intero documento alla ricerca di errori generici. Non è una tecnica di analisi molto pratica, viene usata principalmente nelle prime fasi$_G$ di progetto quando il gruppo non ha ancora buona padronanza o conoscenza delle norme e del *Piano di qualifica v2.0.0*. A supporto di questa tecnica si deve creare una lista di controllo dove vengono elencati gli errori più comuni.
- **Inspection**: Si tratta di una lettura mirata e strutturata, si applica quando si conoscono le problematiche per cui è richiesta la verifica. Questa tecnica viene eseguita da *Verificatori* scelti e distinti dai programmatori che, durante la lettura, cercheranno di individuare gli errori segnalati nella lista di controllo precedentemente stilata. Con l'avanzare nel progetto la lista di controllo viene estesa rendendo questa tecnica ancor più efficace.

3.3.6.2 Analisi dinamica

E' una tecnica di analisi del prodotto software, comporta l'esecuzione del programma. La tecnica viene applicata tramite dei test che hanno la funzione di verificare il corretto funzionamento del prodotto oppure di far emergere eventuali errori in esso. I test sono procedure costose, richiedono tempo, persone e infrastrutture. Un singolo test non basta, i risultati valgono solo per quella determinata esecuzione ma non valgono più a valle di una nuova modifica. I test devono essere ripetibili cioè deve essere possibile, dato lo stesso input e svolto nello stesso ambiente, ottenere lo stesso output. I test sono caratterizzati come segue:

- **Ambiente**: Rappresenta il sistema hardware e software dove viene fatto eseguire il test del prodotto;
- **Stato iniziale**: Rappresenta il punto iniziale da cui far eseguire il test;
- **Ingresso**: Rappresenta l'input inserito;
- **Uscita**: Rappresenta l'output ottenuto;
- **Oggetto della prova**: Rappresenta l'oggetto che viene sottoposto al test di verifica;

3.3.7 Test di verifica

3.3.7.1 Test di unità[TU]

Il test consiste nell'isolare singole unità di software per testare se la parte isolata funziona come previsto. Per unità si intende la parte di software più piccola verificabile. A seconda del paradigma di programmazione adottato può identificarsi con una funzione o una classe, rispettivamente se il paradigma è procedurale o ad oggetti. Le unità vengono sottoposte al test prima di essere integrate in moduli ed essere sottoposte al test delle interfacce tra i vari moduli. Per i test di unità sono richieste le componenti stub_{G} e driver_{G}. I driver simulano un'unità chiamante mentre lo stub simula l'unità chiamata.

3.3.7.2 Test di integrazione[TI]

Il test consiste nella combinazione di unità, le quali devono essere già state sottoposte al test di unità, per formare un unico componente e nel test delle interfacce effettuato successivamente per valutare se l'interazione tra le parti combinate presenta problemi. Le strategie di test di integrazione possono essere di due tipi:

- **Bottom-up**: Prima vengono sviluppate le parti con minori dipendenze funzionali e di maggiore utilità. Successivamente si prosegue risalendo l'albero delle dipendenze. Questa strategia riduce il numero di stub ma non rende immediatamente disponibili funzionalità di alto livello;
- **Top-down**: Vengono prima sviluppate le parti più esterne e successivamente si scende lungo l'albero delle dipendenze. Questa strategia aumenta il numero di stub ma rende presto disponibili funzionalità di alto livello.

3.3.7.3 Test di sistema[TS]

Il test di sistema viene svolto dopo il test di integrazione ed è effettuato sul prodotto software finale, ovvero quando lo si ritiene pronto per la consegna. Il test consiste nella verifica del soddisfacimento dei requisiti da parte del prodotto software.
Il codice identificativo di ogni test di sistema è univoco e conforme alla seguente struttura: **TS{X}{Y}{Z}**.
I campi sono rispettivamente:

- **TS**: Test di sistema;
- **X**: Identifica il grado di richiesta del proponente;
  - **0**: Requisito obbligatorio;
  - **1**: Requisito desiderabile;
  - **2**: Requisito opzionale.
- **Y**: Identifica il tipo di requisito;
  - **F**: Requisito funzionale;
  - **Q**: Requisito qualitativo;
  - **V**: Requisito di vincolo.
- **Z**: Id requisito.

3.3.7.4 Test di regressione[TR]
Il test va eseguito quando vengono modificate parti del sistema, la sua funzione è accertare che modifiche effettuate per correzione o estensione non comportino errori nel sistema. Per verificare che non siano emersi problemi dalle parti verificate si devono eseguire nuovamente i test TU e TI sulla parte di codice che ha ricevuto modifiche.

3.3.7.5 Test di accettazione[TA]
Il test consiste nel collaudo del prodotto in presenza del proponente. Una volta superato il test si prosegue con il rilascio ufficiale del prodotto.

3.3.8 Strumenti

3.3.8.1 Verifica ortografica
Viene utilizzata la verifica dell'ortografia in tempo reale, strumento integrato in TexMaker che sottolinea in rosso le parole errate secondo il dizionario della lingua italiana.

3.3.8.2 GitHub Issues
Per l'assegnazione e il tracciamento degli errori viene utilizzato lo strumento issues di GitHub con linguaggio markdown$_G$. Gli issue sono composti da elenchi puntati i cui elementi sono righe di codice che necessitano di essere riviste. Per ogni issue verrà il *Responsabile di progetto* assegna un membro del gruppo che si occupa della sua risoluzione.

https://github.com/WarMachineSwe/WarMachine/issues

3.3.8.3 Travis CI
Travis CI è un servizio distribuito di integrazione continua$_G$ che permette di eseguire script$_G$ e test automatizzati$_G$ quando avvengono modifiche nella repository.
Ad ogni push$_G$, Travis CI esegue gli scripts specificati e i test automatizzati, il tutto in un ambiente di sviluppo vergine$_G$ garantendo l'effettivo successo della build$_G$ del software. Tramite uno script, Travis, calcola l'indice di gulpease$_G$. Inoltre, deve essere usato anche come strumento di correttezza per il glossario. L'applicazione permette infatti le seguenti verifiche:

- Segnala come errore i termini, trovati nel testo, marcati col comando \gl ma non presenti all'interno del glossario;
- Segnala come errore i termini, trovati nel glossario, che non sono presenti in nessuna sezione del testo.

https://travis-ci.org/WarMachineSwe/WarMachine

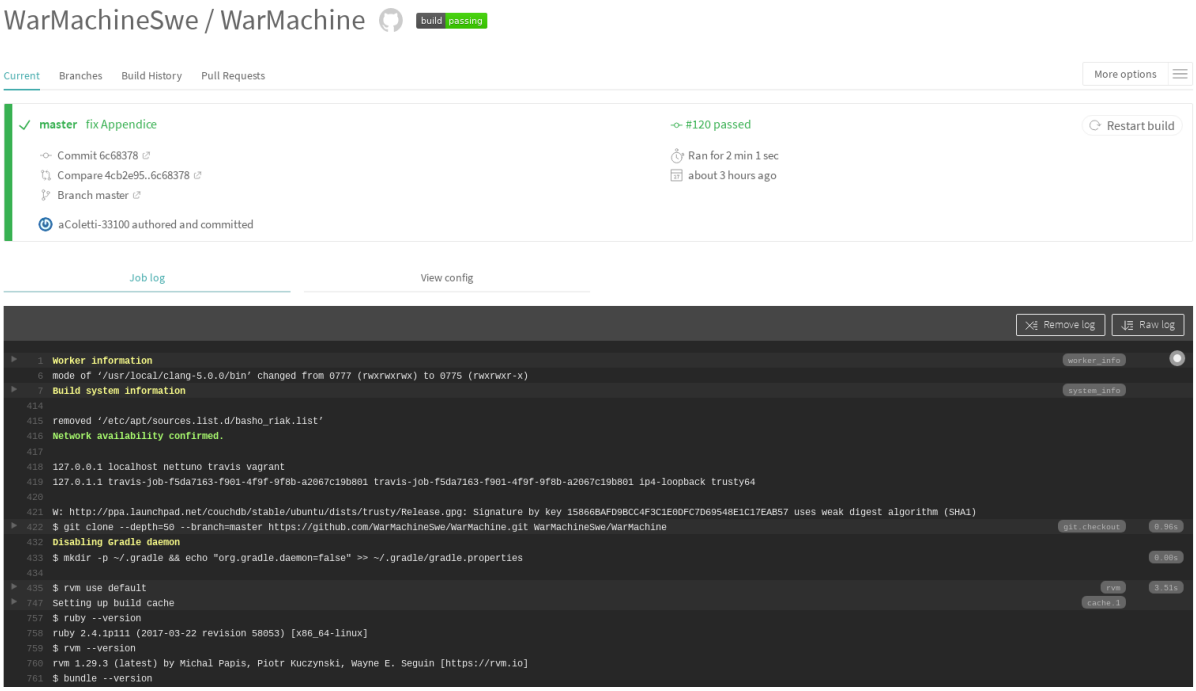

Figura 4: Immagine che illustra una delle funzionalità di Travis CI.

3.3.9 Metriche

Il processo di verifica, per essere informativo, dovrà innanzitutto essere quantificabile.
I dati ottenuti attraverso il processo di verifica dovranno basarsi su metriche stabilite a priori e qualora queste fossero ambigue e approssimate si miglioreranno attraverso il ciclo di vita, in maniera incrementale. Data la natura variabile di tali metriche si sono individuati due range:

- **Accettazione:** Valore richiesto per accettare il prodotto;
- **Ottimale:** Valore entro cui dovrebbe inserirsi la misurazione. Non è vincolante, ma fortemente consigliato e uno scostamento da questo necessita di ulteriori verifiche.

3.3.10 Metriche per i processi

Queste metriche utilizzano indici che ne analizzano l'andamento di variabili critiche come costi e tempi. Tali indici forniscono informazioni che consentono un riscontro immediato sullo stato attuale del progetto; ad ogni incremento questi verranno valutati e, in base ai risultati, il responsabile di progetto sceglierà come procedere.

3.3.10.1 Modello SPICE$_G$

Per gestire e valutare i processi si deve usare il modello SPICE.
Effettuando questo tipo di verifiche il team avrà subito un riscontro della qualità$_G$ del processo. Lo SPICE contiene un modello di riferimento, questo definisce una dimensione di processo e una dimensione di capacità. La dimensione di processo divide i processi nelle seguenti categorie:

- **Customer/Supplier**;
- **Engineering**;
- **Supporting**;
- **Management**;
- **Organization**.

Ad ogni processo viene assegnato un livello della seguente scala:

- 0: processo incompleto;
- 1: processo eseguito;
- 2: processo gestito;
- 3: processo stabilito;
- 4: processo prevedibile;
- 5: processo completo.

La capacità dei processi è misurata tramite i seguenti attributi definiti a livello internazionale:

- 1.1 Process Performance;
- 2.1 Performance Management;
- 2.2 Work Product Management;
- 3.1 Process Definition;
- 3.2 Process Deployment;
- 4.1 Process Measurement;
- 4.2 Process Control;
- 5.1 Process Innovation;
- 5.2 Process Optimization.

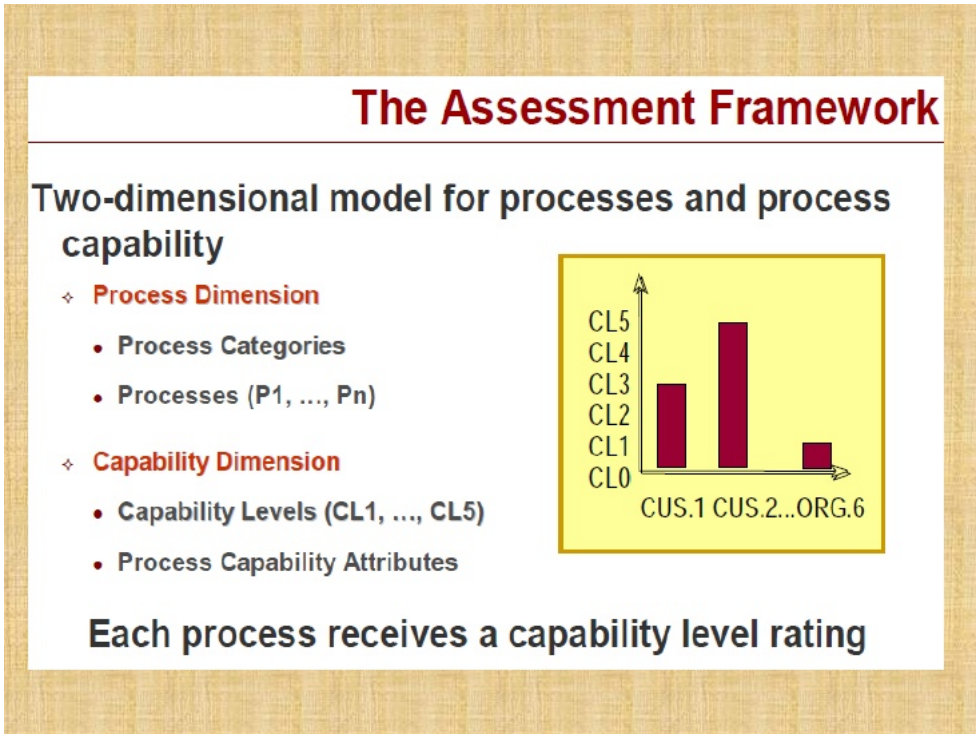

Figura 5: Immagine esplicativa per ISO/IEC 15504.

Ciascun attributo di processo consiste di una o più pratiche generiche che a loro volta sono elaborate in "Indicatori della pratica" che aiutano nella fase di valutazione delle prestazioni.
Ciascun attributo del processo è valutato secondo una scala a quattro valori (N-P-L-F):

- **N**ot achieved (0 - 15%);
- **P**artially achieved (>15% - 50%);
- **L**argely achieved (>50% - 85%);
- **F**ully achieved (>85% - 100%).

La valutazione è fatta sulla base di evidenze oggettive acquisite a fronte di ciascun indicatore di processo durante la fase di assessment che dimostrano l'aderenza agli attributi dei processi.

3.3.10.2 Ciclo Di Deming

Per il miglioramento continuo dei processi e dei prodotto si è adottato il ciclo di Deming. Questo metodo è composto delle seguenti quattro fasi:

- **Plan**: Stabilire gli obiettivi e i processi necessari per fornire risultati in accordo con i risultati attesi, attraverso la creazione di attese di produzione, di completezza e accuratezza delle specifiche scelte. Quando possibile, avvio su piccola scala, per verificare i possibili effetti;

- **Do**: Esecuzione del programma, dapprima in contesti circoscritti. Attuare il piano, eseguire il processo, creare il prodotto. Raccogliere i dati per la creazione di grafici e analisi da destinare alla fase di "Check" e "Act";
- **Check**: Test e controllo, studio e raccolta dei risultati e dei riscontri. Studiare i risultati, misurati e raccolti nella fase del "Do" confrontandoli con i risultati attesi, obiettivi del "Plan", per verificarne le eventuali differenze. Cercare le deviazioni nell'attuazione del piano e focalizzarsi sulla sua adeguatezza e completezza per consentirne l'esecuzione. I grafici dei dati possono rendere questo molto più facile, in quanto è possibile vedere le tendenze di più cicli PDCA, convertendo i dati raccolti in informazioni. L'informazione è utile per realizzare il passo successivo : "Act";
- **Act**: Azione per rendere definitivo e/o migliorare il processo (estendere quanto testato dapprima in contesti circoscritti all'intera organizzazione). Richiede azioni correttive sulle differenze significative tra i risultati effettivi e previsti. Analizza le differenze per determinarne le cause e dove applicare le modifiche per ottenere il miglioramento del processo o del prodotto. Quando un procedimento, attraverso questi quattro passaggi, non comporta la necessità di migliorare la portata a cui è applicato, il ciclo PDCA può essere raffinato per pianificare e migliorare con maggiore dettaglio la successiva iterazione, oppure l'attenzione deve essere posta in una diversa fase del processo.

3.3.10.3 SV - Schedule Variance$_G$

La Schedule Variance indica l'efficacia dei processi valutando l'andamento temporale delle attività che li compongono: vale a dire se la conclusione delle attività è in regola, in ritardo o concordante con i tempi prefissati nel documento di *Piano di progetto v2.0.0*.
Valori utilizzati nella formula:

- **DCR**: Valore (in giorni) delle attività realizzate alla data corrente;
- **DCP**: Costo pianificato (in giorni) per realizzare le attività di progetto alla data corrente.

$$SV = DCR - DCP \tag{1}$$

I risultati ottenibili sono:

- **Valore non accettabile**: Un valore maggiore o uguale al 10% del costo totale prefissato;
- **Valore accettabile**: Un valore minore del 10% del costo totale prefissato;
- **Valore ottimale**: Un valore pari allo 0%, indicatore di una pianificazione accurata e ben gestita.

3.3.10.4 CV - Cost Variance$_G$

E' un indicatore di efficienza. Il costo reale si deve confrontare rispetto al costo prefissato nel documento *Piano di progetto v2.0.0*. In caso il costo reale fosse maggiore del costo prefissato questo indica che sono state richieste risorse maggiori rispetto a quanto previsto precedentemente.
Valori utilizzati nella formula:

- **CR** :Valore (in €) delle attività realizzate alla data corrente;
- **CP** : Costo pianificato (in €) per realizzare le attività di progetto alla data corrente.

$$CV = CR - CP \tag{2}$$

I risultati ottenibili sono:

- **Valore non accettabile**: Un valore maggiore o uguale al 10% del costo totale prefissato;
- **Valore accettabile**: Un valore minore del 10% del costo totale prefissato;
- **Valore ottimale**: Un valore vicino, pari o inferiore allo 0%, indicatore di una pianificazione accurata e ben gestita.

3.3.10.5 Process Efficiency (PE)

Process Efficiency valuta le caratteristiche che influenzano il comportamento di risposta di un'applicazione e l'uso delle risorse. L'efficienza prestazionale è il tempo medio tra l'identificazione di un rischio T_D, dati n rischi, e il tempo in cui ha manifestato suoi effetti T_R.

$$PE = \sum_{i=1,n} \frac{T_{R,i} - T_{D,i}}{n} \tag{3}$$

3.3.10.6 Requirements Coverage (RC)

Requirements Coverage è un indicatore di qualità che indica la percentuale dei requisiti coperti dai test. Per calcolarlo è sufficiente dividere il numero di requisiti coperti con il numero totale di requisiti.

- **RC**: Numero requisiti coperti;
- **RT**: Numero requisiti totali.

$$RC = \frac{RC}{RT} * 100 \tag{4}$$

3.3.10.7 Number of Parameters (NP)

Number of Parameters indica un conteggio del numero ideale di parametri di un metodo. I limiti accettati sono compresi tra 0 e 4. Se la quantità di parametri supera 3 allora i metodi ed i costruttori sono più difficili da comprendere e da usare.

3.3.10.8 Response for Class (RFC)

Response for Class indica il numero di metodi distinti e costruttori invocati da una classe. Questo numero include i metodi nella classe, nella gerarchia di ereditarietà e nei metodi che possono essere richiamati su altri oggetti.
Se il valore di RFC supera 100 significa che c'è una complessità elevata e puo' essere difficile testare il comportamento della classe poiché richiede una profonda comprensione delle potenziali interazioni che gli oggetti della classe possono avere con il resto del sistema. I valori ottimali cadono nell'intervallo [0,50) con valori accettabili fino al 100.

3.3.10.9 Complessità Ciclomatica (CC)

La complessità ciclomatica viene usata per stimare la complessità logica (funzioni, moduli, metodi o classi) del software. Essa misura il numero di percorsi linearmente indipendenti che attraversano il grafo di flusso di controllo ed è espressa nel modo seguente:

$$CC = archi - nodi + 1 \tag{5}$$

dove i nodi rappresentano i gruppi indivisibili di istruzioni mentre un arco connette due nodi se le istruzioni di uno dei nodi possono essere eseguite direttamente dopo l'esecuzione delle istruzioni dell'altro nodo.
Quando la complessità ciclomatica è bassa indica in genere un metodo di semplice comprensione, test e gestione mentre se è maggiore di 25, porta ad un'eccessiva complessità del codice e difficile manutenzione.

3.3.10.10 Density of Comments (DC)

Density of Comments è un indicatore di qualità che indica quanta parte del codice è stata commentata. Questo è calcolato tramite il rapporto tra le righe commentate (CLOC) e tutte le righe utilizzate (LOC).

$$DC = \frac{CLOC}{LOC} \tag{6}$$

(con valore compresi tra 0.0 e 1.0).
Se la densità dei commenti, è inferiore del 20% significa che il codice non è stato commentato abbastanza.

3.3.10.11 Weighted Methods per Class (WMC)

Weighted Methods per Class è la somma delle complessità ciclomatiche di tutti i metodi e i costruttori di una classe. Una classe con un WMC basso di solito punta a un maggiore polimorfismo mentre se il valore risulta alto indica che la classe è complessa, difficile da riutilizzare e da mantenere.

3.3.10.12 Average Interaction Density of Software Components (AIDC)

AIDC è la somma di densità di interazione per ciascun componente diviso per il numero di componenti:

- **CSS**: Numero componenti in un sistema software;
- **IDC$_n$**: Somma di densità delle interazioni per i componenti da 1 a n.

$$AID = \frac{(IDC_1 + IDC_2 + ...IDC_n)}{CSS} \tag{7}$$

Interaction Density of a Component (IDC) è il rapporto tra il numero effettivo di interazioni e le interazioni disponibili in un componente. Ad esempio, un componente può fornire 10 interfacce, ma potrebbero esserci solo 5 delle interfacce utilizzate nel sistema software, in tal caso il valore di IDC sarà 0,5. Il valore massimo di IDC è 1, il che significa che vengono utilizzate tutte le interfacce disponibili. L'equazione è data da:

$$IDC = \frac{I}{Imax} \tag{8}$$

dove I sono il numero di interazioni effettive e Imax sono le interazioni disponibili massime.

3.3.10.13 Test Coverage (TC)

Test Coverage rappresenta il livello di copertura che i test eseguiti forniscono rispetto alle funzionalità offerte dal prodotto. E' misurato come il rapporto tra il numero di test eseguiti ed il numero totale dei test che devono essere eseguiti.

- **TE**: Numero test eseguiti;
- **TDE**: Numero totale dei test da eseguire.

$$TC = \frac{TE}{TDE} * 100 \tag{9}$$

3.3.11 Metriche per i documenti

Per quanto riguarda i documenti, la metrica scelta è un indice di leggibilità basato sulla lingua italiana, gulpease.

3.3.11.1 Indice di Gulpease

Permette di indicare la leggibilità di un testo attraverso un valore compreso fra zero e cento ed è l'indice più diffuso data la sua semplicità d'uso.
L'indice Gulpease considera due variabili linguistiche: la lunghezza della parola e la lunghezza della frase rispetto al numero delle lettere ed è calcolato attraverso la seguente formula:

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

Più alto sarà il valore ottenuto da questa formula, maggiore sarà la leggibilità del documento e viceversa.
In generale è risultato che i testi con un indice:

- **[<80]**: Sono difficili da leggere per chi ha la sola licenza elementare;
- **[<60]**: Sono difficili da leggere per chi ha la sola licenza media;
- **[<40]**: Sono difficili da leggere per chi ha la licenza superiore.

3.3.11.2 Correttezza concettuale

Indica se un documento è corretto rispetto al suo contenuto, in altre parole se i suoi concetti sono coerenti e corretti lungo tutto il documento. L'indice ottenuto con questa metrica rappresenta quanti errori concettuali non sono stati corretti dopo esser stati segnalati da un *Verificatore*. Variabili utilizzate nella formula:

- **enc** = Numero di errori non ancora corretti;
- **ec** = Numero totale degli errori segnalati.

La metrica si calcola con la seguente formula:

$$Errori = \frac{enc}{ec} * 100 \tag{10}$$

I risultati ottenibili sono:

- **[>5]**: Valore non sostenibile;
- **[<=5]**: Valore sostenibile;
- **[0]**: Valore ottimale.

3.3.12 Metriche per il prodotto

3.3.12.1 Copertura requisiti obbligatori (CRO)

Questa metrica permette di tracciare il numero di requisiti obbligatori coperti dal prodotto software. Viene calcolata utilizzando la seguente formula:

- **ROS**: Numero requisiti obbligatori soddisfatti;
- **ROT**: Numero requisiti obbligatori totali.

$$CRO = \frac{ROS}{ROT} * 100 \tag{11}$$

3.3.12.2 Copertura requisiti desiderabili (CRD)

Questa metrica permette di tracciare il numero di requisiti desiderabili coperti dal prodotto software. Viene calcolata utilizzando la seguente formula:

- **RDS**: Numero requisiti desiderabili soddisfatti;
- **RDT**: Numero requisiti desiderabili totali.

$$CRD = \frac{RDS}{RDT} * 100 \tag{12}$$

3.3.12.3 Passed Test Cases Percentage (PTCP)

Passed Test Cases Percentage fornisce informazioni sull'efficienza del prodotto misurando la percentuale di test superati su un totale di prove eseguite.

- **TS**: Numero di test superati;
- **TE**: Numero totale di test eseguiti.

$$PTCP = \frac{TS}{TE} * 100 \tag{13}$$

3.3.12.4 Usability metrics for effectiveness

Effectiveness è indicata come la metrica fondamentale di usabilità che misura il tasso di completamento di un'azione. Esso viene calcolato assegnando un valore binario di "1" se l'utente al test riesce a completare un'attività e "0" se fallisce.

- **ACS**: Numero attività portate a termine con successo;
- **ADS**: Numero totale di attività da eseguire.

$$Effectiveness = \frac{ACS}{ADS} \tag{14}$$

3.3.12.5 Tempo medio di risposta (TRISP)

Il tempo medio di risposta di un'operazione indica il periodo temporale medio che intercorre fra la richiesta al software di una determinata funzionalità e la restituzione del risultato all'utente.

$$TRISP = \sum_{i=1}^{n} T_i \tag{15}$$

con TRISP espresso in secondi, dove T è il tempo intercorso fra la richiesta i di una funzionalità ed il completamento delle operazioni necessarie a restituire un risultato a tale richiesta.

3.4 Processo di gestione dei cambiamenti

3.4.1 Scopo del processo

Lo scopo di questo processo consiste nel gestire e, ove possibile, risolvere i problemi emersi durante lo sviluppo del progetto.

3.4.2 Aspettative

I risultati attesi sono:

- Procedure di segnalazione dei problemi;
- Assegnazione di priorità ai problemi in base alla gravità;
- Garanzia di risoluzione di un problema.

3.4.3 Descrizione del processo

Il processo consiste di procedure per la gestione e, ove possibile, risoluzione di problemi di qualsiasi natura. Le procedure sono finalizzate alla segnalazione, all'assegnazione a risorse e all'assegnazione di una priorità in base alla gravità del problema.

3.4.4 Metodi di segnalazione problemi nel codice

3.4.4.1 GitHub Issues

Per la gestione degli errori nel codice e il loro tracciamento viene utilizzato il processo di segnalazione di errori di GitHub. Per la segnalazione di errori utilizzando GitHub Issues è stato creato un template. Il template ha la funzione di fissare una struttura comune e di facilitare la comprensione per le segnalazioni. Il template consiste di due sezioni:

- **Situazione attesa**: Si identifica con il funzionamento atteso che il codice avrebbe dovuto seguire. I *Verificatori* dovranno evidenziare il funzionamento atteso fornendo esempi corretti di possibili soluzioni;
- **Situazione attuale**: Si identifica con il funzionamento che il codice ha effettuato. Qui i *Verificatori* dovranno includere l'indirizzo URL della riga di codice dove si è verificato il problema.

I *Verificatori* dovranno inoltre assegnare l'issue a una risorsa e associare al problema una priorità secondo tipologia del problema in questione e la sua gravità. L'issue può essere chiuso solo quando, all'avvenuta modifica delle parti segnalate, il *Verificatore* afferma la risoluzione di tale problema.

3.4.5 Metodi di segnalazione problemi universale

Per la segnalazione e risoluzione di problemi di qualsiasi natura, eccetto di codifica, è stato adottata una procedura interna. La procedura si basa sull'applicazione web di project management Asana. E' stata creata una colonna con la funzione di segnalazione dei problemi. L'apertura e la chiusura di un problema nella procedura di segnalazione dei problemi è responsabilità del *Verificatore*. Il *Verificatore* deve aprire un issue o aggiungere una colonna ogni volta che viene rilevato un problema. La procedura si svolge come segue:

- Viene cliccato il pulsante "Aggiungi nuovo task";
- Viene dato un titolo al task;
- Viene descritto il problema in maniera sintetica all'interno del task;
- Viene assegnata una priorità al task tramite la funzione di assegnazione personalizzata dei tag;
- Viene assocciata una data ultima entro cui risolvere il problema;
- Viene assegnata una risorsa all'analisi e risoluzione di tale problema.

Il problema può essere chiuso solo al soddisfacimento del problema sollevato.

4 Processi organizzativi

4.1 Processo di gestione di progetto

4.1.1 Scopo del processo

Lo scopo dei processi in questione è la produzione del documento *Piano di progetto v2.0.0*, rivolto a stabilire e implementare l'organizzazione e la definizione dei ruoli di ogni membro del team.

4.1.2 Aspettative

Le aspettative di tali processi sono:

- produzione del Piano di Progetto;
- definizione dei ruoli di progetto;
- definizione delle modalità di esecuzione dei compiti.

4.1.3 Descrizione del processo

Orario di lavoro:

- dalle 9.30 alle 12.30;
- dalle 14.30 alle 18.30.

Viene trattata la gestione dei seguenti argomenti:

- Ruoli di progetto;
- Comunicazioni;
- Incontri;
- Strumenti di coordinamento;
- Gestione dei rischi.

4.1.4 Ruoli di progetto

Lungo l'intero ciclo di vita del progetto, sono previsti diversi ruoli e si è scelto di farli ricoprire almeno una volta ad ogni membro. Nel *Piano di progetto v2.0.0* vengono pianificate le attività assegnate agli specifici ruoli di progetto. Le attività e i compiti per ogni ruolo sono brevemente spiegati come segue.

4.1.4.1 Amministratore di progetto

Il compito principale di chi ricopre il ruolo di *Amministratore* di progetto è controllare ed amministrare l'ambiente di lavoro, avendo diretta responsabilità sulla capacità operativa e sull'efficienza.
Le responsabilità assunte sono:

- Studio e ricerca di strumenti da fornire ai membri del gruppo che migliorino l'ambiente di lavoro, automatizzando dove possibile;
- Controllare versioni e configurazioni del prodotto software;
- Controllo della qualità sul prodotto, fornendo procedure e strumenti di monitoraggio o segnalazione;
- Risolvere i problemi legati alla gestione dei processi e delle risorse disponibili;
- Gestire il versionamento e l'archiviazione di tutta la documentazione di progetto.

4.1.4.2 Responsabile di progetto

Il *Responsabile di progetto* rappresenta il punto di riferimento del team, per il committente e il fornitore. Egli approva le scelte prese dal gruppo e se ne assume la responsabilità.
Le responsabilità assunte sono:

- Approvazione della documentazione;
- Approvazione dell'offerta economica;
- Gestione delle risorse umane;
- Coordinamento e pianificazione delle attività di progetto;
- Studio e gestione rischi.

4.1.4.3 Analista

L'*Analista* il compito di effettuare un'analisi approfondita sul dominio del problema, in modo da apprenderlo in maniera esaustiva. Non è necessaria la sua presenza durante l'intera durata del progetto.
Le responsabilità assunte sono:

- Comprensione della natura del problema e della sua complessità;
- Produzione dello *Studio di fattibilità v1.0.0* e dell'*Analisi dei requisiti v2.0.0*, delineando delle specifiche il più possibile comprensibili, sia dal proponente che dal committente.

4.1.4.4 Progettista

Il *Progettista* deve avere una conoscenza approfondita delle tecnologie utilizzate e competenze tecniche aggiornate, in modo da poter gestire gli aspetti tecnici e tecnologici del progetto.
Le responsabilità assunte sono:

- Effettuare scelte su aspetti tecnici del progetto, il più possibile efficienti ed ottimizzate;
- Effettuare scelte per rendere il prodotto facilmente mantenibile.

4.1.4.5 Verificatore

Il *Verificatore* deve avere un'ampia conoscenza delle normative del progetto, in modo da poterne garantire una corretta verifica.
La responsabilità assunta è:

- Controllare che le attività di progetto siano conformi alle norme stabilite.

4.1.4.6 Programmatore

Il *Programmatore* è responsabile delle attività di codifica e di creazione delle componenti di supporto, utili a effettuare le prove di verifica e validazione sul prodotto software.
Le responsabilità assunte sono:

- Implementare le soluzioni previste dal Progettista;
- Scrivere codice pulito e conforme alle norme di progetto, in modo da essere facilmente mantenibile;
- Versionare il codice prodotto;
- Realizzare gli strumenti utili per poter compiere le prove di verifica e validazione previste.

4.1.5 Gestione delle comunicazioni

La sezione in questione espone le norme da rispettare per le comunicazioni tra le varie parti (proponente, committenti, componenti del gruppo).

4.1.5.1 Comunicazioni interne

Per le comunicazioni interne tra membri del gruppo *WarMachine* viene utilizzato un software denominato slack$_G$, uno dei maggiori strumenti di messaggistica per la collaborazione aziendale. Questo servizio permette la creazione di canali specifici per una determinata attività, in cui è possibile lo scambio di informazioni e la condivisione di file che potranno essere accessibili a tutto il Team o solo ad alcuni membri.
Per aumentare la produttività, è stato scelto di attivare l'integrazione di Slack con gli altri tool utilizzati: Asana e GitHub.
I canali scelti di utilizzare sono:

- **asana**: Per visualizzare i compiti di una attività assegnati ad ogni membro del gruppo;

- **capitolati**: Per la stesura dello Studio di fattbilità;
- **docs**: Per condividere e per avere in facile reperibilità le documentazioni di scopo informativo per lo sviluppo del sistema;
- **domande**: Per la stesura di domande rivolte ai committenti e proponente;
- **general**: Per discutere gli argomenti di studio individuale;
- **git_code**: Per le notifiche del Bot sui cambiamenti avvenuti sul repository GitHub e relative domande riguardante il codice;
- **git_docs**: Per le notifiche del Bot sui cambiamenti avvenuti sul repository GitHub e relative domande riguardante i documenti;
- **glossario**: Canale per la stesura del *Glossario v2.0.0*;
- **immaginipresentazione**: Per la condivisioni di immagini da utilizzare nelle presentazioni;
- **issues**: Per la segnalazione di errori;
- **links**: Per la condivisione di link utili allo studio del progetto;
- **norme**: Canale relativo alle *Norme di progetto v2.0.0*;
- **pdq**: Canale relativo alle *Piano di qualifica v2.0.0*;
- **perilcodice**: Canale relativo alle decisioni sul codice;
- **presentazione**: Canale relativo alla stesura delle varie presentazioni;
- **tex**: Dedicato alla segnalazione di errori riguardanti i file LaTeX;
- **ui**: Canale per l'organizzazione dell'interfaccia grafica.

Un'altra possibilità di comunicazione usata solo per le organizzazioni sui ritrovi interni, è il gruppo creato su Telegram, un servizio di messaggistica istantanea basato su cloud, denominato "SWE - WarMachine", a cui hanno accesso solo i componenti del gruppo. In caso sia necessario effettuare delle videoconferenze, verrà utilizzata l'applicazione Google Hangouts.

4.1.5.2 Comunicazioni esterne

E' compito del *Responsabile di progetto* occuparsi dei contatti con le componenti esterne al gruppo attraverso la casella email di Google creata appositamente per il team: warmachine.swe@gmail.com. Se necessario poi, sarà lui stesso ad informare gli altri membri del gruppo su quanto discusso con le componenti esterne.
I contatti delle componenti esterne a cui si può fare riferimento sono:

- **tullio.vardanega@math.unipd.it**: Email del committente;
- **rcardin@math.unipd.it**: Email del committente;

- **Gregorio.Piccoli@zucchetti.it**: Email del contatto dell'azienda *Zucchetti s.p.a.*

Un altro strumento di comunicazione di cui si prendono impegno tutti i membri, sono i gruppi Telegram:

- **"Coordinamento capitolato 5"**: Per dubbi e organizzazione sui requisiti, incontri con l'azienda *Zucchetti s.p.a*;
- **"Coordinamento SWE 17/18"**: Per le comunicazioni con tutti i gruppi entranti al secondo lotto sulla scelta dei capitolati e l'organizzazione degli incontri con i committenti.

4.1.6 Gestione degli incontri

4.1.6.1 Incontri interni

Il *Responsabile di progetto* ha il compito di organizzare gli incontri interni, utilizzando Telegram, l'applicazione aderente alle norme (espressa nelle *Comunicazioni interne*, vedi 4.1.5.1), in modo da avere conferma della presenza di ogni membro. Successivamente dovrà fissare l'evento su Asana. Ogni membro del gruppo può richiedere al *Responsabile di progetto* di organizzare un incontro interno segnalando una motivazione e sarà compito di quest'ultimo controllare la disponibilità degli altri membri, e in caso accettare o rifiutare.

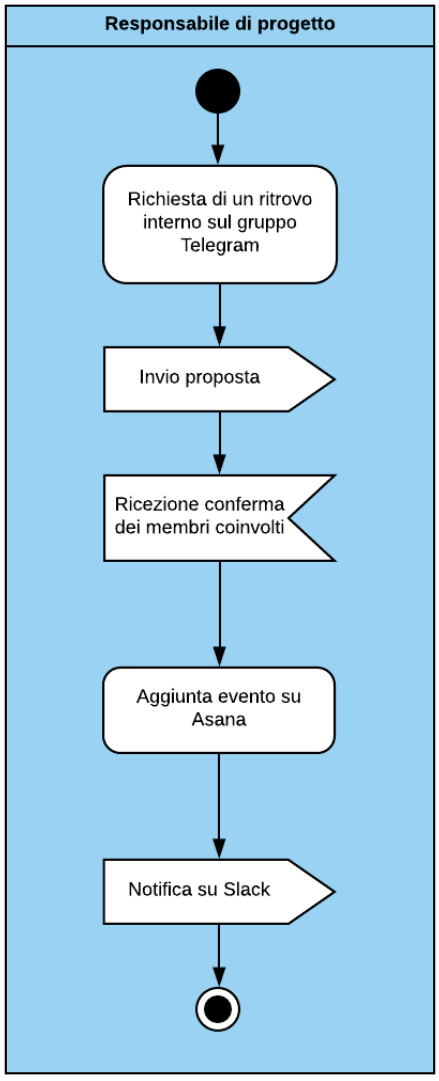

Figura 6: Procedura per l'organizzazione di un ritrovo interno.

4.1.6.2 Incontri esterni

Il *Responsabile di progetto* seguendo le norme (espresse nelle *Comunicazioni esterne*, vedi 4.1.5.2) deve prendere atto a:

- Fissare l'incontro sul servizio di Asana;
- Incaricare un membro del gruppo per stilare il verbale dell'incontro.

Ogni membro del gruppo può richiedere al *Responsabile di progetto* di organizzare un incontro esterno segnalando una motivazione e sarà compito di quest'ultimo controllare la disponibilità degli altri membri, delle componenti esterne e in caso accettare o rifiutare.

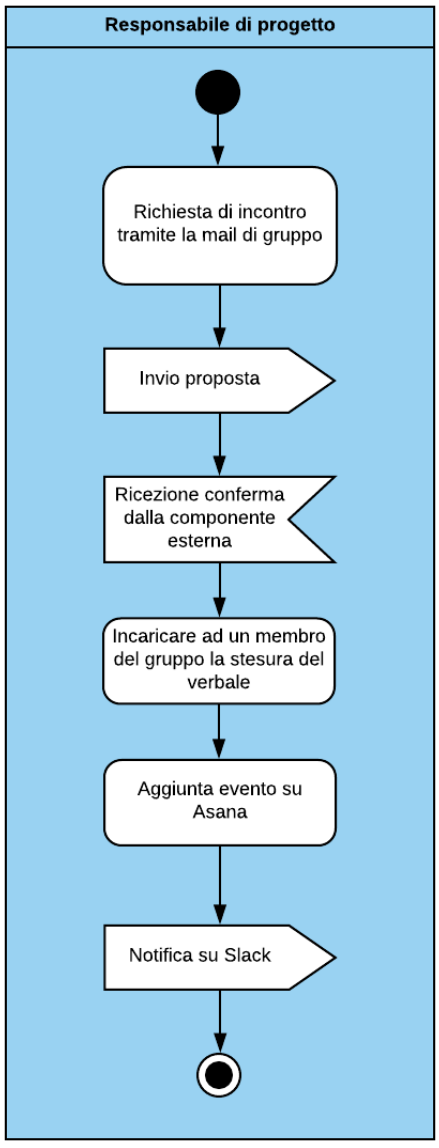

Figura 7: Procedura per l'organizzazione di un ritrovo esterno.

4.1.7 Strumenti di coordinamento

4.1.7.1 Ticketing

Per distribuire il carico di lavoro, il *Responsabile di progetto* assegna le attività fra i membri grazie ad Asana, un'applicazione web per la gestione di progetti. Ogni membro dopo aver effettuato la registrazione, avrà accesso alla bacheca del gruppo "WarMachineSwe" contenente un board con una serie di colonne rappresentati le attività da svolgere per la revisione corrente. Per assegnare un ticket il processo è il seguente:

- Cliccare "Aggiungi colonna" e scrivere il nome dell'attività;
- Aggiungere i membri a cui tale attività deve essere assegnata;
- Fissare una data di scadenza concordata con i membri del gruppo;
- Impostare una checklist con l'elenco delle sezioni;
- Cliccare sul simbolo "+" e inserire una descrizione contenente i compiti necessari allo svolgimento dell'attività e il ruolo assunto in quella fase del progetto.

La persona a cui è stata assegnata l'attività riceverà la notifica sul canale Slack dedicato seguendo le norme indicate precedentemente su 4.1.5.1.

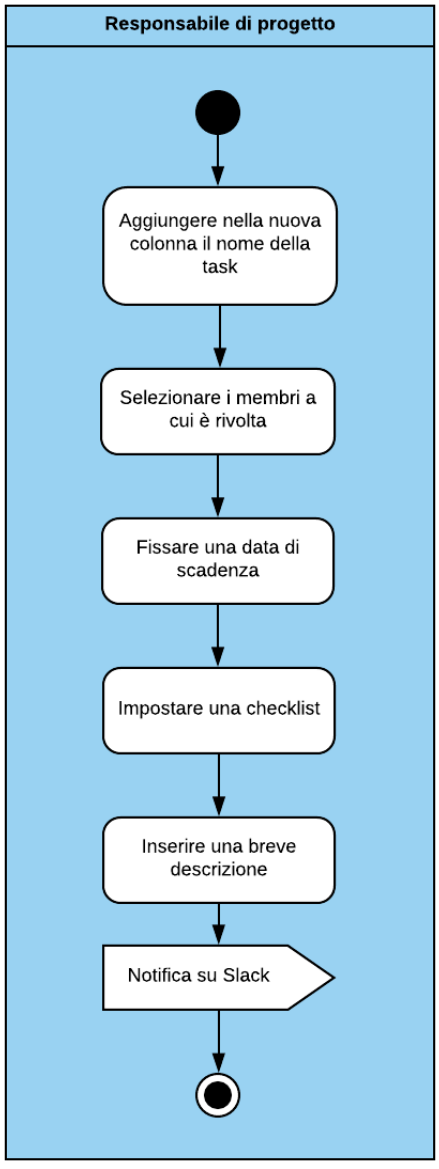

Figura 8: Procedura per l'assegnazione di un ticket.

4.1.8 Strumenti per il versionamento

4.1.8.1 GitHub

GitHub è un servizio web di code hosting basato su Git. Gli utenti possono interagire tramite un sistema di issues tracking$_G$, pull request$_G$ e commenti che permette di migliorare il codice del repository (pubblico (disponibile gratuitamente) o privato (disponibile a pagamento), molto utilizzati per lo sviluppo di progetti open-source) risolvendo bug o aggiungendo funzionalità.

4.1.8.2 GitHub Desktop

GitHub Desktop è l'estensione desktop per lavorare sui vari repository a cui si ha accesso nel corrispondente servizio web GitHub. Tra le sue funzioni permette di eseguire:

- La visualizzazione delle modifiche apportate al repository;
- Nei vari branch azioni di commit selezionando da una lista i file a cui apportare modifiche;
- Effettuare il merge delle modifiche locali con quelle dello storico.

E' accessibile il download al seguente link:

https://desktop.github.com

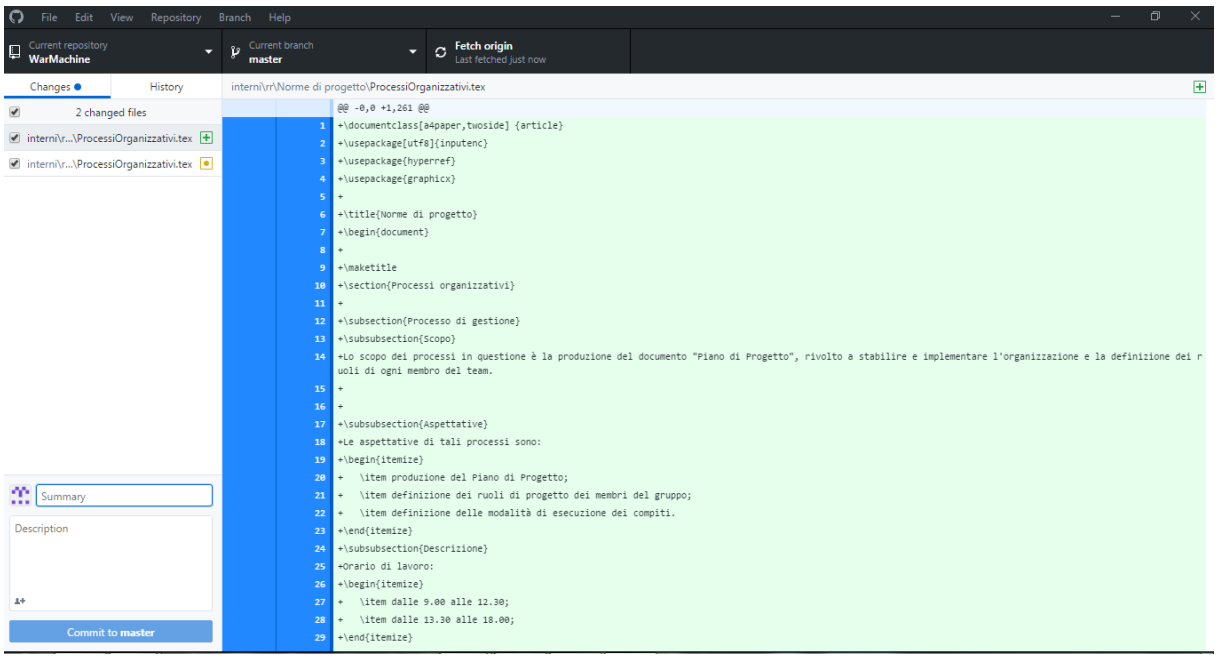

Figura 9: GitHub Desktop su Windows.

4.1.9 Gestione dei rischi

Il *Responsabile di progetto* ha il compito di rilevare i rischi indicati nel documento *Piano di progetto v2.0.0* e nel caso ne vengano individuati di nuovi deve aggiungerli alla sezione di analisi dei rischi. La procedura per la gestione segue il seguente schema:

- Individuare i problemi non calcolati e monitorare i rischi già previsti;
- Registrare ogni riscontro dei rischi nel *Piano di progetto v2.0.0*;
- Aggiungere se sono stati rilevati nuovi rischi nel *Piano di progetto v2.0.0*;
- Ridefinire, se necessario, le strategie di progetto.

4.1.10 Strumenti

4.1.10.1 Sistemi operativi

Il gruppo opera su diversi sistemi operativi:

- MacOS High Sierra 10.13;
- GNU/ Linux$_G$ almeno 14.04 LTS;
- Windows 10 Pro;
- Windows 10 Home.

4.1.10.2 Slack

Slack è uno strumento di collaborazione utile per le comunicazioni interne ad un gruppo di lavoro. L'applicazione appare come un comune sistema di messaggistica, con in più la possibilità di integrare numerosi servizi tra cui GitHub e Gmail. Gli utenti del team possono suddividere gli argomenti di discussione utilizzando degli hashtag$_G$ per organizzare al meglio la comunicazione. L'applicazione è gratuita ed è necessaria la registrazione di un dominio per il proprio Team. Una volta registrati gli utenti possono essere invitati all'interno del gruppo. Lo spazio dedicato al gruppo si trova al seguente indirizzo:

https://swe-group-17.slack.com

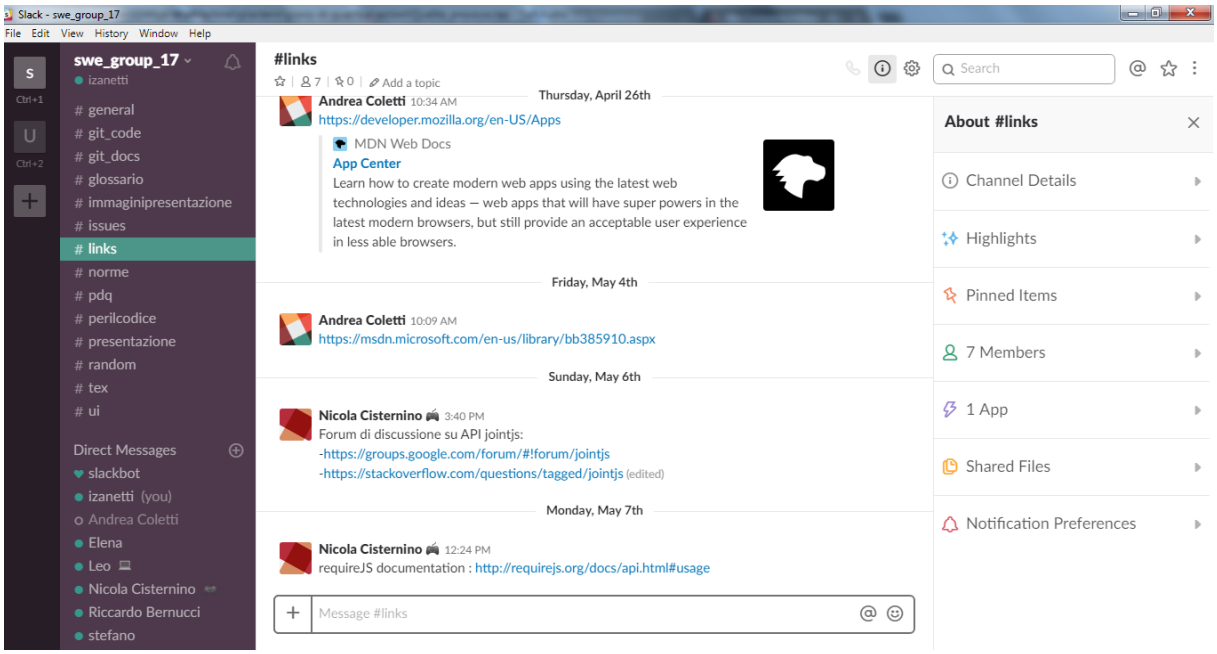

Figura 10: Immagine rappresentativa del profilo Slack del team WarMachine.

4.1.10.3 Telegram

Telegram è un servizio di messaggistica istantanea, usato per le comunicazioni interne ed esterne informali secondo le norme indicate. I client$_G$ ufficiali di Telegram sono distribuiti come software libero per molteplici piattaforme.

4.1.10.4 Asana

Asana è un'applicazione web di gestione di progetti, il cui obiettivo è la corretta suddivisione dei carichi di lavoro tra i membri del gruppo. Essa permette di organizzare il progetto in attività, e le attività in compiti assegnabili ai membri del progetto.
Ai compiti deve essere attribuita una data di calendario che indicherà la scadenza prevista entro cui si vorrebbe concludere il compito. L'applicazione web offre una elevata portabilità ed accessibilità, essendo fruibile direttamente dal web gratuitamente, con un design user-friendly$_G$.

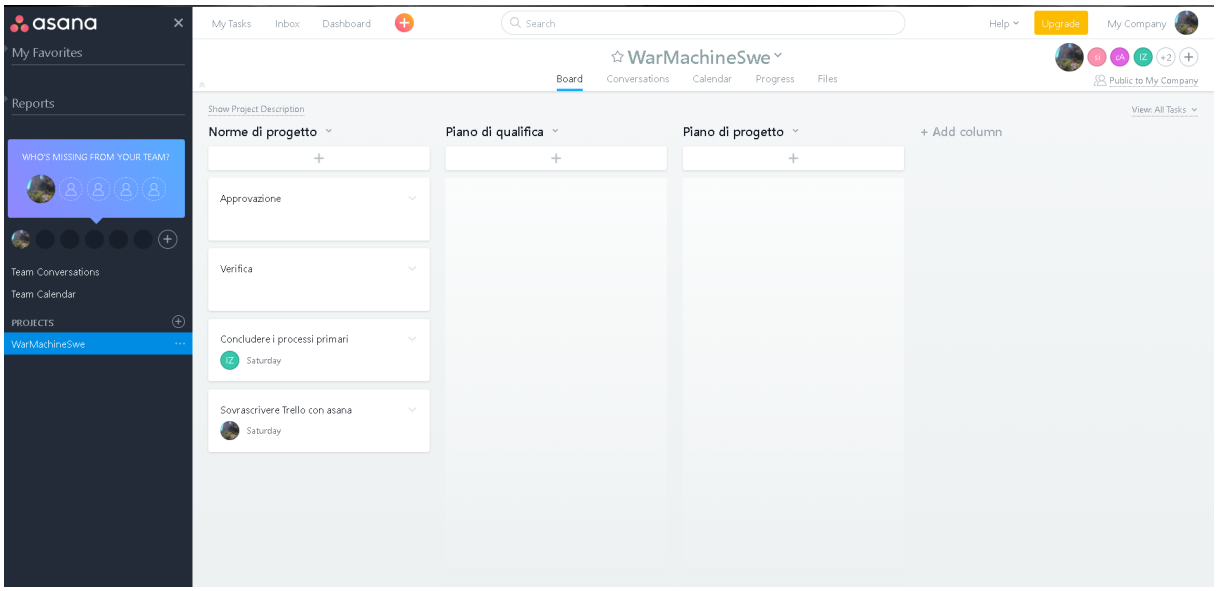

Figura 11: Asana visualizzato sul web.

4.1.10.5 GanttProject

Come strumento per il supporto alla pianificazione di progetto e il corretto allocamento delle risorse è stato scelto GanttProject. GanttProject è uno strumento open-source che permette la creazione diagrammi di Gantt, organizzare i compiti in una work breakdown structure e auto-generare, a partire dai diagrammi di Gantt realizzati, i diagrammi di PERT.

4.2 Processo di formazione

4.2.1 Scopo del processo

Lo scopo di questo processo è l'apprendimento delle conoscenze richieste dalle tecnologie e metodologie richieste nello sviluppo del progetto.

4.2.2 Aspettative

Le aspettative di questo processo riguardano la formazione del personale in modo tale da renderlo in grado di partecipare efficentemente allo sviluppo del progetto commissionato.

4.2.3 Descrizione del processo

Il processo di formazione è responsabilità del singolo individuo. L'individuo deve dedicare parte del suo tempo, che viene concordato internamente al gruppo, all'apprendimento delle tecnologie e delle conoscenze che vengono specificate nel capitolato d'appalto o che il gruppo sceglie di adottare. Il tempo di studio individuale per l'apprendimento viene fissato universalmente all'interno del gruppo, non viene considerato lavoro rendicontato in quanto necessario per la partecipazione al progetto.

4.2.4 Scambio di materiale a fine formativo

Lo scambio di documenti all'interno del gruppo, al fine di favorire l'apprendimento delle metodologie e delle tecnologie richieste, deve avvenire tramite la piattaforma di comunicazione Slack, attraverso i canali adibiti docs e links.

4.2.5 Strumenti

4.2.5.1 W3schools

Per l'approccio e l'apprendimento delle tecnologie web richieste dal progetto si è deciso di seguire i tutorial messi a disposizione dal sito web:

https://www.w3schools.com