Norme di Progetto

Informazioni sul documento

Versione	4.0.0
Data di Creazione	2016-12-03
Data ultima modifica	2017-07-10
Stato	Approvato
Redazione	Jordan Gottardo Giulia Petenazzi Leonardo Brutesco
Verifica	Giovanni Prete
Approvazione	Marco Pasqualini
Uso	Interno
Lista di distribuzione	Professor Tullio Vardanega Professor Riccardo Cardin *Zephyrus*
Email di riferimento	zephyrus.swe@gmail.com

Registro delle modifiche

Versione	Data	Autore	Ruolo	Descrizione
4.0.0	2017-07-10	Marco Pasqualini	*Responsabile*	Approvazione
3.1.0	2017-07-06	Giovanni Prete	*Verificatore*	Verifica documento
3.0.2	2017-07-05	Leonardo Brutesco	*Analista*	Aggiunta sezione per la consegna
3.0.1	2017-06-26	Giulia Petenazzi	*Analista*	Correzioni e modifche minori al processo di sviluppo
3.0.0	2017-04-08	Giovanni Prete	*Responsabile*	Approvazione
2.1.0	2017-04-03	Giovanni Damo	*Verificatore*	Verifica documento
2.0.6	2017-04-03	Jordan Gottardo	*Amministratore*	Aggiunte norme per i manuali utente e manutentore
2.0.5	2017-04-01	Giulia Petenazzi	*Amministratore*	Aggiunte norme di codifica per eseguire import ed export
2.0.4	2017-03-25	Jordan Gottardo	*Amministratore*	Approfondite sezioni di progettazione ad alto livello e di dettaglio
2.0.3	2017-03-19	Jordan Gottardo	*Amministratore*	Aggiunta procedura configurazione Webstorm
2.0.2	2017-03-18	Leonardo Brutesco	*Amministratore*	Aggiunta procedura glossario nei manuali, ampliate norme sullo stile codifica, aggiunti comandi nell'appendice, approfondite norme progettazione, corretta sezione 2.2.5.3
2.0.1	2017-03-15	Leonardo Brutesco	*Amministratore*	Aggiunte procedure per configurazione VirtualBox e FileZilla
2.0.0	2017-03-04	Giovanni Damo	*Responsabile*	Approvazione
1.3.0	2017-03-03	Jordan Gottardo	*Verificatore*	Verifica documento
1.2.1	2017-03-03	Marco Pasqualini	*Analista*	Corretta descrizione WebStorm
1.2.0	2017-03-02	Jordan Gottardo	*Verificatore*	Verifica documento

Versione	Data	Autore	Ruolo	Descrizione
1.1.5	2017-03-02	Marco Pasqualini	*Analista*	Aggiunto diagramma procedura chiusura ticket, aggiunti WebStorm, JSDoc e Babel agli strumenti del processo di sviluppo
1.1.4	2017-03-02	Marco Pasqualini	*Analista*	Aggiunta procedura di rinvio dei task e sezione per condivisione in apprendimento
1.1.3	2017-03-02	Marco Pasqualini	*Analista*	Aggiunte procedure per il calcolo delle metriche
1.1.2	2017-03-01	Giulia Petenazzi	*Analista*	Creazione tabelle riassuntive metriche-strumenti-procedure
1.1.1	2017-02-28	Marco Pasqualini	*Analista*	Aggiunte metriche al processo di verifica. Correzione errori ortografici
1.1.0	2017-02-26	Jordan Gottardo	*Verificatore*	Verifica documento
1.0.10	2017-02-25	Marco Pasqualini	*Analista*	Aggiunta classificazione test e correzione errori nel processo di sviluppo
1.0.9	2017-02-23	Marco Pasqualini	*Analista*	Aggiunte procedure per installazione macchina virtuale e segnalazione richieste
1.0.8	2017-02-21	Giulia Petenazzi	*Analista*	Ampliata sezione gestione delle infrastrutture, aggiunti strumenti Node.js, npm, VirtualBox e integrazioni Slack
1.0.7	2017-02-19	Marco Pasqualini	*Analista*	Ampliate procedure utilizzo Trender
1.0.6	2017-02-17	Marco Pasqualini	*Analista*	Aggiunti strumenti ESLint e JSMeter agli strumenti di analisi statica, Karma, Jasmine e Istanbul agli strumenti di analisi dinamica

Versione	Data	Autore	Ruolo	Descrizione
1.0.5	2017-02-15	Marco Pasqualini	*Analista*	Creato processo di gestione della configurazione. Spostate sezioni Repository, Versionamento, Git, GitHub e relative procedure in gestione della configurazione
1.0.4	2017-02-15	Marco Pasqualini	*Analista*	Aggiunta appendice per comandi LaTeX
1.0.3	2017-02-13	Marco Pasqualini	*Analista*	Inserite procedure per Astah, calcolo Indice Gulpease e aggiunti termini al glossario
1.0.2	2017-02-12	Marco Pasqualini	*Analista*	Inserite procedure di consegna, di creazione dei documenti e di utilizzo Slack. Aggiornate sezioni LaTeX e Apprendimento
1.0.1	2017-01-29	Marco Pasqualini	*Analista*	Messi a glossario i termini solo la prima volta che compaiono in ogni sezione del documento
1.0.0	2016-12-23	Giulia Petenazzi	*Responsabile*	Approvazione
0.2.0	2016-12-20	Leonardo Brutesco	*Verificatore*	Verifica documento
0.1.1	2016-12-20	Giovanni Prete	*Analista*	Correzione errori
0.1.0	2016-12-17	Leonardo Brutesco	*Verificatore*	Verifica documento
0.0.10	2016-12-17	Giovanni Prete	*Analista*	Correzioni e aggiornamento diagrammi
0.0.9	2016-12-16	Giovanni Damo	*Analista*	Correzioni sezione processi primari
0.0.8	2016-12-13	Giovanni Prete	*Analista*	Correzioni sezione processi di organizzativi
0.0.7	2016-12-09	Giovanni Damo	*Analista*	Correzioni sezione processi di supporto
0.0.6	2016-12-07	Giovanni Prete	*Analista*	Aggiunte descrizioni attività e processi
0.0.5	2016-12-07	Giovanni Damo	*Analista*	Terminata sezione processi di supporto
0.0.4	2016-12-05	Giovanni Damo	*Analista*	Terminata sezione processi organizzativi

Versione	Data	Autore	Ruolo	Descrizione
0.0.3	2016-12-05	Giovanni Prete	*Analista*	Terminata sezione processi primari
0.0.2	2016-12-04	Giovanni Prete	*Analista*	Terminata introduzione
0.0.1	2016-12-03	Giovanni Prete	*Analista*	Creazione template e indice

Indice

1 Introduzione

1.1 Scopo del documento

Questo documento definisce le norme che i membri del *gruppo*$_G$ *Zephyrus* si impegnano a rispettare per un corretto svolgimento del progetto *DeGeOP*. Ogni componente del gruppo è tenuto a leggere e rispettare quanto scritto al fine di:

- garantire uniformità nel materiale prodotto;
- favorire la cooperazione tra i membri del gruppo;
- raggiungere il miglior rapporto tra efficacia ed efficienza.

Il presente documento definisce le norme inerenti a:

- comunicazioni interne ed esterne al gruppo;
- stesura dei documenti e convenzioni tipografiche;
- stesura del codice;
- modalità di lavoro durante il *ciclo di vita*$_G$ del progetto;
- organizzazione dell'ambiente di lavoro e di sviluppo.

Ogni membro del gruppo sarà informato di eventuali modifiche alle norme esistenti o aggiunta di nuove norme.

1.2 Scopo del prodotto

Lo scopo del prodotto consiste nella creazione di un'interfaccia web contenente una mappa geografica su cui potranno essere rappresentati:

- il processo produttivo aziendale;
- gli scenari di danno;
- i risultati dell'analisi dei rischi.

Il prodotto verrà utilizzato da agenti assicuratori per l'inserimento delle informazioni utili allo svolgimento dell'analisi dei rischi dell'assicurando.
L'interfaccia dovrà essere in grado di connettersi ai sistemi preesistenti di *RiskApp* per la memorizzazione e gestione dei dati inseriti. A causa del requisito di integrabilità, che è stato deciso di soddisfare, l'applicazione da sviluppare sarà parte integrante dell'attuale applicazione del proponente.

1.3 Glossario

Allo scopo di rendere più semplice e chiara la comprensione dei documenti viene allegato il *Glossario v2.0.0*, nel quale verranno raccolte le spiegazioni di terminologia tecnica o ambigua, abbreviazioni ed acronimi. Per evidenziare un termine presente in tale documento, esso verrà marcato con il pedice $_G$. Solo la prima occorrenza del termine in ogni sezione sarà marcata per non appesantire la lettura del documento.
Tutti i termini del glossario evidenziati sono link ipertestuali al glossario stesso; affinché funzionino correttamente è necessario che la posizione delle directory e dei file forniti non venga alterata.

1.4 Riferimenti

1.4.1 Riferimenti normativi

- standard *ISO*$_G$-8601:
  https://en.wikipedia.org/wiki/ISO_8601;
- standard ISO/*IEC*$_G$ 12207:
  https://en.wikipedia.org/wiki/ISO/IEC_12207;
- specifica *UML*$_G$ 2.0:
  http://www.omg.org/spec/UML/2.0/.

1.4.2 Riferimenti informativi

- standard AS/NZS ISO/IEC 12207:1997:
  http://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf;
- *capitolato*$_G$:
  http://www.math.unipd.it/~tulXlio/IS-1/2016/Progetto/C3.pdf;
- utilizzo di *Teamwork*$_G$:
  http://support.teamwork.com/projects/start/getting-started;
- utilizzo di *Git*$_G$:
  https://git-scm.com/doc;
- utilizzo di *GitHub*$_G$:
  https://guides.github.com/;
- utilizzo di *Astah*$_G$:
  http://astah.net/tutorials;
- utilizzo di *Slack*$_G$:
  https://get.slack.help/hc/en-us.

2 Processi primari

2.1 Fornitura

2.1.1 Scopo

Lo scopo del processo di fornitura è di determinare le procedure e le risorse necessarie allo svolgimento del progetto. Le attività di cui si compone sono:

- studio di fattibilità;
- contrattazione;
- collaudo.

La corretta implementazione del processo deve:

- decidere il progetto da svolgere;
- fissare gli obiettivi per la contrattazione.

2.1.2 Studio di fattibilità

2.1.2.1 Scopo

Individuare gli aspetti fondamentali dei progetti proposti, tramite lo studio dei *capitolati*$_G$ ed il confronto tra i membri del *gruppo*$_G$.

2.1.2.2 Discussione e scelta del capitolato

Il *Responsabile di progetto* ha il compito di organizzare gli incontri del gruppo necessari ad analizzare i capitolati proposti.

2.1.2.3 Struttura studio di fattibilità

Il documento creato dagli *Analisti* per ogni capitolato deve rispettare i seguenti punti:

- **descrizione:** descrizione del prodotto richiesto dal capitolato;
- **dominio applicativo:** ambito di utilizzo del prodotto;
- **dominio tecnologico:** tecnologie richieste per lo sviluppo del progetto;
- **criticità:** individuazione di punti critici e possibili problematiche che potrebbero sorgere durante lo svolgimento del progetto;
- **valutazione finale:** considerazioni finali sulla scelta di accettare o meno il capitolato preso in esame.

2.1.3 Contrattazione

2.1.3.1 Scopo

Presentare una proposta in risposta al capitolato del proponente.

2.1.3.2 Preparazione della proposta

Il gruppo deve redigere e consegnare i seguenti documenti:

- *Norme di progetto*;
- *Studio di fattibilità*;
- *Analisi dei requisiti*;
- *Piano di progetto*;
- *Piano di qualifica*.

Verrà inoltre fornita in allegato la *Lettera di presentazione* del gruppo. Si veda la sezione 3.1.6 per maggiori informazioni sulla gestione dei documenti.

2.1.4 Consegna

2.1.4.1 Scopo

Preparare il materiale richiesto per la **Revisione di accettazione**.

2.1.4.2 Preparazione della consegna

Deve essere preparato un supporto ottico (CD/DVD) contente:

- la *Lettera di presentazione* del gruppo per la **Revisione di accettazione**;
- le versioni finali di tutti i documenti;
- il prodotto finale realizzato comprensivo di:
  - codice sorgente;
  - eventuali unità di compilazione e installazione;
  - istruzioni per l'uso.

Il *Responsabile di progetto* ha il compito di contattare il committente per richiedere un appuntamento per consegnare il supporto preparato.

2.1.4.3 Consegna

Il *Responsabile di progetto* deve contattare il proponente per accordarsi sulle modalità di consegna del prodotto finale.

2.1.5 Procedure

2.1.5.1 Primo contatto con il proponente

Il primo contatto con il proponente da parte del gruppo *Zephyrus* deve essere effettuato da parte del *Responsabile di progetto*. L'oggetto del messaggio deve contenere la sigla del progetto; il messaggio deve essere realizzato secondo la seguente struttura:

```
Spett.le Nome Azienda,
alla cortese attenzione del CEO/Responsabile Nome Cognome;

... corpo del messaggio ...

Vi ringraziamo e speriamo di poterci incontrare presto di persona.

Cordiali saluti,
Zephyrus.
```

2.1.5.2 Consegna materiale per revisione di avanzamento

La consegna del materiale prodotto deve rispettare i vincoli temporali descritti al seguente indirizzo:

http://www.math.unipd.it/~tullio/IS-1/2016/

L'invio del materiale deve essere gestito nel seguente modo:

1. creare una cartella compressa contenente tutti e soli i documenti inerenti la consegna in formato pdf. Il nome della cartella sarà: `Zephyrus-XX.zip` dove al posto di XX sarà presente il codice identificativo della revisione di avanzamento;
2. caricarla sullo spazio web del gruppo, disponibile al seguente indirizzo:

   Altervista Zephyrus

3. contattare con l'email del gruppo, il committente all'indirizzo `tullio.vardanega@math.unipd.it` e allegare il link al file precedentemente caricato su Altervista.

2.2 Sviluppo

2.2.1 Scopo

Il processo di sviluppo contiene le attività necessarie a produrre il prodotto software richiesto. In accordo con lo standard *ISO*$_G$/*IEC*$_G$ (1.4.2), preso come riferimento, si è deciso di istanziare le seguenti attività:

- analisi dei requisiti;
- progettazione;
- codifica.

La corretta implementazione del processo deve:

- fissare gli obiettivi di sviluppo;
- fissare i vincoli tecnologici;
- realizzare un prodotto finale che soddisfi i test di accettazione e che sia conforme alle richieste del proponente.

2.2.2 Analisi dei requisiti

2.2.2.1 Scopo

Individuare i requisiti del progetto tramite lo studio del capitolato ed incontri con il proponente. Individuare i test di sistema. Il risultato deve essere presentato nel documento formale *Analisi dei requisiti* che deve contenere la lista dei casi d'uso e dei requisti.

2.2.2.2 Classificazione dei casi d'uso

I casi d'uso individuati devono essere classificati secondo la seguente notazione:

UC[Codice padre].[Codice identificativo]

dove:

- **codice padre:** indica il codice numerico in forma gerarchica del caso d'uso da cui deriva, viene omesso se non identificabile;
- **codice identificativo:** indica il codice numerico del caso d'uso.

Per ogni caso d'uso bisogna indicare:

- **titolo:** titolo riassuntivo dell'operazione che il caso d'uso modella;
- **attori:** elenco attori coinvolti;
- **descrizione:** concisa e meno ambigua possibile;
- **pre-condizione:** condizioni sempre vere riferite allo stato del sistema che abilitano lo svolgimento del caso d'uso;
- **post-condizione:** condizioni sempre vere riferite allo stato del sistema dopo lo svolgimento del caso d'uso;
- **scenario principali:** ordine con cui vengono eseguiti i casi d'uso figli;
- **scenari alternativi:** possibili scenari alternativi del caso d'uso;
- **estensioni:** spiegazione di tutte le estensioni, se presenti;
- **inclusioni:** spiegazione di tutte le inclusioni, se presenti;
- **generalizzazioni:** spiegazione di tutte le generalizzazioni, se presenti.

2.2.2.3 Classificazione dei requisiti

I requisiti individuati devono essere classificati secondo la seguente notazione:

R[Importanza][Tipologia][Codice]

dove:

- **importanza:** può assumere questi valori:
  - **O:** indica un requisito obbligatorio;
  - **D:** indica un requisito desiderabile;

  - **F:** indica un requisito opzionale (facoltativo).

- **tipologia:** può assumere questi valori:
  - **F:** indica un requisito funzionale;
  - **Q:** indica un requisito di qualità;
  - **P:** indica un requisito prestazionale;
  - **V:** indica un requisito di vincolo.
- **codice:** codice numerico che identifica il requisito, deve essere univoco ed indicato in forma gerarchica, da sinistra a destra, nella notazione X.Y.Z.

Per ogni requisito bisogna inoltre indicare:

- **descrizione:** concisa e meno ambigua possibile;
- **fonte:** l'origine dei requisiti deve essere una delle seguenti:
  - **capitolato:** derivato direttamente dal testo del capitolato;
  - **interno:** deriva da discussioni interne al gruppo;
  - **verbale:** deriva da un verbale steso in seguito ad un incontro con il proponente. Deve essere indicato il nome del verbale a cui si riferisce;
  - **casi d'uso:** deriva da uno o più casi d'uso. Deve essere indicato il codice identificativo dei casi d'uso a cui si riferisce.

2.2.2.4 Diagrammi UML

I diagrammi *UML*$_G$ devono essere realizzati seguendo lo standard UML versione 2.
Per facilitare la lettura dei diagrammi si devono seguire le seguenti convenzioni generali:

- gli elementi devono essere il più possibile omogenei tra loro per dimensione;
- gli elementi devono essere allineati tra loro, sia orizzontalmente che verticalmente, quando possibile;
- i margini di spazio tra gli elementi di un gruppo devono rimanere invariati in gruppi analoghi se si hanno le stesse tipologie di elementi;
- i collegamenti in uscita da un singolo elemento devono essere ad angolo retto invece che diretti.

2.2.3 Progettazione ad alto livello

2.2.3.1 Scopo

Definire la struttura di alto livello del software e identificare le sue componenti. Definire le interfacce esterne ed interne. Individuare i test di integrazione. Il risultato deve essere presentato nel documento formale *Specifica tecnica*.

2.2.3.2 Specifica tecnica

2.2.3.2.1 Diagrami UML

Devono essere forniti i seguenti diagrammi:

- **diagrammi di classe:** hanno lo scopo di descrivere entità con le loro caratteristiche e relazioni, pertanto vengono definiti attributi, metodi e relazioni. tra di esse. Nella progettazione ad alto livello non è necessario che questi diagrammi siano particolarmente dettagliati;
- **diagrammi dei *package*$_G$:** hanno lo scopo di rappresentare la struttura interna del progetto software modellato in termini dei suoi componenti principali. Questo tipo di diagramma viene utilizzato per evidenziare i package e le relazioni tra i componenti interni ed esterni che siano;
- **diagrammi di attività:** hanno lo scopo di definire le attività da svolgere per realizzare una certa funzionalità. Vengono dunque usati per mostrare come determinate interazioni tra componenti realizzino una funzionalità che si intende rendere disponibile.

Viene introdotta una notazione che utilizza i colori per distinguere la provenienza delle componenti dell'applicazione:

- giallo: *componente*$_G$ da implementare;
- rosso: componente offerta da *RiskApp*;
- verde: componente importata da terze parti.

2.2.3.2.2 Design pattern

Per migliorare la comprensibilità delle scelte progettuali e della progettazione stessa devono essere forniti i *design pattern*$_G$ utilizzati secondo la seguente struttura:

- descrizione testuale;
- descrizione grafica;
- motivazione e descrizione dell'utilizzo all'interno del progetto.

2.2.3.2.3 Classificazione dei componenti

I componenti vengono identificati univocamente da una descrizione nella forma:

[NomeProdotto]::[NomeComponente]

dove:

- **NomeProdotto:** è il nome del prodotto software che i *Progettisti* hanno individuato;
- **NomeComponente:** rappresenta il nome assegnato al componente dai *Progettisti*.

Inoltre, ogni componente ha associato un codice univoco nella forma:

[X][Y]

dove:

- **X:** è l'iniziale del nome del prodotto;
- **Y:** è un numero intero incrementale.

2.2.3.2.4 Definizione delle classi

I *Progettisti* devono fornire la seguente definizione per ogni componente:

- nome;
- struttura del componente;
- descrizione;
- classi contenute.

2.2.3.2.5 Tracciamento delle componenti

Tutti i requisiti devono essere riferiti al componente che li soddisfa per poter verificare che ogni requisito sia soddisfatto.
Si veda la sezione 2.2.6.4 in cui viene descritto lo strumento utilizzato per il tracciamento e la sezione 2.2.7.12 per le procedure di gestione dei componenti.

2.2.3.2.6 Test di integrazione

Devono essere definite le classi di verifica necessarie a garantire che tutte le componenti del sistema funzionino correttamente.
Si veda la sezione 2.2.6.4 in cui viene descritto lo strumento utilizzato per il tracciamento e la sezione 2.2.7.13 per le procedure di gestione dei test di integrazione.

2.2.4 Progettazione a basso livello

2.2.4.1 Scopo

Definire la struttura di tutte le componenti e suddivederle in unità che possanno essere realizzate, compilate e testate singolarmente. Individuare i test delle unità. Il risultato deve essere presentato nel documento formale *Definizione di prodotto*.

2.2.4.2 Definizione di prodotto

2.2.4.2.1 Diagrammi UML

Devono essere forniti i seguenti diagrammi:

- **diagrammi di classe:** hanno lo scopo di descrivere entità con le loro caratteristiche e relazioni, pertanto vengono definiti attributi, metodi e relazioni. tra di esse. Nella progettazione di dettaglio è necessario che questi diagrammi siano particolarmente dettagliati;
- **diagrammi di attività:** hanno lo scopo di definire le attività da svolgere per realizzare una certa funzionalità. Vengono dunque usati per mostrare come determinate interazioni tra componenti realizzino una funzionalità che si intende rendere disponibile;
- **diagrammi di sequenza:** hanno lo scopo di descrivere le relazioni che intercorrono, in termini di messaggi, tra attori, oggetti ed entità del sistema software rappresentato.

Viene introdotta una notazione che utilizza i colori per distinguere la provenienza delle componenti dell'applicazione:

- giallo: componente da implementare;
- rosso: componente offerta da *RiskApp*;
- verde: componente importati da terze parti.

2.2.4.2.2 Design pattern

Per migliorare la comprensibilità delle scelte progettuali e della progettazione stessa devono essere forniti i *design pattern*$_G$ utilizzati secondo la seguente struttura:

- descrizione testuale;
- descrizione grafica;
- motivazione e descrizione dell'utilizzo all'interno del progetto.

2.2.4.2.3 Classificazione delle classi

Le classi vengono identificate univocamente da una descrizione nella forma:

[NomeProdotto]::[NomeComponente]::[NomeClasse]

dove:

- **NomeProdotto:** è il nome del prodotto software che i *Progettisti* hanno individuato;
- **NomeComponente:** rappresenta il nome assegnato al componente dai *Progettisti*;
- **NomeClasse:** rappresenta il nome che è stato dato dai *Progettisti* alla classe individuata.

2.2.4.2.4 Definizione delle classi

I *Progettisti* devono fornire la seguente definizione per ogni classe:

- nome;
- visibilità;
- attributi;
- metodi;
- descrizione generica che ne spieghi lo scopo e che definisca le funzionalità.

2.2.4.2.5 Tracciamento delle classi

Tutti i requisiti devono essere tracciati alle classi associate per poter verificare che ogni classe soddisfi almeno un requisito. Si veda le sezione 2.2.6.4 in cui viene descritto lo strumento utilizzato per il tracciamento.

2.2.4.2.6 Test di unità

Devono essere definiti dei test di unità necessari a garantire che tutte le componenti del sistema funzionino correttamente.
Si veda le sezione 2.2.6.4 in cui viene descritto lo strumento utilizzato per il tracciamento e la sezione 2.2.7.18 per le procedure di gestione dei test d'unità.

2.2.5 Codifica e test

2.2.5.1 Scopo

Sviluppare le unità ed i test individuati durante la progettazione. Il risultato finale deve essere il codice sorgente del prodotto da realizzare e dei test necessari.

2.2.5.2 Stile di codifica

Al fine di produrre codice uniforme, leggibile e manutenibile è richiesto che vengano rispettate le seguenti convenzioni:

- i nomi utilizzati devono essere chiari, descrittivi rispetto alla loro funzione e in inglese;
- evitare nomi troppo simili tra loro che possano creare difficoltà nella comprensione del codice;
- deve essere presente almeno un breve commento descrittivo per ogni classe e metodo;
- i commenti devono essere scritti in lingua italiana senza utilizzare abbreviazioni o altre ambiguità;
- le modifiche al codice devono sempre riflettersi sui relativi commenti;
- evitare commenti superflui, inappropriati o scurrili;
- ogni file deve presentare un'intestazione con le seguenti informazioni:
  - nome e cognome dell'autore;
  - percorso e nome del file;
  - data di creazione;
  - data ultima modifica;

Il codice deve seguire le linee guida reperibili all'indirizzo:

https://github.com/airbnb/javascript/tree/master/react

Ulteriori convenzioni adottate sono:

- i nomi delle classi devono rispettare lo stile PascalCase;
- i seguenti nomi devono rispettare lo stile CamelCase:
  - metodi;
  - variabili;
  - nomi dei file;
- tutti i campi dati e le variabili devono essere dichiarati pubblici;
- lo stile css deve essere definito *Inline* per ogni componente e deve essere inserito all'interno del costruttore dello stesso;
- per esportare classi, funzioni, oggetti o tipi primitivi da un file (o modulo) si utilizza l'istruzione `export` di JavaScript ES6:

  - per esportare più valori per file come ad esepio oggetti, variabili, funzioni (può essere usato anche più volte nello stesso modulo). Esempio: `export let objectEmpty = {};`
  - per esportare un componente intero (l'istruzione va inserita necessariamente in fondo al file) `export { NomeClasse };`

- per importare funzioni, oggetti o tipi primitivi da un file (o modulo) si utilizza l'istruzione `import` di JavaScript ES6:
  - per importare un singolo oggetto o funzione. Esempio: `import * as stubStrings from '../stubs/stubStrings-spec';` dove `stubStrings` è un alias del nome del file che si desidera importare. Poi all'interno del file si utilizzerà nel seguente modo: `stubStrings.nomeOggetto`
  - per importare una intera classe esportata: Esempio: `import { Asset } from '../../src/DeGeOP/store/process/asset';` Non vanno inserite le estensioni nei nomi dei file.

2.2.5.3 Documentazione del codice

Per rendere maggiormente manutenibile il codice, tutti i *Programmatori* devono documentare ogni componente del codice prodotto (metodo o classe) con la notazione JSDoc3.
Successivamente verrà generato un sito web contenente il riepilogo di ciascun componente codificato. Quindi è necessario attenersi alle seguenti regole, in modo tale da generare una documentazione uniforme e precisa.
Ogni componente deve avere i seguenti tag:

- **@author <name>:** dove al posto di `<name>`, ci sarà il nome e cognome dello sviluppatore che l'ha realizzato. Esempio: `@auhor Nome Cognome`;
- **@description Creazione: <date>:** dove al posto di `<date>` ci sarà la data di creazione del componente, nel formato YYYY-MM-GG. Esempio: `@description Creazione: 2017-03-01`;
- **@description Ultima modifica: <date>:** dove al posto di `<date>` ci sarà la data dell'utima modifica del componente, nel formato YYYY-MM-GG. Esempio: `@description Ultima modifica: 2017-03-01`;
- **@description <desc>:** dove al posto di `<desc>` ci sarà la descrizione concisa del funzionamento e ruolo del componente. Esempio: `@description Descrizione del funzionamento...`.

Inoltre, ogni classe deve avere i seguenti *tag*$_G$:

- **@class <name>:** dove al posto di `<name>`, ci sarà il nome della classe. Esempio: `@class NomeClasse`;
- **@memberof <parentNamepath>:** dove al posto di `<parentNamepath>` ci sarà il percorso del padre della classe. Esempio: `@memberof app::front-end::nomePadre`;

- **@param <type> <name> [<desc>]:** dove al posto di `<{type}>`, `<name>` e `<desc>` ci saranno rispettivamente, tipo dell'attributo, nome, breve descrizione del membro della classe. Esempio: `@param {string} s Descrizione di s...`
- **@constructs <name>:** dove al posto di `<name>`, ci sarà il nome del costruttore. Esempio: `@constructs NomeCostruttore`.

Inoltre ogni metodo deve avere i seguenti tag:

- **@memberof <parentNamepath>:** dove al posto di `<parentNamepath>` ci sarà il nome della classe del metodo. Esempio: `@memberof NomeClasse`;
- **@param <type> <name> [<desc>]:** dove al posto di `<{type}>`, `<name>` e `<desc>` ci saranno rispettivamente, tipo del parametro, nome, breve descrizione. Esempio Esempio: `@param {number} n Descrizione di n...`;
- **@return <type> <desc>:** dove al posto di `<{type}>` e `desc` ci saranno rispettivamente, tipo di ritorno del metodo e breve descrizione. Esempio `@return {string} Questo metodo ritorna una stringa`

I precedenti tag sono i fondamentali richiesti, ma per una maggiore documentazione si rimanda alla sezione 2.2.6.3.

2.2.5.4 Ricorsione

La ricorsione va sempre evitata se possibile. Per ogni funzione ricorsiva è necessario fornire una prova di terminazione nei commenti.

2.2.5.5 Variabili globali

L'uso di variabili globali va sempre evitato se possibile.

2.2.5.6 Classificazione dei test

I test implementati devono essere classificati secondo la seguente notazione:

T[Tipologia Test][Codice identificativo]

dove:

- **tipologia test:** indica il tipo di test e può assumere i seguenti valori:
  - **U:** per i test di unità (vedi sezione 2.2.4.2.6);
  - **I:** per i test di integrazione (vedi sezione 2.2.3.2.6);
  - **S:** per i test di sistema;
- **codice identificativo:** indica il codice numerico del test.

2.2.6 Strumenti

2.2.6.1 Astah

Astah$_G$ è un'applicazione per la creazione di diagrammi UML ed è utilizzata nel corso delle attività di analisi e progettazione. I diagrammi che verranno realizzati sono:

- di sequenza;
- di attività;
- dei casi d'uso;
- delle classi.

Le principali motivazioni che hanno portato il gruppo alla scelta di questo strumento sono:

- utilizzato in aula dal docente;
- versione professional gratuita con la licenza studente;
- *cross-platform*$_G$.

Indirizzi per il download del programma e della licenza studente:

http://astah.net/download
http://astah.net/student-license-request

2.2.6.2 Babel

Babel$_G$ è un compilatore JavaScript che permette di trasformare in maniera automatica codice scritto con lo standard ES2015 in codice compatibile con la maggior parte dei browser. Il suo utilizzo è necessario poiché, ad oggi, lo standard ES2015 non è pienamente supportato da tutti i browser. Babel può essere integrato facilmente in WebStorm.
Per ulteriori informazioni:

https://babeljs.io/

2.2.6.3 JSDoc3

JSDoc3 è un tool che permette la generazione automatica di documentazione per *JavaScript*$_G$, simile a JavaDoc o PHPDoc. JSDoc3 sfrutta delle annotazioni (es: @author) per documentare il codice direttamente nei file sorgenti. La documentazione finale sarà generata sotto forma di sito web. JSDoc3 scansionerà i file javascript interpretando le annotazioni generando così le pagine *HTML*$_G$.
Questo strumento è utilizzato durante l'attività di codifica e le principali motivazioni che hanno portato il gruppo alla sua scelta sono:

- larga diffusione;
- documentazione vasta, professionale e altamente usabile.

L'indirizzo per la documentazione dello strumento è:

http://usejsdoc.org/index.html

2.2.6.4 Trender

Il gruppo utilizza l'applicazione *Trender*$_G$ per gestire i dati ricavati dall'analisi dei requisiti. Trender è stato sviluppato dal gruppo InfiniTech per il progetto dell'anno 2014/15, utilizza un *database*$_G$ *MySQL*$_G$ per la persistenza dei dati ed è disponibile al seguente indirizzo:

`https://github.com/campagna91/Trender`.

Le funzionalità offerte da Trender sono:

- tracciamento dei requisiti;
- tracciamento dei casi d'uso;
- tracciamento dei verbali;
- tracciamento degli attori presenti nel sistema;
- tracciamento dei *packages*$_G$;
- tracciamento delle classi;
- tracciamento dei test;
- tracciamento delle voci del glossario;
- possibilità di creare il codice *LaTeX*$_G$ di quanto archiviato nei punti precedenti;
- statistiche su: requisiti, casi d'uso, componenti, test.

Lo spazio di lavoro dedicato al gruppo si trova al seguente indirizzo:

`http://zephyrusar.altervista.org/trender/`

2.2.6.5 WebStorm

WebStorm$_G$ è un *IDE*$_G$ fornito dall'azienda JetBrains per lo sviluppo di applicazioni e siti web. Questo software permette di supportare lo sviluppatore nella realizzazione di prodotti web. Tra le principali funzionalità sono presenti:

- assistenza nella codifica dei principali framework web;
- auto-completamento per la sintassi di molti linguaggi (es. Javascript, HTML, *CSS*$_G$);
- gestione di progetti complessi;
- integrazione con i più popolari tool per lo sviluppo web;
- integrazione di un sistema di debugging per il codice JavaScript;
- compatibilità con sistemi di build automatico.

Questo strumento è utilizzato durante l'attività di codifica e le principali motivazioni che hanno portato il gruppo alla sua scelta sono:

- strumento completo e professionale;
- licenza gratuita per studenti;

- integrabilità con strumenti per analisi statica e test;
- integrabilità con *GitHub*$_G$;
- cross-platform.

L' indirizzo per il download è:

https://www.jetbrains.com/webstorm/download/

2.2.7 Procedure

2.2.7.1 Configurazione Astah

- **Rimozione lock file:** per evitare errori dovuti alla condivisione dei file Astah nella cartella Dropbox è necessario disabilitare l'impostazione che blocca il file quando è aperto nel seguente modo:
  - aprire Astah;
  - entrare nella sezione del menù `Preferences->File`;
  - rimuovere la spunta su `Lock file when opening`.
- **Qualità immagini esportate:** per impostare la qualità ad almeno 720p, è necessario:
  - aprire Astah;
  - entrare nella sezione del menù `Preferences->Image Export`;
  - settare il campo dati `Resolution to export diagram" a 200`,
  - applicare le modifiche e salvare.

2.2.7.2 Creazione diagrammi UML con Astah

Per la creazione dei diagrammi UML è stata scelta l'applicazione Astah di cui si rimanda alla sezione 2.2.6.1 per il download e l'installazione.
Dopo aver aperto Astah, un diagramma UML deve essere creato rispettando il seguente ordine:

1. selezionare dal menù `Diagram` il tipo di diagramma desiderato;
2. realizzare il diagramma con le convenzioni descritte nella sezione 2.2.2.4;
3. salvarlo nella cartella del documento di riferimento all'interno dello spazio Dropbox `/Diagrammi Astah/`.

2.2.7.3 Linking elementi all'interno di un diagramma Astah

Per facilitare la consultazione dei diagrammi, soprattutto tra i casi d'uso, Astah mette a disposizione la funzionalità di collegare tra loro dei componenti all'interno di un diagramma. Per poter effettuare un link bisogna seguire i seguenti passi:

1. selezionare un elemento all'interno del diagramma;
2. click destro su quest'ultimo e selezionare la voce `Hyperlink -> Edit Hyperlink`;
3. click su `Add Model & Element`;
4. selezionare l'elemento che si vuole collegare.

Per la modifica o la rimozione dei link si segue la stessa procedura fino al punto 2 per poi cliccare su `benvegn` piuttosto che su `Delete`.

2.2.7.4 Aggiunta attore su Trender

L'aggiunta di un attore deve rispettare i seguenti passi:

1. login al sito http://zephyrusar.altervista.org/trender/;
2. cliccare sulla voce del menu: `Actors`;
3. cliccare sul tasto `+` rosso, in basso a destra;
4. compilare i campi: `nome attore, descrizione, padre (opzionale)`;
5. cliccare sul tasto `insert`.

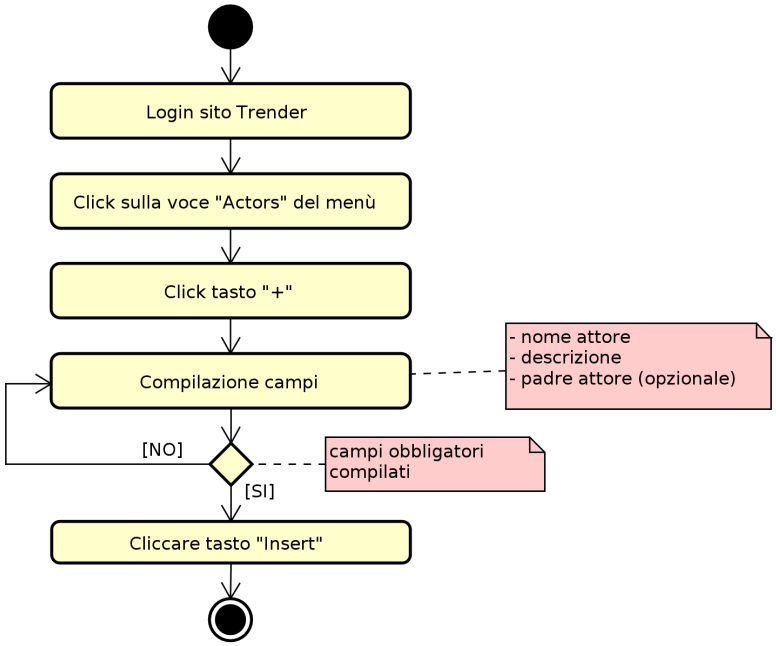

Figura 1: Aggiunta attore.

2.2.7.5 Modifica ed eliminazione attore su Trender

Questa funzionalità non è stata implementata, pertanto bisogna effettuare la modifica dei campi dati dell'attore o l'eliminazione attraverso l'interfaccia di PhpMyAdmin selezionando la tabella: `Actors`.

2.2.7.6 Aggiunta use case su Trender

L'aggiunta di uno use case deve rispettare i seguenti passi:

1. login al sito http://zephyrusar.altervista.org/trender/;
2. cliccare sulla voce del menu: `Usecases`;
3. cliccare sul tasto `+` rosso, in basso a destra;
4. compilare i campi: padre, tipo, titolo, descrizione, pre-condizione, post-condizione, scenario, estensione, path immagine, didascalia immagine;

5. cliccare sul tasto `insert`.

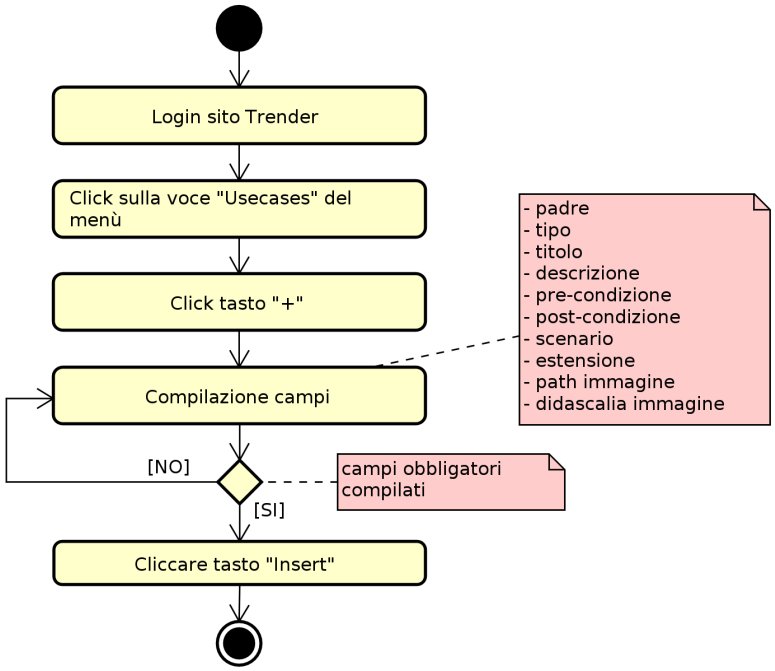

Figura 2: Aggiunta use case.

2.2.7.7 Modifica use case su Trender

La modifica di uno use case deve rispettare i seguenti passi:

1. login al sito http://zephyrusar.altervista.org/trender/;
2. cliccare sulla voce del menu: `Usecases`;
3. cliccare sull' `id` dello use case;
4. modificare i campi: padre, tipo, titolo, descrizione, pre-condizione, post-condizione, scenario, estensione, path immagine, didascalia immagine, associazione use case-attore, associazione use case requisito;
5. cliccare sul tasto `save`.

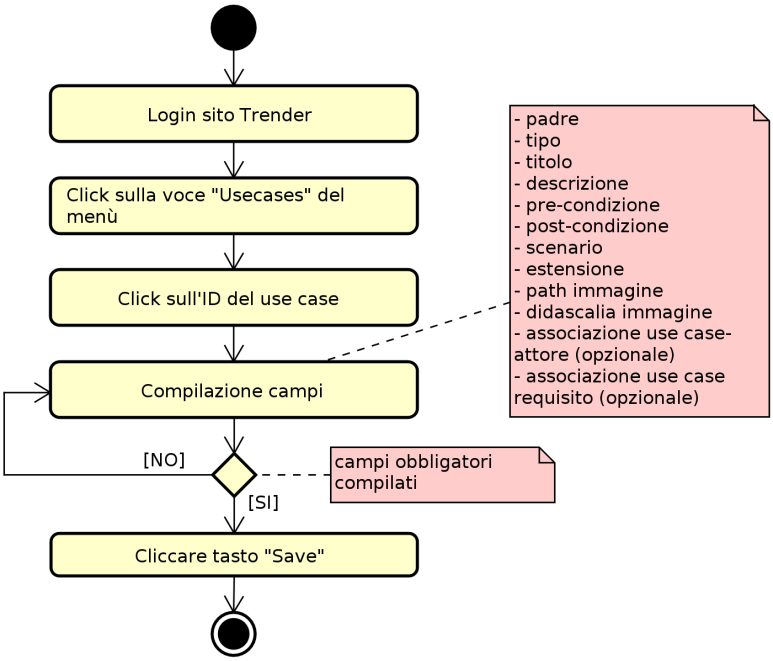

Figura 3: Modifica use case.

2.2.7.8 Eliminazione use case su Trender

L'eliminazione di uno use case deve rispettare i seguenti passi:

1. login al sito http://zephyrusar.altervista.org/trender/;
2. cliccare sulla voce del menu: `Usecases`;
3. cliccare sull' `id` dello use case;
4. cliccare sul tasto `delete use case`.

2.2.7.9 Aggiunta requisito su Trender

L'aggiunta di un requisito deve rispettare i seguenti passi:

1. login al sito http://zephyrusar.altervista.org/trender/;
2. cliccare sulla voce del menu: `Requirements`;
3. cliccare sul tasto `+` rosso, in basso a destra;
4. compilare i campi: padre, importanza(obbligatorio, desiderabile, facoltativo), tipo(funzionale, prestazionale, qualitativo, vincolo), sorgente, descrizione;
5. cliccare sul tasto `insert`.

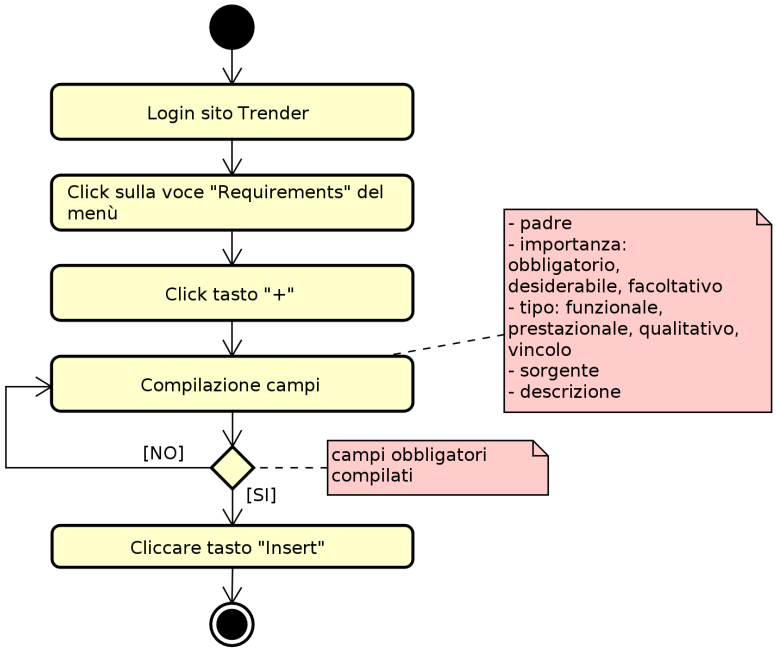

Figura 4: Aggiunta requisito.

2.2.7.10 Modifica requisito su Trender

La modifica di un requisito deve rispettare i seguenti passi:

1. login al sito http://zephyrusar.altervista.org/trender/;
2. cliccare sulla voce del menu: `Requirements`;
3. cliccare sull' `id` del requisito;
4. modificare i campi: padre, importanza(obbligatorio, desiderabile, facoltativo), tipo(funzionale, prestazionale, qualitativo, vincolo), sorgente, descrizione, associazione classe (opzionale), package (opzionale), test di sistema (opzionale), test di validazione (opzionale);
5. cliccare sul tasto `save`.

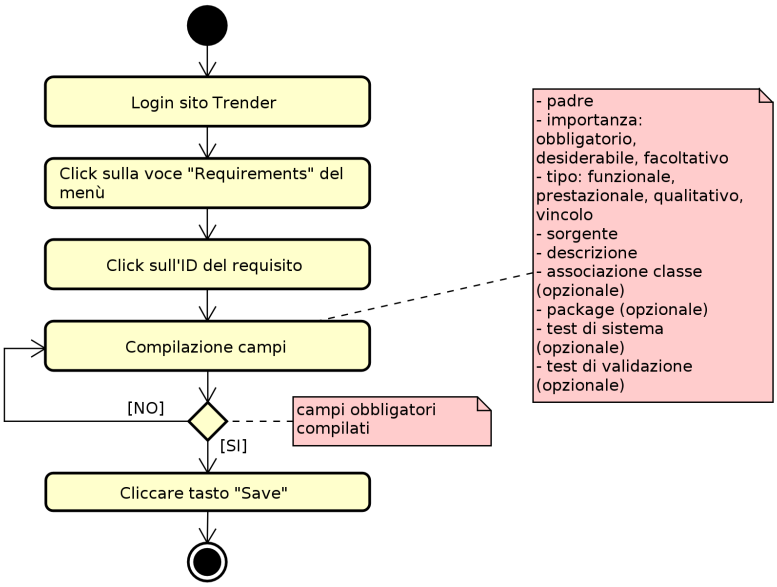

Figura 5: Modifica requisito.

2.2.7.11 Eliminazione requisito su Trender

L'eliminazione di un requisito deve rispettare i seguenti passi:

1. login al sito http://zephyrusar.altervista.org/trender/;
2. cliccare sulla voce del menu: `Requirements`;
3. cliccare sull' `id` del requisito;
4. cliccare sul tasto `delete requirement`.

2.2.7.12 Aggiunta package su Trender

L'aggiunta di un package deve rispettare i seguenti passi:

1. login al sito http://zephyrusar.altervista.org/trender/;
2. cliccare sulla voce del menu: `Packages`;
3. cliccare sul tasto `+` rosso, in basso a destra;
4. compilare i campi: nome, descrizione, path immagine, didascalia, padre (opzionale);
5. cliccare sul tasto `insert`.

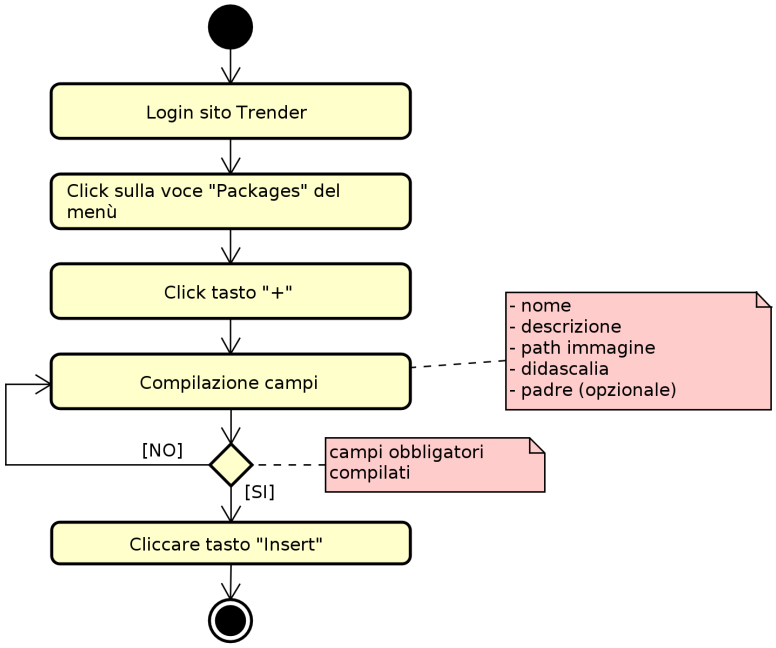

Figura 6: Aggiunta package.

2.2.7.13 Modifica package su Trender

La modifica di un package deve rispettare i seguenti passi:

1. login al sito http://zephyrusar.altervista.org/trender/;
2. cliccare sulla voce del menu: `Packages`;
3. cliccare sull' `id` del packages;
4. modificare i campi: nome, descrizione, path immagine, didascalia, padre (opzionale), test d'integrazione;
5. cliccare sul tasto `save`.

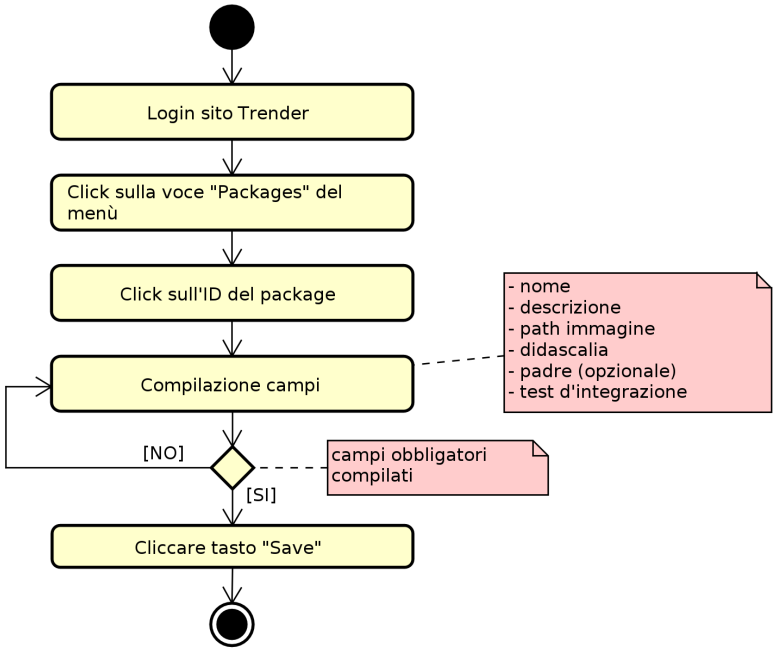

Figura 7: Modifica package.

2.2.7.14 Eliminazione package su Trender

L'eliminazione di un package deve rispettare i seguenti passi:

1. login al sito http://zephyrusar.altervista.org/trender/;
2. cliccare sulla voce del menu: `Packages`;
3. cliccare sull' `id` del package;
4. cliccare sul tasto `delete package`.

2.2.7.15 Aggiunta classe su Trender

L'aggiunta di una classe deve rispettare i seguenti passi:

1. login al sito http://zephyrusar.altervista.org/trender/;
2. cliccare sulla voce del menu: `Classes`;
3. cliccare sul tasto `+` rosso, in basso a destra;
4. compilare i campi: selezione package, nome classe, descrizione, utilizzo;
5. cliccare sul tasto `insert`.

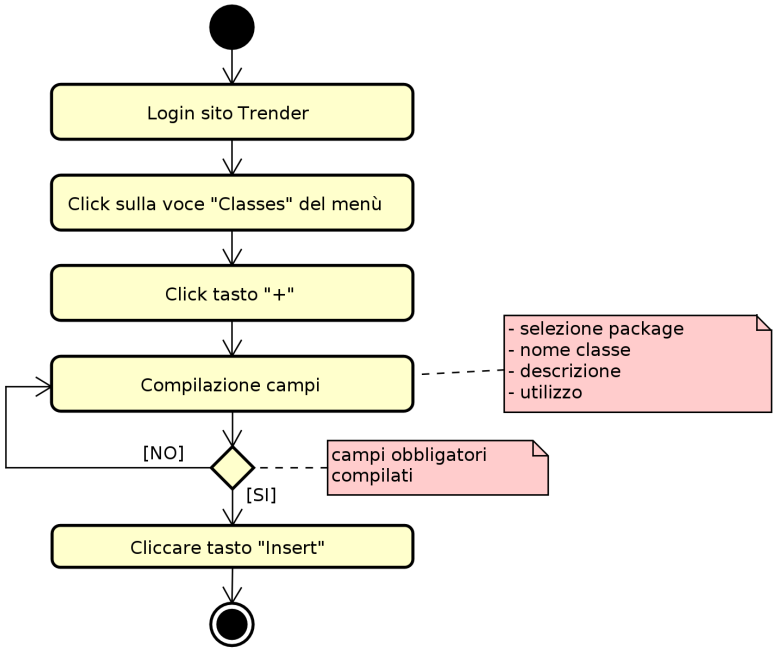

Figura 8: Aggiunta classe.

2.2.7.16 Modifica classe su Trender

La modifica di una classe deve rispettare i seguenti passi:

1. login al sito http://zephyrusar.altervista.org/trender/;
2. cliccare sulla voce del menu: `Classes`;
3. cliccare sull' `id` della classe;
4. modificare i campi: selezione package, nome classe, descrizione, utilizzo, classe base, classe target, tipo relazione, attributi (tipo, nome, descrizione), metodi (tipo di ritorno, segnatura, descrizione, parametri (tipo parametro, nome, descrizione);
5. cliccare sul tasto `save`.

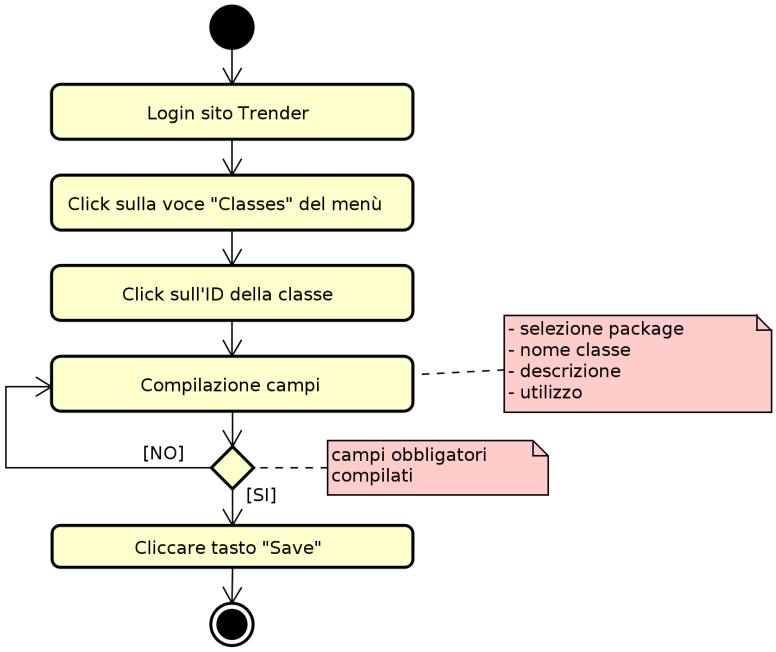

Figura 9: Modifica classe.

2.2.7.17 Eliminazione classe su Trender

L'eliminazione di una classe deve rispettare i seguenti passi:

1. login al sito http://zephyrusar.altervista.org/trender/;
2. cliccare sulla voce del menu: `Classes`;
3. cliccare sull' `id` della classe;
4. cliccare sul tasto `delete class`.

2.2.7.18 Aggiunta test di unità su Trender

L'aggiunta di un test di unità deve rispettare i seguenti passi:

1. login al sito http://zephyrusar.altervista.org/trender/;
2. cliccare sulla voce del menu: `Test`;
3. cliccare sul tasto `+` rosso, in basso a destra;
4. compilare i campi: package, classe, metodo da combinare, descrizione del test;
5. cliccare sul tasto `insert`.

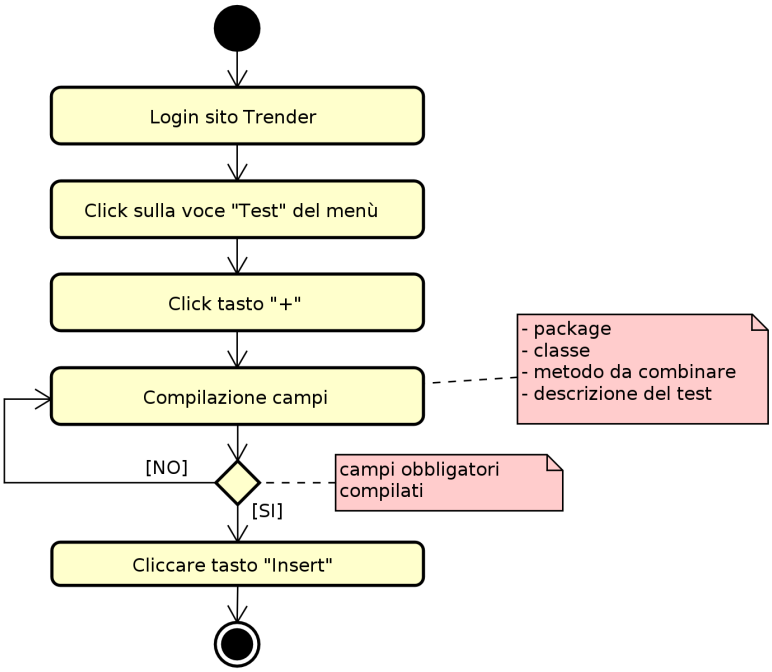

Figura 10: Aggiunta test di unità.

2.2.7.19 Modifica ed eliminazione Test di unità su Trender

Questa funzionalità non è stata implementata, pertanto bisogna effettuare la modifica dei campi dati del test o l'eliminazione attraverso l'interfaccia di PhpMyAdmin selezionando la tabella: `UnitTest`.

2.2.7.20 Esportazione da Trender a codice LaTeX

L'esportazione del codice LaTeX da Trender deve rispettare i seguenti passi:

1. login al sito http://zephyrusar.altervista.org/trender/;
2. cliccare sulla voce del menu: `Prints`;
3. selezionare i campi desiderati per la stampa (di solito tutti);
4. cliccare sul tasto `print all`;
5. selezionare la sezione del codice LaTeX da visualizzare;
6. copiare il codice LaTeXe incollarlo sul documento desiderato.

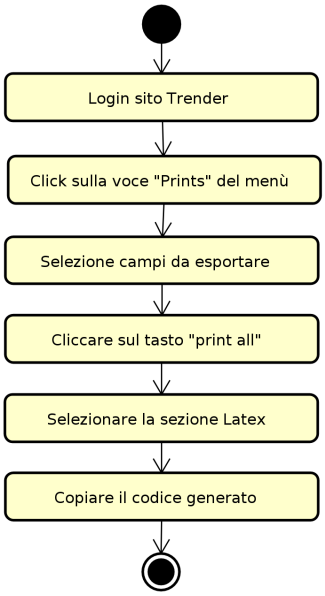

Figura 11: Esportazione codice LaTeX da Trender.

2.2.7.21 Impostazioni Trender

Alcune impostazioni di Trender si possono modificare rispettando i seguenti passi:

1. login al sito http://zephyrusar.altervista.org/trender/;
2. cliccare sulla voce del menu a forma di ingranaggio;
3. inserire la sorgente dei requisiti: nome sorgente.

2.2.7.22 Configurazione WebStorm

Lo stile di codifica adottato (vedi sezione 2.2.5.2) e personalizzato per questo progetto deve essere importato all'interno di WebStorm nel tool ESLint. Per fare questo è necessario seguire i seguenti passaggi:

1. aprire WebStorm;
2. aprire il menu `File -> Preferences`;
3. entrare nel menu `Language & Frameworks->JavaScript->Code Quality Tools->ESLint`;
4. abilitare il tool essicurandosi di aver installato prima Node.js (vedi sezione 4.2.6.2);
5. selezionare nel campo ESLint package il file che si trova all'interno del progetto `DeGeOp/node_modules/eslint`;
6. salvare le impostazioni premendo il tasto `Ok`.

3 Processi di supporto

3.1 Documentazione

3.1.1 Scopo

Lo scopo del processo di documentazione è di redigere e mantenere la documentazione durante l'intero *ciclo di vita*$_G$ del software. La corretta implementazione del processo deve:

- dare una chiara visione dei documenti che devono essere prodotti durante il ciclo di vita del software;
- fornire le norme necessarie alla stesura dei documenti;
- produrre documenti formali coerenti.

3.1.2 Template

È stato creato un template LaTeX$_G$ per garantire che tutti i documenti creati dal *gruppo*$_G$ abbiano la stessa struttura grafica e lo stesso stile di formattazione. Ogni membro del gruppo deve creare i documenti richiesti utilizzando tale template e deve ridurre al minimo indispensabile eventuali variazioni personali nella formattazione.

3.1.3 Struttura dei documenti

3.1.3.1 Prima pagina

La prima pagina di ogni documento deve presentare la seguente struttura:

- logo del gruppo;
- titolo del documento;
- informazioni del documento:
  - versione;
  - data di creazione;
  - data di ultima modifica;
  - stato del documento;
  - redattori del documento;
  - verificatori;
  - responsabile approvazione;
  - lista di distribuzione;
  - email del gruppo.

3.1.3.2 Registro delle modifiche

La seconda pagina di ogni documento formale deve contenere una tabella con la lista delle modifiche apportate al documento. Ogni riga della tabella deve essere compilata per intero con le seguenti informazioni:

- versione del documento successiva alla modifica;

- data in cui è stata eseguita la modifica;
- autore della modifica;
- ruolo dell'autore;
- descrizione concisa della modifica apportata. Inoltre se quest'ultima riguarda una specifica correzione o decisione, inserire prima del testo della modifica:
  - **<Correzione post codice_revisione>:** se riferita a errori segnalati dal docente; dove per codice_revisione s'intende: RR, RP, RQ o RA.
  - **<ID_Decisione>:** se riferita a una decisione presa in un verbale.

3.1.3.3 Indice

Nella pagina successiva alla fine del registro delle modifiche ogni documento formale deve contenere l'indice dei suoi contenuti; tale indice deve permettere la lettura *ipertestuale*$_G$ del documento. L'indice deve essere numerato a partire da 1, ogni sottosezione riparte da 1 e aggiunge il proprio indice a quello del padre separandolo con un punto. Eventuali appendici invece di essere numerate saranno indicate con lettere maiuscole a partire da A seguendo l'ordine alfabetico internazionale.

3.1.3.4 Note a piè di pagina

Eventuali note vanno indicate in basso a sinistra nella pagina in cui compaiono con il loro numero identificativo e la loro descrizione.

3.1.3.5 Contenuto principale

Tutte le pagine successive all'indice del documento devono contenere un'intestazione e un piè di pagina. L'intestazione deve contenere:

- nome della sezione allineato a sinistra;
- nome del documento allineato a destra.

Il piè di pagina deve contenere:

- nome del gruppo e del progetto allineati a sinistra;
- pagina corrente rispetto alle pagine totali del documento allineate a destra.

3.1.4 Norme tipografiche

3.1.4.1 Glossario

Ogni parola contenuta nel *Glossario* deve essere scritta in corsivo e contrassegnata da una "G" a pedice come da esempio:

termine$_G$

3.1.4.2 Stile del testo

Per facilitare la stesura del documento, migliorarne correttezza e leggibilità, TexStudio mette a disposizione dei tool per il controllo grammaticale. Ogni membro del gruppo deve controllare nelle impostazioni del proprio strumento che siano attivati:

- **individua ripetizioni:** durante la scrittura del documento, se una parola viene ripetuta troppo viene sottolineata di verde;
- **individua errori ortografici:** gli errori vengono sottolineati di rosso;
- **lingua predefinita:** italiano.

Le seguenti convenzioni devono essere rispettate nella stesura dei documenti:

- **grassetto:** deve essere utilizzato nei seguenti casi:
  - titoli di sezioni e paragrafi;
  - termini di elenchi puntati per i quali si fornisce una descrizione;
  - riferimenti alle revisioni di avanzamento.
- **corsivo:** deve essere utilizzato nei seguenti casi:
  - nome del gruppo;
  - nome del proponente;
  - nome del progetto;
  - citazioni;
  - abbreviazioni;
  - parole presenti nel glossario;
  - ruoli del progetto;
  - nomi dei documenti.
- **monospace:** deve essere utilizzato nei seguenti casi:
  - nomi di file;
  - codice di programmazione;
  - indirizzi email.
- **maiuscolo:** le uniche parole che possono essere scritte interamente a caratteri maiuscoli sono gli acronimi e le sigle.

3.1.4.3 Titoli

I titoli delle sezioni e dei paragrafi vanno scritti con solo la prima lettera maiuscola a meno di nomi propri e di termini indicati nella sezione 3.1.4.5.

3.1.4.4 Elenchi puntati

Le seguenti convenzioni devono essere rispettate nella creazione di elenchi puntati:

- ogni elemento dell'elenco deve iniziare con la lettera minuscola a meno che non sia un nome proprio;
- ogni elemento dell'elenco tranne l'ultimo deve terminare con un punto e virgola;
- l'ultimo elemento dell'elenco deve terminare con il punto.

3.1.4.5 Formati

I seguenti formalismi devono essere utilizzati durante la stesura dei documenti:

- **date:** le date presenti nei documenti devono seguire lo standard *ISO*$_G$ 8601:2004 (vedi riferimento 1.4.1):

  YYYY-MM-GG

  dove:

  - **YYYY:** rappresenta l'anno espresso con quattro cifre;
  - **MM:** rappresenta il mese espresso con due cifre;
  - **GG:** rappresenta il giorno espresso con due cifre.

  È possibile utilizzare il comando LaTeX `\frmdata{GG}{MM}{YYYY}` per la formattazione delle date.

- **orari:** gli orari presenti nei documenti devono seguire lo standard ISO 8601:2004 (vedi riferimento 1.4.1):

  hh:mm

  dove:

  - **hh:** rappresentano le ore espresse con due cifre da 00 a 23;
  - **mm:** rappresentano i minuti espressi con due cifre da 00 a 59.

  È possibile utilizzare il comando LaTeX `\frmora{hh}{mm}` per la formattazione delle ore.

- **valute:** le valute presenti nei documenti devono essere scritte utilizzando il simbolo della valuta usata seguito dal numero. Le cifre decimali devono essere separate dalla virgola, le cifre non decimali devono essere separate da un punto a gruppi di tre:

  [Simbolo valuta] 1.234.567,89

  Ad esempio: € 3.869,25

- **nomi ricorrenti:** I seguenti termini ricorrenti vanno sempre inseriti con i relativi comandi LaTeX forniti dal template (vedi appendice B) per garantire omogeneità in tutti i documenti:

  - nome del gruppo;
  - email di riferimento del gruppo;
  - nome del proponente;
  - nome del progetto svolto;
  - ruoli di progetto;
  - nomi dei documenti senza versione;
  - nomi dei documenti con versione;
  - revisioni di avanzamento del progetto;
  - nomi di strumenti o tecnologie.

3.1.4.6 Nomi propri

I nomi propri di persona devono essere scritti come nome e cognome.

3.1.5 Componenti grafiche

3.1.5.1 Tabelle

Tutte le tabelle devono avere un indice numerico univoco che le identifichi all'interno del documento ed una breve didascalia. Le tabelle devono essere centrate orizzontalmente.

- Le tabelle semplici vengono riportate con il comando:

```
\begin{tabular}[H]
...
\end{tabular}
```

dove:

  - **[H]:** è una preferenza di collocazione per oggetti mobili e sta a indicare di inserire la tabella esattamente nel punto dove si vuole che compaia. Per altri parametri quali: t,b,p,! si veda la guida LaTeXnella sezione 4.3.1.1.

- Le tabelle su più pagine vengono riportate con il comando:

```
\begin{longtable}
\endfirsthead
\endhead
...
\end{longtable}
```

dove:

  - **\endfirsthead:** specifica l'intestazione della tabella nella prima pagina in cui compare;
  - **\endhead:** specifica l'intestazione della tabella dalla seconda pagina in poi.

- Le tabelle multiriga e multicolonna vengono riportate con il comando:

```
\begin{tabular}[H]{lccc}
\toprule
\multirow {2}*{Elemento} &
\multicolumn{3}{c}{Strati} \\
\cmidrule(lr){2-4}
             & K & L & M \\
\midrule
idrogeno     & $1$ &  &  \\
litio        & $2$ & $1$ &  \\
sodio        & $2$ & $8$ & $1$  \\
\bottomrule
\end{tabular}
```

che produce la seguente tabella d'esempio:

Elemento	Strati		
	K	L	M
idrogeno	1		
litio	2	1	
sodio	2	8	1

3.1.5.2 Immagini

Tutte le immagini inserite all'interno di un documento devono avere ampi margini orizzontali che le separino in modo netto dai paragrafi precedenti e successivi per migliorare la leggibilità. Le immagini devono essere centrate orizzontalmente e devono avere larghezza fissa. I diagrammi *UML*$_G$ devono essere inseriti nei documenti come immagini.

- Le immagini vengono inserite con il seguente comando:

```
\begin{figure}[H]
\includegraphics[width=0.8\textwidth]{img/nomeImmagine}
\caption{Descrizione \ref{sec:ciclodivitadoc}}
\label{fig:ciclovitadoc}
\end{figure}\mbox{}\\
```

  - **[H]:** come descritto sopra per le tabelle;
  - **\caption:** specifica la didascalia dell'immagine;
  - **[width=0.8\textwidth]:** specifica la dimensione dell'immagine, che in questo esempio ricopre l'80% della larghezza del corpo del documento. Si può esprimere con dei valori che vanno da 0 a 1.

3.1.6 Classificazione dei documenti

3.1.6.1 Documenti informali

Tutti i documenti non ancora approvati dal responsabile di progetto sono da ritenersi informali e pertanto ad uso unicamente interno.

3.1.6.2 Documenti formali

Un documento diventa formale in seguito all'approvazione da parte del *Responsabile di progetto*. Solo i documenti formali possono essere distribuiti all'esterno del gruppo. Prima di poter essere approvato un documento deve essere verificato come descritto nella sezione 3.1.8.2 e secondo le procedure descritte nella sezione 3.3.

3.1.6.3 Glossario

Il *Glossario* ha lo scopo di chiarire il significato di alcuni termini calati in un determinato contesto. Al suo interno ci sono parole presenti nei documenti e per rientrarci devono avere le seguenti caratteristiche:

- trattare argomenti tecnici;
- essere sigle;
- descrivere argomenti sconosciuti o ambigui;
- essere strumenti utilizzati nel progetto.

Ogni termine deve essere accompagnato dalla sua spiegazione chiara, concisa e meno ambigua possibile, lunga al massimo dieci righe. Siccome l'inserimento di un termine a glossario può avvenire durante la stesura di un documento, per evitare confusione, si può anche inserire solamente il termine senza la spiegazione (come descritto nella sezione 3.1.8.3). Si deve comunque completare la spiegazione il prima possibile.

3.1.6.4 Verbali

Per ogni incontro deve essere nominato un segretario che si occuperà della stesura di un verbale. Il verbale deve contenere i seguenti punti:

- **estremi della riunione:**
  - data;
  - ora inizio;
  - ora fine;
  - luogo dove si è svolto l'incontro;
  - lista dei partecipanti;
  - lista degli assenti con eventuali motivazioni;
  - nome del segretario.
- **ordine del giorno:** elenco degli argomenti che saranno discussi;
- **corpo del verbale:** verbale dell'incontro;
- **decisioni prese:** elenco delle decisioni prese identificate in modo univoco.

Una volta approvato dal *Responsabile di progetto* il verbale deve essere distribuito a tutti i componenti del gruppo; I varbali esterni dovranno essere inviati anche al proponente. I nomi dei file dei verbali devono rispettare il seguente formato:

`Verbale[Tipologia]_[ID]_[Data riunione].pdf`

dove:

- **tipologia:** `Esterno` o `Interno`;
- **ID:** identificativo numerico, si distingue tra verbali interni ed esterni, parte da 1;
- **data riunione:** data in cui si è svolta la riunione in formato YYYYMMGG.

I nomi delle decisioni devono rispettare il seguente formato:

`V[ID tipologia]_[ID].[Numero decisione]`

dove:

- **ID Tipologia:** `I` o `E` che significano rispettivamente Interno o Esterno;
- **ID:** identificativo numerico, si distingue tra verbali interni ed esterni, parte da 1;
- **Numero decisione:** numero crescente univoco per quantificare la decisione.

Per la redazione dell'elenco delle decisioni è necessario utilizzare i comandi LaTeX

- \ `itemVE` - per i verbali esterni;
- \ `itemVI` - per quelli interni.

3.1.7 Strumenti

3.1.7.1 Latex

Per la stesura dei documenti è stato scelto il *linguaggio di markup*$_G$ LaTeX. Questo permette di preparare documenti formali divisi in sezioni facilitando la collaborazione tra più editori, di separare il contenuto dalla formattazione grafica e di gestire in maniera automatica vari elementi del testo.
Avendo un elevato livello di personalizzazione e automatizzazione, si riesce efficacemente a gestire:

- indici;
- numerazioni di paragrafi, sezioni e pagine
- tabelle complesse;
- formule matematiche;
- riferimenti;
- immagini;
- template di documento.

Le principali motivazioni che hanno portato il gruppo alla scelta di questo strumento sono:

- elevato livello di automatizzazione e personalizzazione;
- possibilità di creare comandi specifici;
- alta professionalità;

La distribuzione di LaTeX consigliata ai membri del gruppo è: **TeX Live**. Le istruzioni per l'installazione di questa o altre distribuzioni sono disponibili al seguente indirizzo:

https://www.latex-project.org/get/

3.1.7.2 TeXstudio

TeXstudio$_G$ è un editor per la creazione di documenti in LaTeX. Questo software offre un ambiente di lavoro completo per aiutare la stesura dei documenti. Inoltre integra un compilatore e visualizzatore PDF per il documento prodotto. Tra le principali funzionalità sono presenti:

- evidenziazione della sintassi;
- strumenti per il controllo ortografico;
- completamento automatico.

Questo strumento è utilizzato durante le attività di analisi e progettazione per redigere i documenti necessari. La versione in uso è la 2.11.0 o superiore.
Le principali motivazioni che hanno portato il gruppo alla scelta di questo strumento sono:

- gratuito;
- *cross-platform*$_G$;
- già conosciuto da alcuni membri del gruppo.

Indirizzo per il download:

http://www.texstudio.org

3.1.8 Procedure

3.1.8.1 Creazione di un documento

Viene fornita nel *repository*$_G$ Documenti, la cartella `Modello nuovo documento` contenente la struttura di base che ogni documento deve avere. Al suo interno sono presenti:

- **il file principale del documento:** `ModelloNuovoDocumento.tex`;
- **la cartella che contiene i file dei capitoli:** `sections`;
- **la cartella che contiene le immagini:** `img`.

Per creare un nuovo documento è sufficiente copiare la cartella descritta sopra e inserirla nella posizione desiderata; facendo le opportune modifiche ai nomi dei file e all'intestazione del documento.
Per un corretto funzionamento del glossario e del sistema di linking dei file LaTeX si deve prima compilare il glossario e successivamente il documento appena creato. Ciò va fatto solo la prima volta che si compila il documento.

3.1.8.2 Ciclo di vita dei documenti

Tutti i documenti tranne i verbali devono seguire il ciclo di vita descritto di seguito (vedi anche figura 12):

1. al termine della stesura di un documento, i redattori ne richiedono la verifica al *Responsabile*;
2. il *Responsabile* assegna la verifica ad un *Verificatore*;
3. il *Verificatore* segnala eventuali modifiche o correzioni da eseguire ai redattori;
4. quando il *Verificatore* ritiene che il documento sia pronto per l'approvazione la richiede al *Responsabile*;
5. il *Responsabile* approva il documento rendendolo formale o lo rifiuta fornendo le motivazioni.

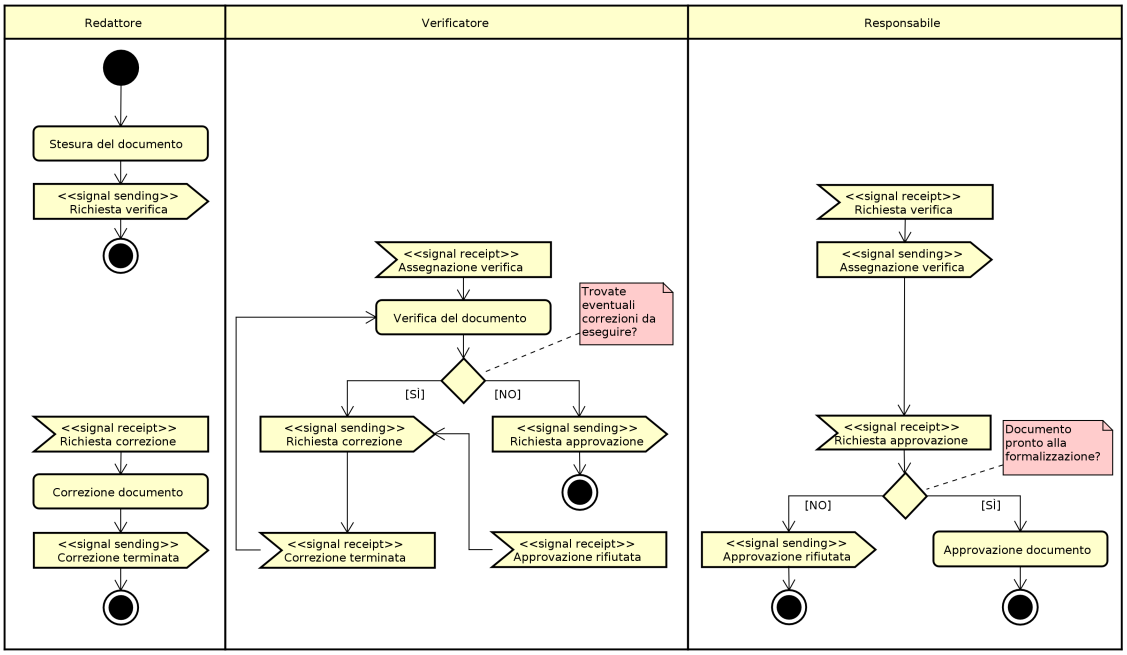

Figura 12: Ciclo di vita di un documento. Riferita nella sezione 3.1.8.2

3.1.8.3 Aggiunta nuovi termini a glossario

Durante la stesura dei documenti, quando si ritiene che un termine debba essere inserito a glossario si deve seguire la seguente procedura:

1. entrare nello spazio condiviso del gruppo su *Google Drive*$_G$ (si veda sezione 4.1.5.3) e aprire il foglio `Termini da inserire nel glossario`;
2. inserire il termine in fondo alla pagina.

3.1.8.4 Inserimento termini a glossario nei Manuali

I manuali Utente e Manutentore hanno dei glossari indipendenti, quindi non godono della stessa procedura applicata per l'inserimento dei termini a glossario degli altri documenti. È stato scelto di usare il pacchetto LaTeX `{glossaries}` per questo tipo di glossari. Per il suo corretto funzionamento, questo comando, necessita di compilazioni separate rispetto a quelle effettuate da TeXstudio. Ulteriori informazioni riguardo a questo pacchetto si possono trovare alla pagina:

`https://it.sharelatex.com/learn/Glossaries`

Per automatizzare questa procedura è stato creato uno script `./mkglo.sh` da utilizzare solo quando si inseriscono nuovi termini nei manuali.

Di seguito è descritta la procedura da seguire **ogni volta** che si desidera inserire un termine a glossario:

1. inserire nel testo il comando `\gls{NomeTermine}`;
2. aprire il file LaTeX `GlossarioManUtente.tex` oppure `GlossarioManMan.tex` contenente la definizione dei termini a glossario;

3. per ogni termine inserire la seguente struttura:

```
\newglossaryentry{NomeTermine}
{
name=NomeTermine,
first=\gloss{\glslink{NomeTermine}{NomeTermine}},
sort=NomeTermine,
description={Descrizione}
}
```

4. aprire un terminale e posizionarsi nella cartella contenente il manuale;
5. (opzionale se non si hanno i permessi giusti per lo script) digitare il comando `chmod +x mkglo.sh`;
6. digitare il comando `./ mkglo.sh NomeManuale.tex`.

3.2 Gestione della configurazione

3.2.1 Scopo

Lo scopo della gestione della configurazione è di individuare e gestire le parti che compongono i prodotti da realizzare (ovvero i Configuration Item, CI). La corretta implementazione del processo deve:

- individuare i CI;
- gestire la loro organizzazione all'interno del repository;
- gestire il versionamento.

3.2.2 Repository

3.2.2.1 Struttura dei repository

Il gruppo ha scelto di utilizzare *GitHub*$_G$ per il versionamento e il salvataggio dei file inerenti le attività di progetto. L'*Amministratore* di progetto deve creare e organizzare i repository necessari e assicurarsi che tutti i membri del gruppo vi possano accedere. Ogni membro deve essere registrato su GitHub e aver attivato l'account studente.

Il repository dedicato ai documenti si trova al seguente indirizzo:

https://github.com/JordanGottardo/Documenti

Il repository dedicato al codice si trova al seguente indirizzo:

https://github.com/JordanGottardo/DeGeOp

Le cartelle nel repository vengono organizzate nel seguente modo a partire dalla root:

- **documenti:** sono presenti le cartelle per ogni revisione del progetto:
  - **01-RR**: contenente i documenti e i file le necessari alla **Revisione dei requisiti**;

  - **02-RP**: contenente i documenti e i file le necessari alla **Revisione di progettazione**;
  - **03-RQ**: contenente i documenti e i file le necessari alla **Revisione di qualifica**;
  - **04-RA**: contenente i documenti e i file le necessari alla **Revisione di accettazione**;
  - **Script**: contenente script e altri strumenti utili per la stesura e la verifica dei documenti;
  - **Modello nuovo documento:** contenente i file con la struttura di base la creazione di un nuovo documento;
  - **Template**: contenente i file del template da usare per la creazione dei documenti LaTeX.
  - **Consegne**: contenente i file delle consegne suddivise per revisione.

- **codice**: sono presenti le cartelle riguardanti la codifica dell'applicazione:
  - **dist:** contentente i file dell'elaborazione della cartella src;
  - **node_modules:** contenente i file di configurazione dei moduli utilizzati nel progetto;
  - **src/DeGeOP:**
    * **action:** contenente i file riguardanti le actions;
    * **reducer:** contenente i file riguardanti i reducers;
    * **store:** contenente i file riguardanti lo store;
    * **view:** contenente i file riguardanti la view.

Ogni componente React è necessario che sia contenuto in un unico file JavaScript.

3.2.2.2 Nomi dei file

I nomi dei documenti presenti nel repository devono rispettare la notazione *camel case*$_G$ con le seguenti caratteristiche:

- la prima lettera di ogni file è minuscola fino al presentarsi della parola successiva che inizia con la lettera maiuscola e cosi via;
- è possibile utilizzare unicamente caratteri alfanumerici e il trattino basso;
- i nomi non possono contenere spazi o elementi di punteggiatura.

3.2.3 Versionamento

3.2.3.1 Controllo di versione

Il controllo di versione del file sorgente, sia per il codice sia per i documenti, viene fatto utilizzando il software *Git*$_G$ e la piattaforma GitHub.

3.2.3.2 Versionamento documenti

Tutti i documenti devono essere identificati da una versione, ad ogni nuova versione deve corrispondere una riga nel registro delle modifiche. La versione corrente di un documento deve sempre essere riportata all'interno dello stesso e va inoltre indicata in coda al nome del file con il seguente formato:

`NomeDocumento_vX.Y.Z.pdf`

- **X:**
  - inizia da 0;
  - viene incrementato solo dal *Responsabile di progetto*, al momento della sua approvazione;
  - non può essere maggiore del numero di revisioni.
- **Y:**
  - inizia da 0;
  - viene incrementato solo dai *Verificatori* ad ogni verifica eseguita;
  - quando viene incrementato X, viene riportato a 0.
- **Z:**
  - inizia da 0;
  - viene incrementato solo dai redattori al completamento di ogni task di modifica del documento;
  - quando vengono incrementati X o Y, viene riportato a 0.

3.2.3.3 Versionamento applicazione

L'applicazione realizzata verrà versionata nel seguente modo:

- **X:**
  - inizia da 0;
  - viene incrementato solo dal *Responsabile di progetto*, corrisponde all'ultima versione stabile del applicazione;
  - la versione 1 coinciderà con la revisione di accettazione.
- **Y:**
  - inizia da 0;
  - viene incrementato solo dai *Verificatori* ad ogni incremento di funzionalità eseguito;
  - quando viene incrementato X, viene riportato a 0.
- **Z:**
  - inizia da 0;
  - viene incrementato solo dai *Programmatori* al completamento di ogni task di modifica;
  - quando vengono incrementati X o Y, viene riportato a 0.

3.2.3.4 Messaggi di commit

Ogni volta che si effettuano modifiche sui file del repository locale per poi esser caricate in quello remoto, bisogna specificarne le motivazioni. Per uniformare l'ambiente di lavoro è stato scelto un formato standard per la scrittura dei messaggi di *commit*$_G$ che devono contenere:

- breve messaggio riassuntivo delle operazioni svolte;
- lista dei file modificati;
- lista delle modifiche per i singoli file.

Si veda la sezione 3.2.5.2 per il dettaglio sulla procedura da seguire.

3.2.4 Strumenti

3.2.4.1 Git

Git è un *sistema software di controllo di versione*$_G$ distribuito e *open source*$_G$. La versione utilizzata al momento della stesura di questo documento è la 2.11.0 o superiore. Le principali motivazioni che hanno portato il gruppo alla scelta di questo strumento sono:

- ampio uso in ambito lavorativo;
- performance superiori rispetto ad altri sistemi di versionamento;
- sistema distribuito anziché centralizzato.

Indirizzo per il download e documentazione:

`https://git-scm.com`

L'utilizzo di Git verrà effettuato tramite riga di comando. Si lascia libertà ai membri del gruppo per l'installazione di eventuali interfacce grafiche personalizzate.

3.2.4.2 GitHub

GitHub è un servizio web di hosting per lo sviluppo di progetti software. Tra le caratteristiche principali:

- utilizzo del sistema di controllo di versione Git;
- possibilità di inserire documentazione e immagini, oltre al codice sorgente ;
- *issue*$_G$ tracking;
- funzionalità simili ai social network come follower, commenti e notifiche;
- visione di grafici e statistiche su sviluppatori e repository.

Le principali motivazioni che hanno portato il gruppo alla scelta di questo strumento sono:

- possibilità di creare repository private attivando il piano studente con l'email universitaria;
- sistema largamente diffuso e conosciuto da alcuni membri del gruppo;
- integrabile con altre applicazioni (es: *Slack*$_G$, *IDE*$_G$).

3.2.5 Procedure

3.2.5.1 Installazione e configurazione di Git

L'installazione di Git varia a seconda del sistema operativo utilizzato. Di seguito verranno elencate le principali procedure d'installazione.

Per i sistemi *Linux*$_G$:

- aprire il terminale;
- eseguire il comando sudo `apt-get update`;
- eseguire il comando `apt-get install git`.

Per i sistemi *Windows*$_G$:

- accedere al sito ufficiale `https://git-scm.com/download/win`;
- scaricare l'eseguibile;
- installare l'eseguibile seguendo la procedura guidata.

Per i sistemi *MacOS*$_G$:

- accedere al sito ufficiale `https://git-scm.com/download/mac`;
- scaricare l'eseguibile;
- aprire il file appena scaricato e avviare l'installazione cliccando sul file `.pkg`.

La configurazione iniziale prevede l'inserimento del nome dell'utente e la connessione con l'account GitHub tramite la seguente procedura:

- `git config -global user.name <Nome Cognome>`: imposta il nome dell'utente, che comparirà come autore delle commit effettuate;
- `git config -global user.email <email>`: imposta l'email dell'utente; deve essere impostata la stessa email utilizzata per la registrazione su GitHub.

Per la creazione di una cartella locale del repository si deve seguire la seguente procedura:

- creare una nuova cartella;
- aprire il terminale;
- posizionarsi all'interno della cartella precedentemente creata;
- eseguire il comando: `git init`;
- recuperare l'indirizzo URL del progetto su GitHub;
- eseguire il comando: `git clone <indirizzo appena recuperato>`.

3.2.5.2 Aggiornamento del repository

L'aggiornamento del repository deve essere svolto eseguendo i seguenti comandi:

1. eseguire `git pull` per scaricare le modifiche da remoto;
2. se sono presenti conflitti:
   (a) eseguire `git stash` per salvare momentaneamente le modifiche locali;
   (b) eseguire `git pull`;
   (c) eseguire `git stash apply` per ripristinare le modifiche;
3. eseguire `git add NomeFile` per ognuno dei file in cui sono state effettuate delle modifiche;
4. eseguire `git commit -m "descrizione modifiche"`;
5. eseguire `git push` per inviare le modifiche al repository remoto.

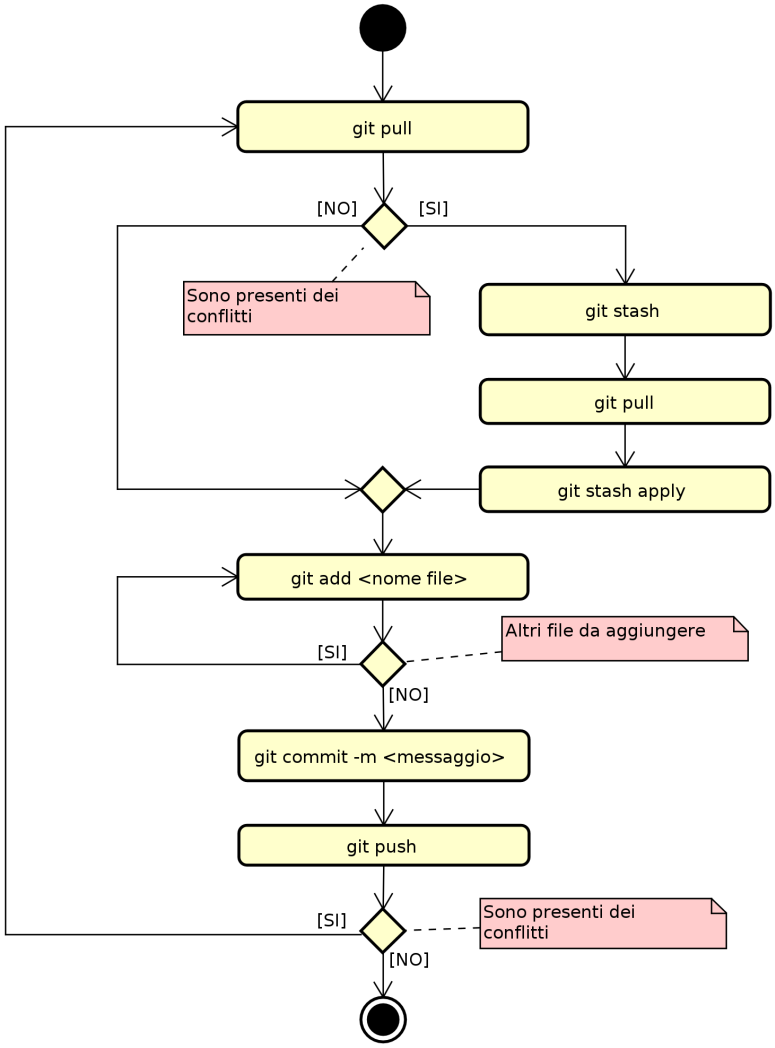

Figura 13: Procedura di aggiornamento del repository.

3.2.5.3 Comandi utili Git

Per l'approfondimento dei principali comandi riguardanti l'utilizzo di Git è stata redatta una guida, consultabile al seguente indirizzo: Guida Git.

3.3 Verifica

3.3.1 Scopo

Lo scopo del processo di verifica è di garantire che ogni attività dei processi svolti non introduca errori nel prodotto e che soddisfi i requisiti o le condizioni necessarie per essere considerata accettabile. La corretta implementazione del processo deve:

- fornire le procedure di verifica necessarie;
- individuare i criteri per la verifica;
- individuare eventuali difetti perchè possano essere corretti.

3.3.2 Documenti

Il *Responsabile di progetto* deve assegnare i compiti ai *Verificatori*, ognuno di essi deve effettuare un controllo accurato delle seguenti regole:

- devono essere rispettate le norme tipografiche (vedi sez. 3.1.4);
- devono essere rispettate le componenti grafiche (vedi sez. 3.1.5);
- devono essere rispettate le regole per i termini a glossario (vedi sez. 3.1.6.3);
- devono essere utilizzati periodi brevi, corretti e il più possibile semplici;
- devono essere usati i comandi LaTeX descritti nell'appendice B;
- deve essere rispettata la struttura del documento (vedi sez. 3.1.3).

3.3.3 Metriche

3.3.3.1 Processi

3.3.3.1.1 Livello CMM ($LCMM$)

La metrica scelta è il Livello *CMM*$_G$ (LCMM). La scala assume valori da 1 (peggiore) a 5 (migliore).

3.3.3.1.2 Schedule Variance (SV)

La metrica utilizzata è la Schedule Variance (SV). È implementata come differenza tra la pianificazione dei costi del lavoro eseguito e del lavoro pianificato. Entrambi questi valori sono intesi nella loro accezione temporale (giorni) e non monetaria.

$$SV = BCWP - BCWS$$

dove $BCWP$=Budgeted Cost of Work Performed e $BCWS$=Budgeted Cost of Work Scheduled.

3.3.3.1.3 Cost Variance (CV)

La metrica utilizzata è la Cost Variance (CV). È implementata come differenza tra costo pianificato e costo effettivo del lavoro eseguito.

$$CV = BCWP - ACWP$$

dove $BCWP$=Budgeted Cost of Work Performed e $ACWP$=Actual Cost of Work Performed.

3.3.3.1.4 Rischi Non Preventivati (RNP)

È stato creato un contatore RNP (Rischi Non Preventivati) che aumenta di 1 ogni volta che si presenta un rischio non preventivato nel *Piano di progetto v4.0.0*. Il contatore rimane attivo durante tutto il ciclo di vita del progetto.

3.3.3.1.5 Righe Documento Per Ora ($RDCO$)

È stato creato un contatore RDCO (Righe Documento Per Ora) che tiene conto delle righe di documento che vengono redatte da un membro del gruppo.

$$RDCO = \frac{RP}{OI}$$

dove RP=Righe Prodotte e OI=Ore Impiegate nella scrittura del documento

3.3.3.1.6 Numero Comandi Richiesti (NCR)

È stato creato un contatore NCR (Numero Comandi Richiesti) che tiene conto dei comandi personalizzati per il template LaTeX che vengono richiesti al *Responsabile di progetto*.

3.3.3.1.7 Risoluzione Verticale (RV)

Metrica che tiene conto dei pixel verticali di un'immagine. Come riferimento viene preso il valore di RV minimo tra le immagini.

3.3.3.1.8 Percentuale Tracciamento Modifiche (PTM)

$$PTM = \frac{NTC}{NARM}$$

dove NTC=Numero di Task Completati relativi ad un documento e $NARM$=Numero Aggiunte Registro Modifiche.

3.3.3.1.9 Righe Codice Per Ora ($RCPO$)

$$RCPO = \frac{RC}{OI}$$

dove RC=Righe di Codice scritte e OI=Ore Impiegate.

3.3.3.1.10 Use Case senza Scenario Principale ($UCSP$)

Metrica che tiene conto del numero di use case senza scenario principale.

3.3.3.1.11 Percentuale di Requisiti Obbligatori Coperti ($PROC$)

$$PROC = \frac{ROC}{ROI}$$

dove ROC=Requisiti Obbligatori Coperti, ovvero assegnati ad un *package*$_G$ e ROI=Requisiti Obbligatori Individuati.

3.3.3.1.12 Grado di Accoppiamento (GA)

Metrica che indica le dipendenze uscenti delle componenti (package/classi) del sistema verso altre componenti. Viene preso come riferimento il massimo GA tra quelli presenti.

3.3.3.1.13 Grado di Utilità (GU)

Metrica che indica le dipendenze entranti nelle componenti del sistema. Viene preso come riferimento il minimo GU tra quelli presenti.

3.3.3.2 Documenti

3.3.3.2.1 Indice Gulpease (IG)

La metrica utilizzata è l'*Indice Gulpease*$_G$ (IG), un indice di leggibilità di un testo tarato sulla lingua italiana. La scala va da 0 a 100, dove "0" indica un documento di bassa leggibilità e "100" uno di alta. Risulta che i testi con indice:

- inferiore a 80 sono difficili da leggere per chi ha la licenza elementare;
- inferiore a 60 sono difficili da leggere per chi ha la licenza media;
- inferiore a 40 sono difficili da leggere per chi ha un diploma superiore.

$$IG = 89 + \frac{300 \cdot NF - 10 \cdot NL}{NP}$$

dove NF è il Numero di Frasi, NL è il Numero di Lettere e NP è il Numero di Parole presenti nel testo.

3.3.3.2.2 Errori riguardanti le Norme interne e Non Corretti ($ENNC$)

La metrica utilizzata è il numero di Errori riguardanti le Norme interne rinvenuti e Non Corretti ($ENNC$). È implementata con un contatore che aumenta di 1 ogni volta che un errore riguardante le norme interne rilevato da un *Verificatore* non viene corretto. Il contatore fa riferimento ad uno specifico *periodo*$_G$ e viene azzerato successivamente.

3.3.3.2.3 Errori Ortografici Non Corretti ($EONC$)

La metrica utilizzata è un contatore di Errori Ortografici Non Corretti ($EONC$), che aumenta di 1 ogni volta che un errore ortografico rilevato da un *Verificatore* non viene corretto. Il contatore fa riferimento ad uno specifico periodo e viene azzerato successivamente.

3.3.3.2.4 Errori Concettuali Non Corretti ($ECNC$)

La metrica utilizzata è chiamata Errori Concettuali Non Corretti ($ECNC$). La formula è data dal complemento a 1 del rapporto tra errori concettuali corretti e rilevati. Tali errori possono essere rilevati dai Verificatori o dal committente.

$$ECNC = \left(1 - \frac{ECC}{ECR}\right) \cdot 100$$

dove ECC=Errori Concettuali Corretti e ECR=Errori Concettuali Rinvenuti.

3.3.3.2.5 Livello Annidamento Indice (LAI)

Metrica che conta il livello di annidamento dell'indice. Viene preso come riferimento il massimo LAI tra quelli presenti.

3.3.3.3 Codice

3.3.3.3.1 Implementazione delle Funzionalità Obbligatorie (IFO)

La metrica utilizzata è chiamata Implementazione delle Funzionalità Obbligatorie (IFO). Essa consiste in un rapporto tra il numero di requisiti obbligatori soddisfatti e identificati

$$IFO = \frac{ROS}{ROI} \cdot 100$$

dove ROS=numero di Requisiti Obbligatori Soddisfatti e ROI=numero di Requisiti Obbligatori Identificati.

3.3.3.3.2 Implementazione delle Funzionalità Desiderabili (IFD)

La metrica utilizzata è chiamata Implementazione delle Funzionalità Desiderabili (IFD). Essa consiste in un rapporto tra il numero di requisiti desiderabili soddisfatti e identificati

$$IFD = \frac{RDS}{RDI} \cdot 100$$

dove RDS=numero di Requisiti Desiderabili Soddisfatti e RDI=numero di Requisiti Desiderabili Identificati.

3.3.3.3.3 Numero di Statement per Metodo (NSM)

Metrica che conta il Numero di *Statement*$_G$ per Metodo. Viene preso come riferimento il massimo NSM tra quelli presenti.

3.3.3.3.4 Numero di Parametri per Metodo (NPM)

Metrica che conta il Numero di Parametri per Metodo. Viene preso come riferimento il massimo NPM tra quelli presenti.

3.3.3.3.5 Numero Campi Dati Per Classe ($NCDPC$)

Metrica che conta il Numero di Campi Dati per Classe. Viene preso come riferimento il massimo $NCDPC$ tra quelli presenti.

3.3.3.3.6 Numero Ciclomatico (NC)

La metrica è basata sul Numero Ciclomatico (NC). Il conteggio viene fatto a livello di metodo.

$$NC = e - n + 2p$$

dove e=numero di archi, n=numero di nodi, p=numero di componenti connesse.
Viene preso come riferimento il massimo NC tra quelli presenti.

3.3.3.3.7 Numero di Variabili dichiarate e Non Utilizzate ($NVNU$)

Metrica che conta il numero di variabili dichiarate e non utilizzate.

3.3.3.3.8 Rapporto tra le linee di Commento e le linee di Codice (RCC)

La metrica è implementata come Rapporto tra le linee di Commento e le linee di Codice (RCC).

$$RCC = \left(\frac{LDCM}{LDCC}\right) \cdot 100$$

dove $LDCC$=Linee Di Codice e $LDCM$=Linee Di Commento.

3.3.3.3.9 Superamento dei Test Pianificati (STP)

La metrica utilizzata è il Superamento dei Test Pianificati (STP). I test presi in considerazione sono quelli necessari a verificare l'implementazione delle funzionalità previste dai requisiti.

$$STP = \frac{TS}{TP} \cdot 100$$

dove TS=numero di Test Superati e TP=numero di Test Pianificati.

3.3.3.3.10 Breakdown Avoidance (BA)

La metrica utilizzata è la Breakdown Avoidance (BA).

$$BA = \left(1 - \frac{NI}{NSA}\right) \cdot 100$$

dove NI=Numero di Interruzioni e NSA=Numero di Situazioni Anomale.

3.3.3.3.11 Failure Avoidance (FA)

La metrica utilizzata è la Failure Avoidance (FA).

$$FA = \left(\frac{SAE}{SAT}\right) \cdot 100$$

dove SAE=Situazioni Anomale Evitate e SAT=Situazioni Anomale Testate.

3.3.3.3.12 Statement Coverage (SC)

Metrica utilizzata per calcolare la copertura degli statement.

$$SC = \frac{NSE}{NSM}$$

dove NSM=Numero Statement del Metodo e NSE=Numero Statement Eseguiti dal test.
Viene preso come riferimento il minimo SC tra quelli presenti.

3.3.3.3.13 Branch Coverage (BC)

Metrica utilizzata per calcolare la copertura dei flussi logici.

$$BC = \frac{NFLT}{NFLM}$$

dove $NFLT$=Numero di Flussi Logici del Metodo Testati e $NFLM$=Numero di Flussi Logici del Metodo. Viene preso come riferimento il minimo BC tra quelli presenti.

3.3.4 Strumenti

3.3.4.1 Tabella metriche processo-strumenti

Metrica	Strumento
LCMM	Procedura interna 3.3.5.5
SV	*Teamwork*$_G$, vedi la sezione 4.1.6.3
CV	Teamwork, vedi sezione 4.1.6.3
RNP	Procedura interna 3.3.5.6
RDCO	Script interno, vedi procedura 3.3.5.8
NCR	Procedura interna 4.2.7.5
RV	Script interno, vedi procedura 3.3.5.9
PTM	Procedura interna 4.1.6.3.3
RCPO	Script interno, vedi procedura 3.3.5.8
UCSP	Statistiche Trender
PROC	Statistiche Trender
GA	Statistiche Trender
GU	Statistiche Trender

Tabella 1: Tabella metriche processo-strumenti

3.3.4.2 Tabella metriche documenti-strumenti

Metrica	Strumento
IG	Script interno, vedi procedura 3.3.5.7
ENCC	Procedura interna 3.3.5.4
EONC	Procedura interna 3.3.5.4
ECNC	Procedura interna 3.3.5.4
LAI	Script interno, vedi procedura 3.3.5.10

Tabella 2: Tabella metriche documenti-strumenti

3.3.4.3 Tabella metriche codice-strumenti

Metrica	Strumento
IFO	Statistiche Trender
IFD	Statistiche Trender
NSM	JSMeter
NPM	ESLint
NCDPC	Statistiche Trender
NC	JSMeter
NVNU	ESLint
RCC	JSMeter
STP	Statistiche Trender
BA	Jasmine
FA	Jasmine
SC	WebStorm tramite Karma e Istanbul
BC	WebStorm tramite Karma e Istanbul

Tabella 3: Tabella metriche codice-strumenti

3.3.4.4 Verifica ortografica

Viene utilizzato il correttore in tempo reale integrato nell'applicazione *TeXstudio*$_G$ descritta nella sezione 3.1.7.2. Il correttore identifica e sottolinea eventuali refusi ortografici; un'analisi più approfondita del testo è compito dei *Verificatori*.

3.3.4.5 Indice leggibilità

La valutazione dell'indice di leggibilità è fatta secondo l'*indice Gulpease*$_G$ utilizzando uno script creato appositamente e fornito assieme al template. Lo script può analizzare sia il file sorgente scritto in TEX che il file pdf risultante.

3.3.4.6 Analisi statica

3.3.4.6.1 ESLint

ESLint è un tool che permette di effettuare analisi statica su file *JavaScript*$_G$ con lo scopo di ottenere codice più uniforme e privo di errori. I controlli che effettua sul codice vengono realizzati nei confronti di un insieme di regole personalizzabili, regole che gli sviluppatori possono attivare o disattivare in base alle loro guide di stile di codifica interna (vedi sezione: 2.2.5.2).
La documentazione e l'installazione di ESLint sono disponibili al seguente indirizzo:

`https://github.com/eslint/eslint`

3.3.4.6.2 JSMeter

JSMeter è uno strumento che permette di calcolare diversi indici di misurazione della complessità del codice JavaScript descritti nel *Piano di qualifica v4.0.0*. La documentazione e l'installazione di JSMeter sono disponibili al seguente indirizzo:

`http://jsmeter.info`

3.3.4.7 Analisi dinamica

3.3.4.7.1 Karma

Karma è un ambiente di testing, utile ad effettuare test su browser e dispositivi. Per descrivere i test vengono utilizzati dei framework esterni, come per esempio Mocha o Jasmine. Karma viene utilizzato assieme ad un framework per l'esecuzione dei test di unità.
La documentazione e l'installazione di Karma sono disponibili al seguente indirizzo:

https://github.com/karma-runner/karma

3.3.4.7.2 Jasmine

Jasmine$_G$ è un framework per implementare test di codice JavaScript, non dipendendo da browser o altri framework è adatto ad eseguire test di siti web ed è facilmente estendibile. Oltre a mettere a disposizione un set di operazioni di confronto permette anche di utilizzare driver e stub. I test implementati possono essere eseguiti tramite l'inserimento del codice in uno specifico file html o utilizzando un test runner esterno come Karma. La documentazione e l'installazione di Jasmine sono disponibili al seguente indirizzo:

https://jasmine.github.io/index.html

3.3.4.7.3 Istanbul

Instanbul è uno strumento che permette di calcolare alcune metriche di complessità del codice JavaScript descritte nel *Piano di qualifica v4.0.0* e verrà installato come modulo per *Node.js*$_G$. La documentazione e l'installazione di Istanbul sono disponibili al seguente indirizzo:

https://github.com/gotwarlost/istanbul

3.3.5 Procedure

3.3.5.1 Attività manuale di verifica

Ogni attività manuale di verifica deve essere accompagnata da un resoconto (si veda la sez. 3.3.5.4 per i dettagli). Per svolgere l'attività di verifica, il *Verificatore* deve rispettare le seguenti indicazioni:

1. aprire il file sorgente da verificare (se è un documento, anche il file pdf);
2. annotare gli errori rilevati da correggere;
3. inserire nel codice sorgente uno dei seguenti commenti:
   - `TODO` per segnalare un'aggiunta da eseguire al codice;
   - `FIXME` per segnalare un errore rilevato per cui è stata proposta una soluzione;
   - `BUG` per segnalare un'anomalia da rivedere e correggere;
4. se è un documento, calcolare indice gulpease (vedi sez. 3.3.4.5) e riportare l'esito nel resoconto;
5. aprire un subticket come riportato nella sez. 3.3.5.4 riportando gli errori.

3.3.5.2 Analisi

3.3.5.2.1 Analisi statica

Tecnica utilizzata per l'analisi e la verifica del codice sorgente e della documentazione associata. Può essere applicata secondo due strategie:

- ***walkthrough***$_G$**:** lettura completa del codice sorgente da analizzare. Va utilizzata unicamente durante il primo periodo del progetto in quanto risulta onerosa e non efficiente. Gli *Analisti* che la utilizzano devono stilare una lista di controllo con gli errori rilevati più frequentemente;
- ***inspection***$_G$**:** lettura mirata del codice sorgente da analizzare. È necessario avere una lista di controllo per poter localizzare eventuali punti critici in cui cercare errori. Dopo ogni analisi la lista di controllo deve essere incrementata con eventuali nuovi errori rilevati.

3.3.5.2.2 Analisi dinamica

Tecnica per l'analisi del prodotto software, richiede l'esecuzione del programma e viene eseguita tramite dei test che verificano il funzionamento del prodotto. I test devono essere ripetibili e a parità di condizioni iniziale e ambiente devono fornire lo stesso output. Per ogni test deve quindi essere definito:

- **ambiente:** sistema hardware e software in cui viene eseguito il test;
- **stato iniziale:** stato iniziale da cui si parte ad eseguire il test;
- **input:** input inserito;
- **output:** output atteso.

3.3.5.3 Task di verifica

La gestione dei task di verifica avviene tramite la creazione di ticket nell'applicazione *Teamwork*$_G$ che viene descritta nella sezione 4.1.5.2.
Al termine di ogni ticket il *Responsabile di progetto* dovrà seguire la seguente procedura:

1. creare un ticket di verifica con i riferimenti al task da verificare e contrassegnarlo con la dicitura *[VERIFICA]*;
2. assegnare il ticket creato ad un *Verificatore*;
3. al completamento del ticket di verifica contrassegnare il ticket originale con la dicitura *[VERIFICATO]*.

Vedi anche figura 14.

3.3.5.4 Gestione delle anomalie

La gestione delle anomalie durante il processo di verifica avviene tramite la creazione di ticket nell'applicazione Teamwork che viene descritta nella sezione 4.1.5.2. Nel caso in cui un *Verificatore* dovesse trovare delle anomalie durante un task di verifica queste dovranno essere gestite nel seguente modo:

1. deve essere creato un subticket contrassegnato da uno dei seguenti tag, questi saranno utilizzati in un secondo momento per il calcolo delle relative metriche:
   - *[EN]* se l'anomalia riguarda il rispetto delle norme (vedi metrica 3.3.3.2.2);
   - *[EO]* se l'anomalia riguarda un errore ortografico (vedi metrica 3.3.3.2.3);
   - *[EC]* se l'anomalia riguarda un errore concettuale (vedi metrica 3.3.3.2.4);
   - *[ERR]* se l'anomalia non riguarda nessuno dei campi precedenti.
2. assegnare i subticket creati agli assegnatari del task di cui si sta eseguendo la verifica;
3. al completamento di tutti i subticket chiudere il ticket di verifica.

Vedi anche figura 14.

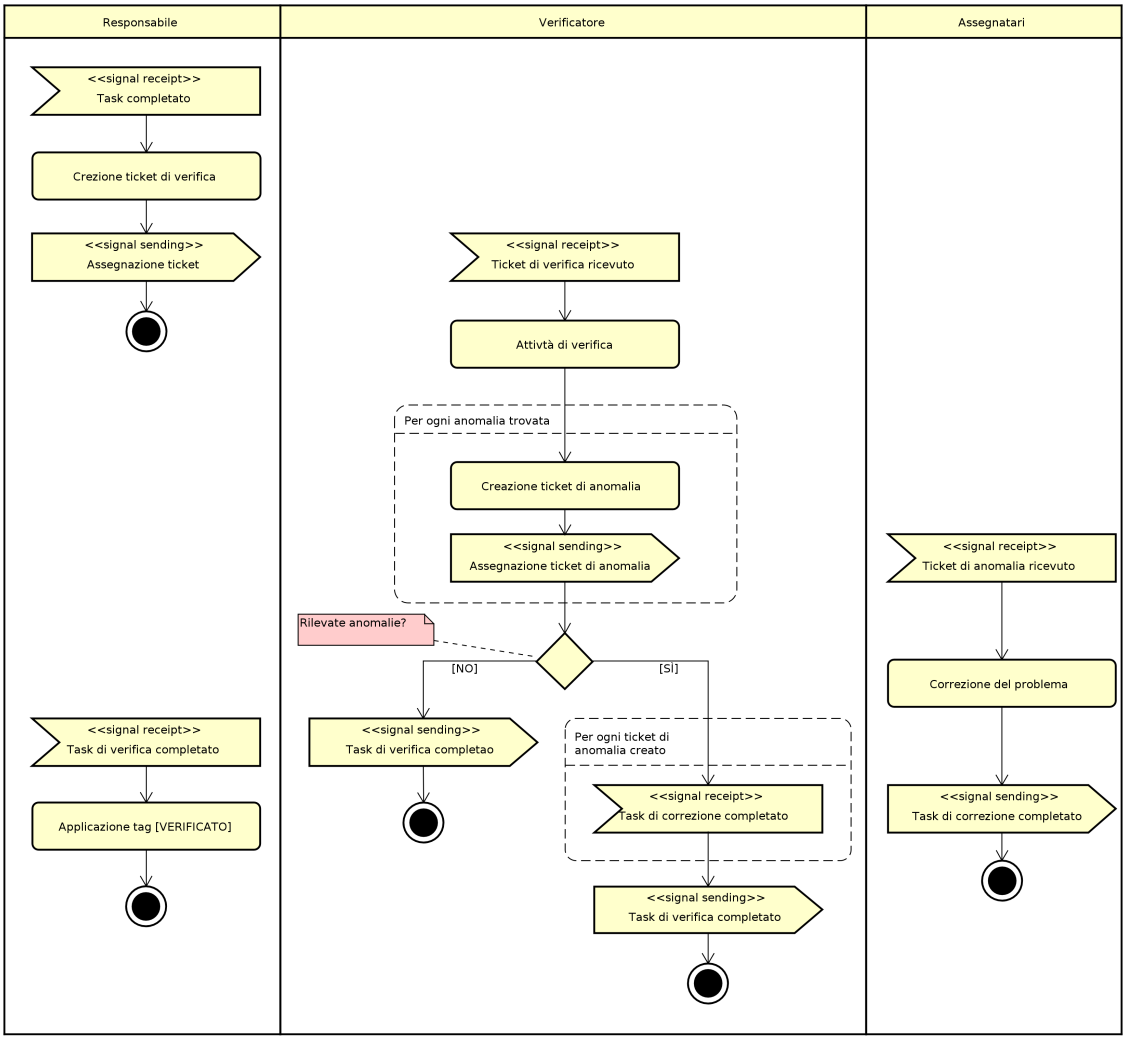

Figura 14: Procedura di verifica dei task e gestione delle anomalie. Riferita nelle sezioni 3.3.5.3 e 3.3.5.4

3.3.5.5 Calcolo CMM

Al termine di ogni periodo il *Responsabile di progetto* ha il compito di individuare il livello CMM che si è raggiunto. La decisione deve essere presa tenendo in considerazione:

- le norme prodotte fino a quel momento ed il loro rispetto;
- i processi svolti, i loro risultati e la loro riproducibilità;
- le metriche individuate ed i loro valori.

3.3.5.6 Calcolo rischi non preventivati

Al momento di attualizzare un rischio il *Responsabile di progetto* deve verificare che tale rischio sia presente nel *Piano di progetto*. Se così non fosse dovrà procedere nel seguente modo:

- creare un task ed inserire nel titolo il tag *[RISK]*;
- scrivere nella descrizione del task il rischio riscontrato;
- assegnare il task ad un analista responsabile del *Piano di progetto* in modo che sia inserito nel documento.

Il *Responsabile di progetto* dovrà tenere traccia dei ticket creati con il tag *[RISK]* per calcolare il valore della metrica.

3.3.5.7 Calcolo dell'indice Gulpease

Per l'esecuzione dello script per il calcolo dell'indice Gulpease (descritto nella sezione 3.3.4.5) si dovranno seguire i seguenti passi:

1. aprire una riga di comando o terminale;
2. posizionarsi nella cartella `/script` all'interno dello spazio di lavoro condiviso *Dropbox*$_G$;
3. eseguire il comando: `gulpeasepdf.pl NomeDocumento.pdf`;
4. copiare l'esito prodotto dall'output del terminale.

3.3.5.8 Calcolo linee modificate

1. aprire una riga di comando o terminale;
2. posizionarsi nella cartella `/script` all'interno del repository desiderato (Documenti o DeGeOp);
3. eseguire il comando `git pull` per assicurarsi che il repository sia aggiornato;
4. eseguire il comando: `./checkLOCgit.sh [-d numero_giorni]`;
5. le immagini segnalate dal comando non rispettano la metrica.

3.3.5.9 Verifica risoluzione immagini

1. aprire una riga di comando o terminale;
2. posizionarsi nella cartella `/script` all'interno del repository desiderato (Documenti o DeGeOp);
3. eseguire il comando `git pull` per assicurarsi che il repository sia aggiornato;
4. eseguire il comando: `./checkRes.sh <dimensione_minima>`;
5. le immagini segnalate hanno la dimensione verticale minore di quella richiesta.

3.3.5.10 Verifica annidamento indice

1. aprire una riga di comando o terminale;
2. posizionarsi nella cartella `/script` all'interno del repository desiderato (Documenti o DeGeOp);
3. eseguire il comando `git pull` per assicurarsi che il repository sia aggiornato;
4. eseguire il comando: `./checkSubSub.sh`;
5. il comando restituisce i documenti e le righe in cui è usato un livello di annidamento troppo elevato.

3.4 Validazione

3.4.1 Scopo

Lo scopo del processo di validazione è di determinare in maniera oggettiva che il prodotto esaminato sia conforme ai requisiti richiesti e che soddisfi il compito per cui è stato creato. La corretta implementazione del processo deve:

- utilizzare gli stessi strumenti e le stesse procedure del processo di verifica;
- fornire tutti i dati necessari alla valutazione del prodotto;
- verificare che tutte le metriche stabilite siano soddisfatte.

3.4.2 Procedure

L'attività di verifica deve essere svolta rispettando il seguente ordine:

1. i *Verificatori* eseguono i test sul prodotto finale tracciando gli esiti ottenuti;
2. il *Responsabile di progetto* analizza i risultati ottenuti decidendo se accettarli o ripetere alcuni test;
3. una volta accettati il *Responsabile di progetto* consegna i risultati al proponente.

4 Processi organizzativi

4.1 Gestione processi

4.1.1 Scopo

Lo scopo della gestione dei processi è di migliorare l'organizzazione e la cooperazione tra i membri del *gruppo*$_G$. La corretta implementazione del processo deve:

- stabilire le modalità di comunicazione del gruppo;
- definire i ruoli ed i compiti specifici;
- fornire la documentazione su strumenti di organizzazione e relative procedure.

4.1.2 Comunicazioni

4.1.2.1 Interne

Per la comunicazione interna dei membri del gruppo, viene utilizzata l'applicazione *Slack*$_G$ descritta in maniera piu dettagliata nella sezione 4.1.5.1.

4.1.2.2 Esterne

Per le comunicazioni esterne è stata creata la seguente mail:

zephyrus.swe@gmail.com

Il *Responsabile di progetto* è la persona incaricata di inviare comunicazioni con questo indirizzo. Tutte le mail ricevute verranno inoltrate automaticamente ad ogni membro del gruppo a titolo informativo.

4.1.3 Incontri

4.1.3.1 Interni

Il *Responsabile di progetto* ha il compito di organizzare gli incontri interni rispettando la procedura descritta nella sezione 4.1.6.1. Qualsiasi membro del gruppo può richiedere un incontro interno. Sarà compito del *Responsabile di progetto* accettare o rifiutare la richiesta. Al termine dell'incontro deve essere redatto un verbale, di cui si rimanda alla sezione 3.1.6.4 per la sua composizione in dettaglio.

4.1.3.2 Esterni

Il *Responsabile di progetto* ha il compito di organizzare gli incontri esterni con il proponente o committente seguendo la procedura descritta nella sezione 4.1.6.2. Qualsiasi membro del gruppo può richiedere un incontro esterno. Sarà compito del *Responsabile di progetto* accettare o rifiutare la richiesta. Al termine dell'incontro deve essere redatto un verbale, di cui si rimanda alla sezione 3.1.6.4 per la sua composizione in dettaglio.

4.1.4 Ruoli di progetto

Ogni componente del gruppo deve ricoprire almeno una volta ciascuno dei ruoli previsti nello sviluppo del progetto. Nel *Piano di progetto* vengono assegnati compiti e ruoli che i membri del gruppo si impegnano a rispettare. Di seguito sono elencati i diversi incarichi, delineando per ciascuno mansioni e responsabilità.

4.1.4.1 Responsabile di progetto

Il *Responsabile di progetto* è la figura che rappresenta il gruppo e il progetto presso committente e proponente. Approva le scelte prese dal gruppo e se ne assume la responsabilità. La sua presenza segue tutta la durata del progetto. Le sue responsabilità sono:

- gestione delle risorse;
- approvazione della documentazione di progetto;
- analisi e mitigazione dei rischi;
- coordinamento e pianificazione delle attività di progetto seguendo il *Piano di progetto*.

4.1.4.2 Amministratore di progetto

L'*Amministratore* è responsabile dell'ambiente di lavoro del gruppo, ne controlla l'efficienza e l'operatività. Le sue principali responsabilità sono:

- controllo delle versioni e configurazioni del prodotto;
- gestione del versionamento e dell'archiviazione della documentazione;
- risoluzione dei problemi inerenti la gestione di processi e risorse;
- controllo e miglioramento degli strumenti di lavoro e dell'infrastruttura;
- redazione e aggiornamento delle *Norme di progetto*.

4.1.4.3 Analista

L'*Analista* ha il compito di esaminare e studiare attentamente il dominio del problema, la sua presenza non è necessaria per tutta la durata del progetto. Le sue responsabilità sono:

- capire il dominio di lavoro del cliente;
- analizzare e capire la natura del problema posto dal cliente;
- redigere lo studio di fattibilità e l'analisi dei requisiti.

4.1.4.4 Progettista

Il *Progettista* è responsabile di tutto ciò che riguarda la progettazione software. Deve avere conoscenze tecniche e tecnologiche aggiornate per la gestione del progetto. Le sue responsabilità sono:

- fornire una soluzione attuabile entro i limiti di tempo;
- descrivere il funzionamento del sistema a diversi livelli di dettaglio;
- effettuare scelte su aspetti tecnici del progetto, rendendolo efficiente, robusto e manutenibile.

4.1.4.5 Programmatore

Il *Programmatore* si occupa dell'attività di codifica. Le sue responsabilità sono:

- implementare le scelte dettate dal *Progettista*, senza apportare modifiche personali;
- documentare il codice prodotto;
- rispettare le convenzioni riportate nel presente documento;
- realizzare strumenti per verifica e validazione.

4.1.4.6 Verificatore

Il *Verificatore* è responsabile dell'attività di verifica. Deve avere una profonda conoscenza delle *Norme di progetto* ed è presente per tutta la durata del progetto. Le sue responsabilità sono:

- controllare l'osservazione delle *Norme di progetto* lungo tutte le attività del progetto stesso.

4.1.5 Strumenti

4.1.5.1 Slack

Slack è un servizio gratuito di messaggistica professionale disponibile su piattaforme mobile, desktop e web. Le principali motivazioni che hanno portato il gruppo alla scelta di questo strumento sono:

- possibilità di integrazione con molti servizi, tra cui: *Dropbox*$_G$, *GitHub*$_G$, *Google Drive*$_G$, bot etc. Vedi sezione 4.2.6.1;
- possibilita di creare canali tematici personalizzati con la profilazione delle utenze e delle notifiche;
- utilizzato come ulteriore canale per comunicare con *RiskApp* in quanto anche loro lo utilizzano;
- possibilità di richiamare all'attenzione un membro del gruppo con il comando `@`;
- possibilità di classificare un commento come "rilevante" tramite il comando `pin to`;
- possibilità di inserire dei reminder sui messaggi;
- *cross-platform*$_G$ e *cross-device*$_G$;
- interfaccia piu ricca e organizzata rispetto alle usuali applicazioni di messaggistica.

Lo spazio dedicato al gruppo si trova al seguente indirizzo:

`https://zephyrus-swe.Slack.com/home`

4.1.5.2 Teamwork

Teamwork$_G$ è un'applicazione web di project management che permette di sfruttare le seguenti funzionalità principali:

- gestione dei task;
- gestione degli appuntamenti con scadenze a calendario;
- pianificazione del lavoro;
- rendiconto ore lavoro su intero progetto e/o su specifici task.

Le principali motivazioni che hanno portato il gruppo alla scelta di questo strumento sono:

- funzionalità essenziali gratuite ;
- alta portabilità ed accessibilità essendo fruibile via web;
- interfaccia semplice e funzionale.

Lo spazio di lavoro dedicato al gruppo di trova al seguente indirizzo:

`https://swe2016.teamwork.com/`

4.1.5.3 Condivisione file

Dropbox è un servizio che offre la possibilità di salvare file su una piattaforma *cloud*$_G$ personale e mantenerli sincronizzati tra diversi dispositivi tramite un client. Il gruppo ha scelto Dropbox per gestire file che non necessitano versionamento, inoltre le principali motivazioni che ne hanno portato alla scelta sono:

- alta velocità nella sincronizzazione dei file;
- servizio già conosciuto e largamente utilizzato da tutti i membri del gruppo;
- lo spazio disponibile nella versione gratuita è sufficente per consentire una discreta quantità di file associati a questo progetto;
- alta portabilità e usabilità essendo cross-platform e cross-device.

Indirizzo per il download:

`https://www.dropbox.com`

Google Drive è un servizio cloud, di memorizzazione e sincronizzazione online offerto da *Google*. Il servizio comprende il *file hosting*$_G$, il file sharing e la modifica collaborativa di documenti. Per accedere allo spazio condiviso è necessario che ogni membro sia prima aggiunto dal *Responsabile di progetto*. Il gruppo ha scelto questo strumento per la possibilità di redigere documenti in maniera collaborativa come attività preliminare a fine organizzativo. Le principali motivazioni che hanno portato il gruppo alla scelta di questo strumento sono:

- servizio già conosciuto e utilizzato da tutti i membri del gruppo;
- gratuito;
- alta portabilità e usabilità essendo cross-platform e cross-device.

Lo spazio di lavoro dedicato al gruppo di trova al seguente link:

Google Drive Zephyrus

4.1.6 Procedure

4.1.6.1 Organizzazione incontri interni

L'organizzazione degli incontri interni deve essere svolta rispettando il seguente ordine:

1. controllare il foglio condiviso su *Google Drive*$_G$ *Orari impegni ricorrenti*;
2. postare sul canale Slack `#incontri-interni` alcune proposte di incontro, indicando: luogo, data e orario;
3. assicurarsi che tutti i membri del gruppo abbiano espresso la preferenza; in caso contrario, entro 24 ore, contattarli telefonicamente;
4. aggiungere l'evento nel calendario di *Teamwork*$_G$ con le seguenti caratteristiche:
    - **titolo evento:** Riunione interna;
    - **dove:** data, ora inizio, ora fine;
    - **dettagli:** luogo concordato, breve descrizione degli argomenti da trattare;
    - **persone:** assegnare i partecipanti;
    - **reminders:** assegnare dei promemoria se lo si ritiene necessario.

4.1.6.2 Organizzazione incontri esterni

L'organizzazione degli incontri esterni deve essere svolta rispettando il seguente ordine:

1. contattare il proponente per avere informazioni sulla sua disponibilità;
2. postare sul canale Slack *#incontri-esterni* alcune proposte di incontro, indicando: luogo, data e orario;
3. assicurarsi che tutti i membri del gruppo abbiano espresso la preferenza; in caso contrario, entro 24 ore, contattarli telefonicamente;
4. aggiungere l'evento nel calendario di Teamwork con le seguenti caratteristiche:
    - **Titolo evento:** Riunione esterna;
    - **Dove:** data, ora inizio, ora fine;
    - **Dettagli:** luogo concordato, breve descrizione degli argomenti da trattare;
    - **Persone:** assegnare i partecipanti;
    - **Reminders:** assegnare dei promemoria se lo si ritiene necessario.

4.1.6.3 Gestione dei ticket

Accedere allo spazio Teamwork del gruppo, posizionandosi nella sezione `"All Task"` oppure tramite il seguente link: https://swe2016.teamwork.com/#projects/140646/tasks. Premere su `"Add Task"` per la creazione o sul nome di un task già esistente per la modifica.

4.1.6.3.1 Creazione di un ticket

La creazione di un ticket deve essere svolta dal *Responsabile di progetto* rispettando il seguente ordine:

1. inserimento titolo task;
2. assegnazione delle persone incaricate al suo svolgimento;
3. inserimento data prevista di completamento;
4. inserimento del tempo stimato per il completamento;
5. inserimento descrizione;
6. inserimento priorità;
7. selezionare eventuali dipendenze da altri task;
8. inserimento di un *tag*$_G$ che identifichi il ruolo delle persone incaricate;
9. salvataggio task.

Vedi figura 15.

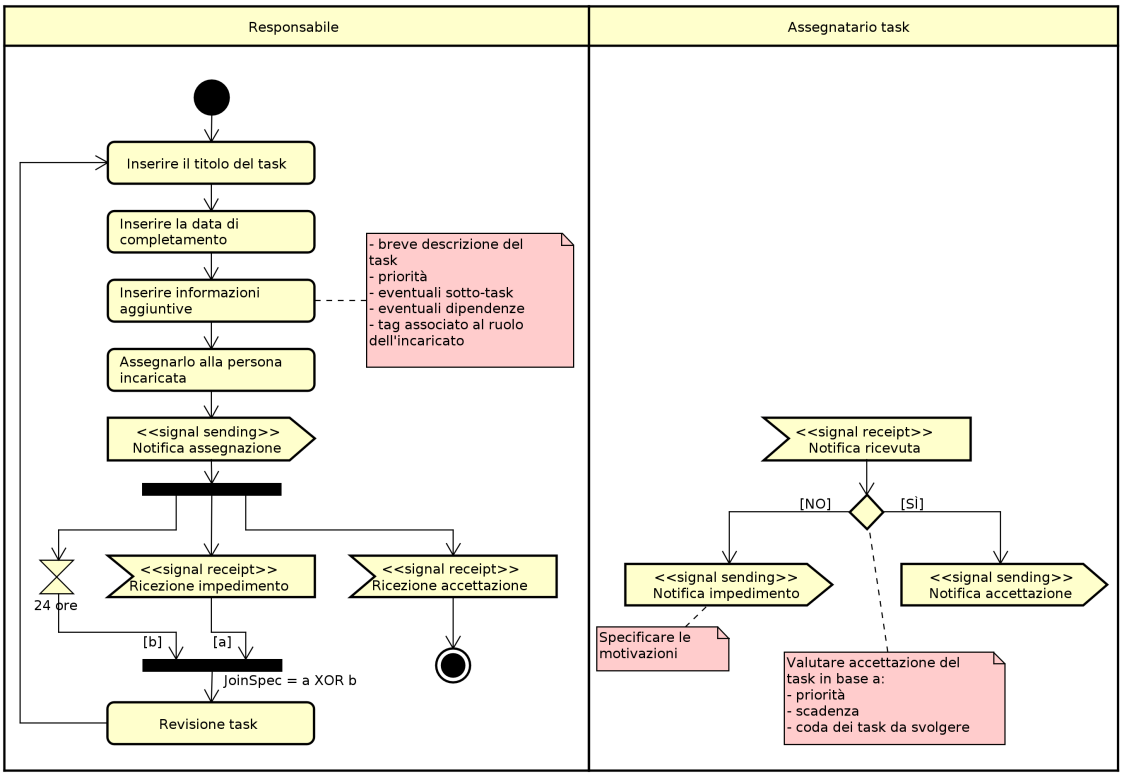

Figura 15: Procedura assegnazione ticket. Riferita nella sezione 4.1.6.3.1

4.1.6.3.2 Modifica di un ticket

La modifica di un ticket deve essere svolta rispettando il seguente ordine:

1. ricerca del ticket da modificare;
2. modifica di uno o più campi;
3. verifica inserimento dei campi dati: titolo, assegnatario, descrizione e data di completamento;
4. salvare le modifiche apportate.

Vedi figura 16.

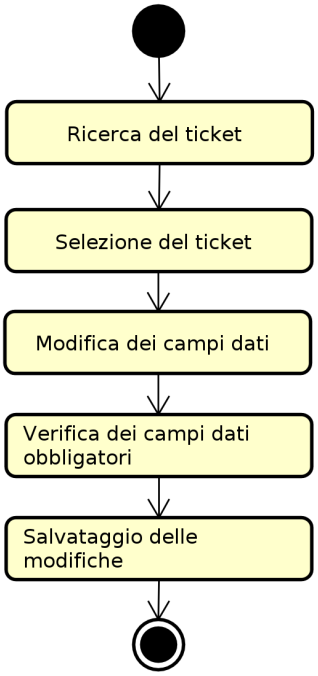

Figura 16: Procedura modifica ticket. Riferita nella sezione 4.1.6.3.2

4.1.6.3.3 Chiusura di un ticket

La chiusura di un ticket da parte dell'incaricato deve essere svolta rispettando la seguente procedura:

1. ricerca del ticket da chiudere;
2. aprire le proprietà del ticket e cliccare su `"Log time"`;
3. se il ticket è relativo ad un task di modifica di un documento:
   (a) eseguire un commit con solo l'aggiornamento del registro delle modifiche del documento;
   (b) copiare il codice del commit ed inserirlo nella descrizione del task;

(c) eseguire il push delle modifiche.

4. inserire il tempo speso per completare il task;
5. selezionare `"Billable"` se il tempo speso è rendicontabile;
6. selezionare `"Task is now complete"`;
7. cliccare su `"Log this time"`.

Vedi figura 17.

Una procedura automatica implementata nella chat Slack controllerà che il codice del commit inserito nel task sia presente nei messaggi inviati da GitHub, se così non fosse provvederà ad inviare un avviso al *Responsabile di progetto*.

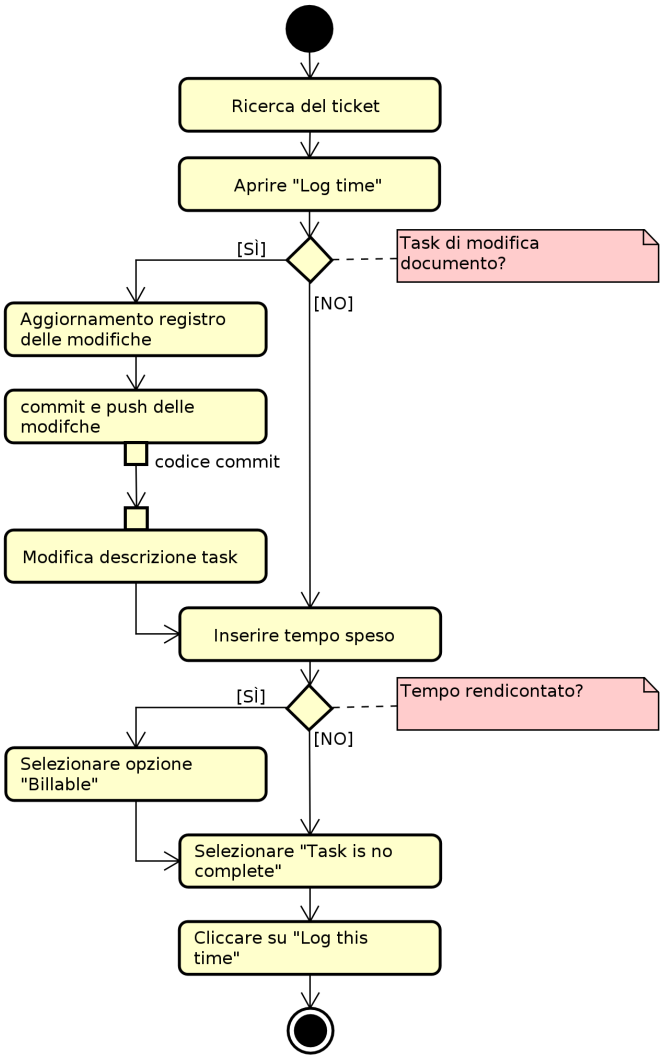

Figura 17: Procedura chiusura ticket. Riferita nella sezione 4.1.6.3.3

4.1.6.4 Gestione delle milestone

Accedere allo spazio Teamwork del gruppo, posizionandosi nella sezione *"Milestones"* oppure tramite il seguente link: https://swe2016.teamwork.com/#/projects/140646/milestones/upcoming. Premere su *"Add Milestone$_G$"* per la creazione o sul nome di una milestone esistente per la modifica.

4.1.6.4.1 Creazione milestone

La creazione di una milestone deve essere svolta rispettando il seguente ordine:

1. inserimento titolo milestone;
2. inserimento data;
3. assegnazione persone coinvolte;
4. assegnazione reminders;
5. inserimento descrizione;
6. salvataggio milestone.

Vedi figura 18.

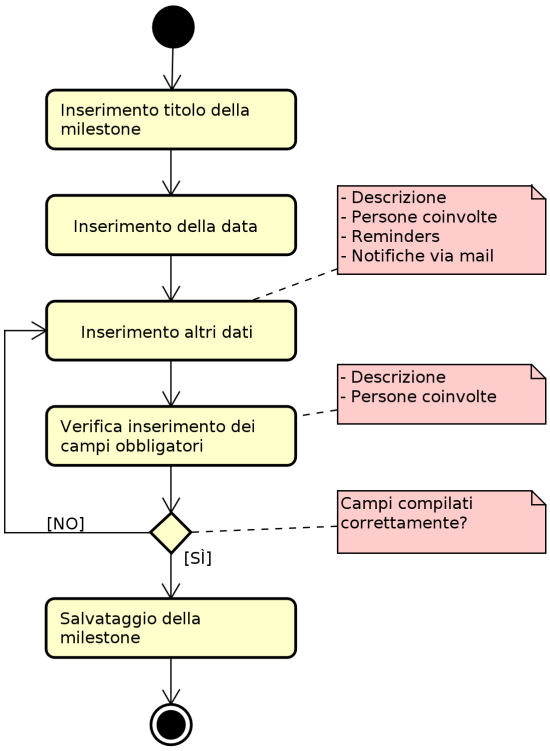

Figura 18: Procedura inserimento milestone. Riferita nella sezione 4.1.6.4.1

4.1.6.4.2 Modifica milestone

La modifica di una milestone deve essere svolta rispettando il seguente ordine:

- selezione milestone;
- modifica di uno o più campi dati;
- verifica inserimento dei campi dati: titolo, data, descrizione e assegnatari;
- salvataggio milestone.

Vedi figura 19.

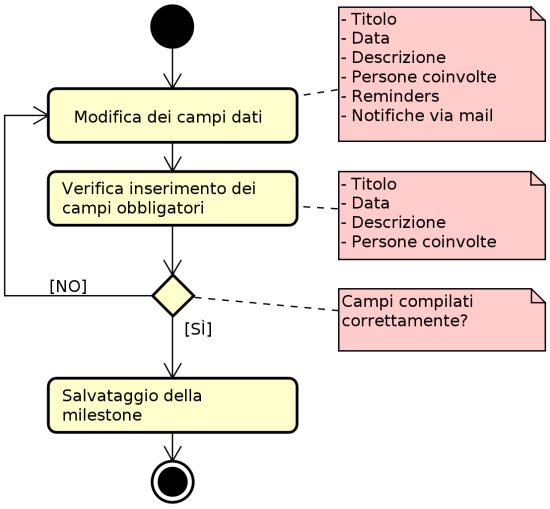

Figura 19: Procedura modifica milestone. Riferita nella sezione 4.1.6.4.2

4.1.6.5 Creazione nuovo canale Slack

La creazione di un nuovo canale Slack può essere fatta solo da parte del *Responsabile* e deve rispettare il seguente ordine:

1. effettuare il login su Slack nel sito o nell'applicazione, con l'utente *zephyrus.swe*;
2. cliccare nel menù di sinistra la voce `CHANNELS`;
3. cliccare il pulsante `New Channel`;
4. scegliere se rendere il canale pubblico o privato;
5. scegliere il nome del canale;
6. compilare l'eventuale descrizione riguardo la tematica del canale;
7. inserire i nomi delle persone che si desidera invitare;
8. cliccare sul bottone `Create Channel`.

Vedi figura 20.

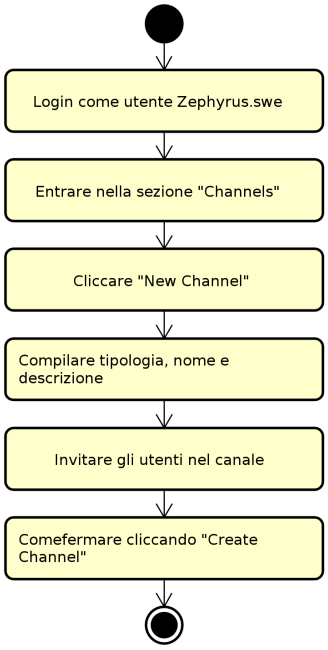

Figura 20: Procedura di creazione di un canale Slack. Riferita nella sezione 4.1.6.5

4.1.6.6 Modifica impostazioni canale Slack

In un canale Slack si possono modificare varie impostazioni, tra cui:

- invitare o rimuovere membri;
- preferenze di notifica;
- integrare app esterne;
- silenziare il canale;
- uscire dal canale.

Tranne la prima, tutte le altre sono a discrezione dell'utente. Di seguito si presenta la procedura eseguibile solo dal *Responsabile* per aggiungere un membro al canale.

1. effettuare il login su Slack nel sito o nell'applicazione, con l'utente `zephyrus.swe`;
2. cliccare nel canale che si desidera apportare modifiche;
3. cliccare il pulsante con l'icona a forma di ingranaggio e successivamente `Invite team Members`;
4. inserire il nickname della persona che si vuole aggiungere;
5. cliccare sul bottone `Invite`.

N.B. Per poter visualizzare il nome della persona da aggiungere è necessario che quest'ultima faccia già parte del Team su Slack, in caso contrario bisogna invitarla tramite indirizzo mail. Vedi figura 21.

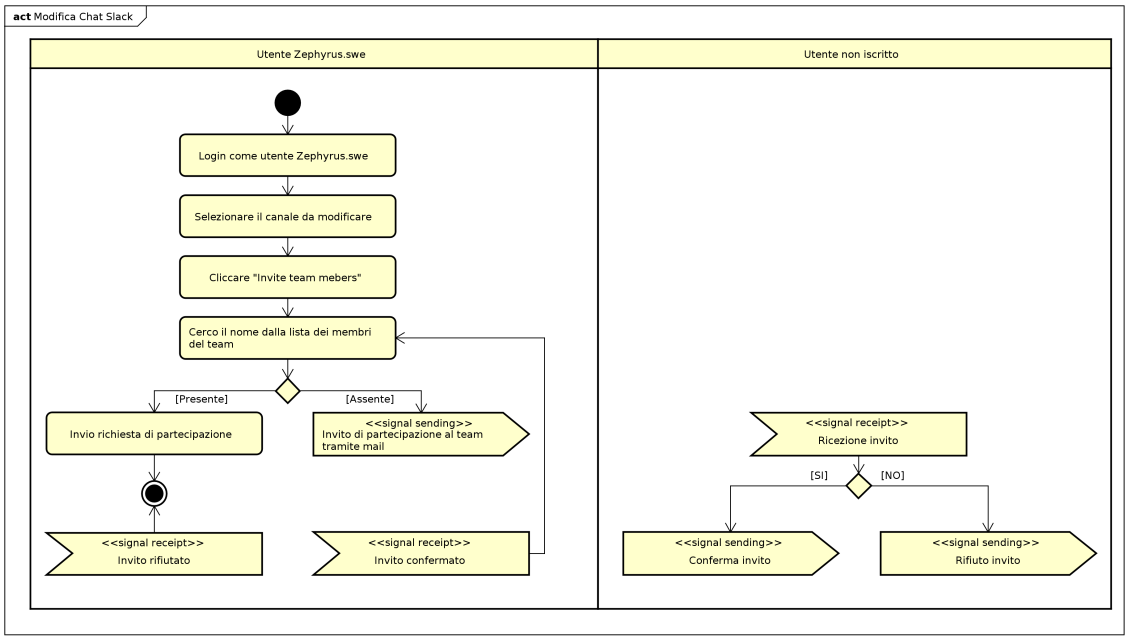

Figura 21: Procedura di creazione di un canale Slack. Riferita nella sezione 4.1.6.6

4.2 Gestione delle infrastrutture

4.2.1 Scopo

Lo scopo del processo di gestione delle infrastrutture è di stabilire e mantenere le infrastrutture e gli strumenti necessari allo svolgimento dei processi durante lo svolgimento del progetto. La corretta implementazione del processo deve:

- fornire un ambiente di lavoro idoneo;
- garantire il funzionamento di tutte le infrastrutture necessarie;
- indicare il corretto utilizzo delle infrastrutture.

4.2.2 Ambiente di sviluppo

L'*Amministratore* ha il compito di decidere l'ambiente ottimale per lo sviluppo e il test dell'applicazione. Se necessario è possibile coinvolgere il proponente nella decisione, in questo caso sarà compito del *Responsabile di progetto* organizzare le riunioni o i contatti necessari. Una volta deciso l'ambiente di sviluppo l'*Amministratore* dovrà indicare a tutti i membri del gruppo le procedure necessarie per il suo corretto utilizzo, eventuali modifiche o aggiornamenti dovranno essere segnalati tempestivamente tramite gli appositi canali di comunicazione.

4.2.3 Aggiornamento applicazioni

L'*Amministratore* ha il compito di informare tutti i membri del gruppo di eventuali aggiornamenti disponibili per applicazioni in uso e se una loro installazione potrebbe generare incompatibilità con altri strumenti. Inoltre deve mantenere aggiornato, situato nello spazio Google Drive, il documento condiviso:

`Strumenti utilizzati`

contenente le versioni degli strumenti installati per ogni membro del gruppo.

4.2.4 Gestione account e password

L'*Amministratore* ha il compito di mantenere aggiornato il documento contenente le informazioni riguardanti gli account degli strumenti e dei siti in uso per lo svolgimento del progetto, comunicando eventualmente le variazioni ai membri del gruppo interessati. Inoltre deve monitorare l'utilizzo degli spazi per la condivisione dei file per assicurarsi che vengano utilizzati unicamente per gli scopi del progetto e per garantire che tutti i membri del gruppo possano accedervi senza limitazioni.

4.2.5 Gestione comunicazioni

L'*Amministratore* ha il compito di controllare e gestire i canali presenti nella chat Slack per mantenere un livello di conversazione accettabile e costruttivo fra tutti i membri del gruppo e per garantire che la chat non venga utilizzata impropriamente. Nello specifico l'*Amministratore* dovrà:

- assicurarsi che il linguaggio utilizzato sia rispettoso di tutti i membri del gruppo e consono all'ambiete di lavoro;
- assicurarsi che ogni canale venga utilizzato unicamente allo scopo preposto;
- se necessario, creare nuovi canali e invitarvi i membri del gruppo che li dovranno utilizzare;
- archiviare canali non più utilizzati se ritenuto opportuno;
- eliminare messaggi ritenuti inopportuni o non in linea con le indicazioni di cui sopra;
- gestire eventuali estensioni di Slack;
- farsi carico di eventuali lamentele o segnalazioni da parte dei membri del gruppo e riportarle al *Responsabile di progetto* se necessario.

4.2.6 Strumenti

4.2.6.1 Integrazioni Slack

L'*Amministratore* ha il compito di controllare e gestire le integrazioni della chat Slack:

- **Teamwork:** il canale `#task` dovrà contenere tutti i messaggi provenienti da Teamwork e relativi alla creazione, modifica e chiusura dei ticket;
- **GitHub:** il canale `#dev` dovrà contenere tutti i messaggi provenienti dai commit eseguiti sui repository git utilizzati;
- **Condivisione:** il canale `#share` dovrà contenere tutti i messaggi provenienti dalle modifiche eseguite su Dropbox o GoogleDrive.

4.2.6.2 Node.js

Node.js$_G$ è una piattaforma event-driven per il motore *JavaScript*$_G$ V8, disponibile sulle principali piattaforme, anche se maggiormente performante su sistemi operativi UNIX-like. L'indirizzo per il download e la documentazione:

`https://nodejs.org/en/download/`

4.2.6.3 npm

npm (node package manager) è il gestore di pacchetti di default utilizzato da Node.js. Questo strumento consente l'installazione e la gestione di moduli esterni che forniscono funzionalità aggiuntive al sistema node di base, risolvendo varie dipendenze. Le principali motivazioni che hanno portato il gruppo alla scelta di questo strumento sono:

- larga diffusione;
- più di 400.000 pacchetti installabili;
- ampia documentazione.

Indirizzo per il download e la documentazione:

https://www.npmjs.com

4.2.6.4 Sistemi operativi

I membri del gruppo operano sui seguenti sistemi operativi:

- *Linux*$_G$ distribuzioni: Debian v8.0, Ubuntu v16.04 e superiori;
- *Windows*$_G$ versione: 10;
- *MacOS*$_G$ versione: 10.12.

4.2.6.5 VirtualBox

VirtualBox è un'applicazione opensource e gratuita sviluppata da Oracle per l'esecuzione di macchine virtuali, permette quindi di eseguire ed utilizzare all'interno di qualsiasi computer un sistema virtualizzato con caratteristiche diverse dal sistema che lo ospita. L'applicazione è stata scelta di comune accordo con il proponente, che fornirà l'immagine di una macchina virtuale con le caratteristiche necessarie per poter essere utilizzata come ambiente di sviluppo e test.
VirtualBox è disponibile per tutti i principali sistemi operativi e la sua installazione non prevede procedure particolari. La versione utilizzata è quella stabile, oltre all'applicazione è necessario installare anche l'*Extension Pack*. Tutte le informazioni necessarie al download e all'installazione possono essere trovate al seguente indirizzo:

https://www.virtualbox.org/wiki/Downloads

4.2.6.6 FileZilla Client

FileZilla Client è un software opensource cross-platfom che permette il trasferimento di file in rete attraverso il protocollo FTP.
Il programma è disponibile per i sistemi operativi Linux, Microsoft Windows, e macOS. Tra i vari protocolli supportati, oltre all'FTP c'è l'SFTP, e l'FTP su SSL/TLS. Le principali motivazioni che hanno portato il gruppo alla scelta di questo strumento sono:

- applicazione semplice e leggera;
- già conosciuta da molti membri del gruppo.

Indirizzo per il download e la documentazione:

https://filezilla-project.org/download.php?type=client

4.2.7 Procedure

4.2.7.1 Backup

Con cadenza mensile l'*Amministratore* deve eseguire un backup su dispositivo esterno dei seguenti dati:

- *database*$_G$ *Trender*$_G$;
- repository Documenti;
- repository Codice;
- cartella Dropbox.

4.2.7.2 Configurazione VirtualBox

La macchina virtuale fornita da *RiskApp* deve essere configurata per poter consentire un agevole ambiente di sviluppo e test per il progetto.
Inizialmente si configura la VirtualBox precedentemente installata seguendo i seguenti passi:

1. aprire VirtualBox;
2. entrare nel menù `File -> Preferences -> Network -> NAT Networks`;
3. cliccare sul bottone `"+" (Add)`;
4. cliccare sul bottone `Edit`;
5. cliccare su `"Port Forwarding"`;
6. selezionare `IPv4`;
7. cliccare sul bottone `"+" (Add)`;
8. editare i campi in modo da avere le seguenti due regole:

Name	Protocol	Host IP	Host Port	Guest IP	Guest Port
Rule 1	TCP	127.0.0.1	10080	10.0.2.15	80
Rule 2	TCP	127.0.0.1	10022	10.0.2.15	22

9. chiudere le finestra cliccando su `Ok`.

In questo modo si può testare il sito dal browser del proprio PC invece che da quello della macchina virtuale, aprendo il browser e digitando l'URL `http://localhost:10080`. Successivamente si procede con la configurazione della macchina virtuale precedentemente installata come descritto nella sezione 4.2.7.3 rispettando i seguenti passi:

1. aprire Virtualbox;
2. cliccare con il tasto destro sulla macchina virtuale "Zephyrus" `Settings -> Network`;
3. impostare il campo "Connect to:" a `Nat Network`;
4. impostare il campo "Name" a `NatNetwork`;
5. chiudere le finestre cliccando sul bottone `Ok`;
6. avviare la macchina virtuale premendo sul bottone `Start`.

4.2.7.3 Installazione macchina virtuale

1. Scaricare il file con l'immagine dal seguente link:

   https://www.satellite1.info/u/Zephyrus.ova

2. aprire l'applicazione VirtualBox sul proprio computer;
3. aprire il menù `File` e selezionare `Import Appliance`;
4. selezionare il file `Zephyrus.ova` scaricato nel punto 1 e cliccare su `Next`;
5. cliccare su `Import` e attendere la fine del processo;
6. terminata l'importazione fare click destro sulla nuova macchina virtuale presente nell'elenco a sinistra e selezionare `Settings`;
7. selezionare la sezione `Network`;
8. nella tab `Adapter 1` modificare l'opzione `Attached to: NAT` e cliccare `OK`;
9. la macchina virtuale è ora pronta all'uso.

4.2.7.4 Configurazione FileZilla Client per invio file su macchina virtuale

Configurazione per l'invio dei file alla macchina virtuale Zephyrus tramite SFTP.

1. aprire FileZilla;
2. entrare nel menù `File -> gestore siti`;
3. cliccare sul bottone `Nuovo sito`;
4. editare i seguenti campi:
   - `Host: sftp://127.0.0.1`
   - `Username: admin`
   - `Password: admin`
   - `Port: 10022`
   - `Protocollo: SFTP - SSH File Transfer Protocol`
   - `Tipo di accesso: normale`
5. premere il bottone `Ok` per salvare le modifiche.

Il file js caricato ora da riskapp è:

/home/admin/ayako/frontend/static/akane/akane.js

che è un link verso:

/opt/ayako/ayako/ayako/frontend/static/akane/akane.js

ora basta solamente sovrascrivere il loro file con quello prodotto dal gruppo *Zephyrus*.

4.2.7.5 Segnalazioni e richieste

Qualora un componente del gruppo voglia inviare una segnalazione o una richiesta riguardo l'infrastruttura o gli strumenti utilizzati essa dovrà essere fatta tramite il sistema di ticketing nel seguente modo:

1. creare un nuovo task su teamwork;
2. inserire uno dei seguenti tag nel titolo del task:
   - *[NCR]* per richiedere l'inserimento di un nuovo comando personalizzato nel template LaTeX;
   - *[RICHIESTA]* per una richiesta generica;
   - *[SEGNALAZIONE]* per segnalare un problema o un'anomalia;
3. inserire il ruolo del richiedente della modifica nel corpo del task;
4. spiegare in maniera concisa la natura della richiesta e le sue motivazioni nel corpo del task;
5. assegnare il task all'*Amministratore*;
6. solamente se necessario impostare una data di scadenza del task.

Una volta ricevuto il task l'*Amministratore* potrà:

- accettare o rifiutare direttamente la richiesta motivando la scelta;
- coinvolgere se necessario il *Responsabile di progetto* assegnandogli il task in questione con eventuali commenti.

Richieste inviate in maniera diversa da quanto descritto in questa procedura non verrano prese in considerazione.

4.3 Apprendimento

4.3.1 Scopo

Lo scopo del processo di apprendimento è di garantire che ogni membro del gruppo abbia conoscenze e capacità sufficienti per svolgere le attività assegnatagli. Nel caso in cui un componente del gruppo ritenga di non essere in grado di svolgere un task dovrà segnalarlo immediatamente al *Responsabile di progetto* che dovrà organizzare le attività necessarie all'apprendimento.

4.3.1.1 Guide e documentazione

Riguardo la formazione, tutti i membri del gruppo devono procedere in modo autonomo con lo studio delle tecnologie che verranno utilizzate nel corso del progetto, prendendo come riferimento, oltre al materiale indicato nella sezione 1.4.2, anche la seguente documentazione:

- **Documentazione ufficiale React:** https://facebook.github.io/react/docs/hello-world.html;
- **Redux:**
  - **Documentazione ufficiale:** http://redux.js.org;

  - **Video tutorial ufficiali consigliati:** https://egghead.io/courses/getting-started-with-redux;

- **Web development:**
  - http://www.w3schools.com;
  - https://developer.mozilla.org/it/docs/Web;
- **LaTeX:**
  - **Guida:** http://www.guitex.org/home/it/doc;
  - **Forum:** https://www.latex-project.org;
- **Programmazione:**
  - https://www.codeschool.com;
  - **Libreria OpenLayers:** http://openlayers.org/en/latest/examples/;
  - **Tutorial JavaScript:** https://developer.mozilla.org/en-US/docs/Web/JavaScript/A_re-introduction_to_JavaScript.

4.3.1.2 Condivisione materiale

Ogni componente del gruppo è libero di utilizzare per l'apprendimento personale altro materiale oltre a quello indicato nelle *Norme di progetto*. Nel caso in cui lo ritenesse utile potrà inoltre condividere tale materiale utilizzando il canale *#apprendimento* della chat o gli strumenti di condivisione messi a disposizione (vedi sezione 4.1.5.3).

4.3.2 Procedure

4.3.2.1 Rinvio task

Nel caso in cui un componente si trovi nella situazione di non essere in grado di completare un task esso dovrà procedere come segue:

1. modificare il titolo del ticket aggiungendo il tag *[DELAY]* (vedi sezione 4.1.6.3.2);
2. aggiungere un commento al ticket spiegando le difficoltà trovate;
3. assegnare il ticket al *Responsabile di progetto*.

Una volta ricevuto il ticket il *Responsabile di progetto* dovrà valutare come procedere.

A Lista di controllo

Durante l'applicazione del *walkthrough*$_G$ ai documenti, sono riportati di seguito gli errori più frequenti. Per migliorare l'efficienza e l'efficacia da parte dei verificatori è opportuno che si basino sui seguenti controlli:

- **lingua italiana:**
  - la prima parola di una voce dell'elenco puntato inizia con la lettera maiuscola;
  - la voce finale dell'elenco puntato non termina con il punto;
  - una voce intermedia dell'elenco puntato non termina con il punto e virgola.
- **norme stilistiche:**
  - il carattere "e maiuscolo accentato" è scritto E' invece di È;
  - i due punti in grassetto dopo un termine in grassetto.
- *LaTeX*$_G$**:**
  - date e orari non scritti con i rispettivi comandi `\frmdata{GG}{MM}{YYYY}` e `\frmora{hh}{mm}`;
  - mancato utilizzo dei comandi personalizzati;
  - utilizzo scorretto delle parentesi graffe dopo i comandi LaTeX;
  - mancato aggiornamento dell'intestazione del documento dopo una modifica;
  - link e riferimenti non funzionanti o assenti.
- *UML*$_G$**:**
  - casi d'uso non proporzionati correttamente tra loro;
  - collegamenti in uscita non ad angolo retto.
- **glossario:**
  - mancata evidenziazione di termini presenti nel *Glossario*;
  - termini evidenziati impropriamente non presenti nel *Glossario*.
- **nomi dei documenti:**
  - mancata indicazione della versione di riferimento di un documento.

B Lista principali comandi LaTeX personalizzati

Per facilitare l'utilizzo e la consultazione dei comandi *LaTeX*$_G$ creati appositamente dal *gruppo*$_G$ è possibile consultare la seguente tabella che ne riassume i principali.

Comando	Descrizione	Comando	Descrizione
\frmdata{20}{05}{2017}	2017-05-20	**\revereq**	**Revisione dei requisiti**
\frmora{10}{10}	10:10	**\pdp**	*Piano di progetto*
\mailzep	zephyrus.swe@gmail.com	**\pdq**	*Piano di qualifica*
\riskapp	*RiskApp*	**\ndp**	*Norme di progetto*
\zephyrus	*Zephyrus*	**\sdf**	*Studio di fattibilità*
\mail{indirizzo@mail.it}	indirizzo@mail.it	**\adr**	*Analisi dei requisiti*
\Tullio	Professor Tullio Vardanega	**\st**	*Specifica tecnica*
\Cardin	Professor Riccardo Cardin	**\ddp**	*Definizione di prodotto*
\progetto	*DeGeOP*	**\man**	*Manuale utente*
\responsabile	*Responsabile*	**\gl**	*Glossario*
\responsabilediprogetto	*Responsabile di progetto*	**\ldp**	*Lettera di presentazione*
\amministratore	*Amministratore*	**\pdpv**	*Piano di progetto v4.0.0*
\amministratori	*Amministratori*	**\pdqv**	*Piano di qualifica v4.0.0*
\analista	*Analista*	**\ndpv**	*Norme di progetto v3.0.0*
\analisti	*Analisti*	**\sdfv**	*Studio di fattibilità v1.0.0*
\progettista	*Progettista*	**\adrv**	*Analisi dei requisiti v3.0.0*
\progettisti	*Progettisti*	**\stv**	*Specifica tecnica v2.0.0*
\programmatore	*Programmatore*	**\ddpv**	*Definizione di prodotto v2.0.0*
\programmatori	*Programmatori*	**\manutv**	*Manuale utente v2.0.0*
\verificatore	*Verificatore*	**\manmanv**	*Manuale manutentore v2.0.0*
\verificatori	*Verificatori*	**\glv**	*Glossario v2.0.0*
\vduei	*VerbaleInterno_2_20161220*	**\revacc**	**Revisione di accettazione**
\vtrei	*VerbaleInterno_3_20161227*	**\revaqual**	**Revisione di qualifica**
\vquattroi	*VerbaleInterno_4_20170103*	**\revprog**	**Revisione di progettazione**
\vunoe	*VerbaleEsterno_1_20161203*	**\itemVI**	Cod decisioni verbali int
\vduee	*VerbaleEsterno_2_20161227*	**\itemVE**	Cod decisioni verbali est
\js	JavaScript	**\jsv**	JavaScript ES6
\vtree	*VerbaleEsterno_3_20170209*	**\vquattroi**	*VerbaleInterno_4_20170103*
\vcinquei	*VerbaleInterno_5_201700206*	**\hicode**	`NomeComando`

Tabella 4: Principali comandi LaTeX personalizzati. Riferita nelle sezioni: 3.1.4.5, 3.3.2