

MashUp

Software Engineering Group

info@mashup-unipd.it

Informazioni Documento

Nome documento	<i>Norme di Progetto</i>
Versione	<i>v5.0.0</i>
Data redazione	2014-12-10
Redattori	Roetta Marco
Verificatori	Tesser Paolo
Approvazione	Cusinato Giacomo
Lista distribuzione	<i>MashUp</i>
Uso	Interno

Sommario

Questo documento vuol definire le norme volte alla regolamentazione delle attività del gruppo *MashUp* necessarie al processo di sviluppo del progetto BDSMApp.

Diario Revisioni

Modifica	Autore & Ruolo	Data	Versione
<i>Approvazione documento</i>	Cusinato Giacomo <i>Responsabile di Progetto</i>	2015-05-11	v5.0.0
<i>Verifica del documento</i>	Tesser Paolo <i>Verificatore</i>	2015-05-11	v4.1.0
<i>Ampliamento lista di controllo con errori tipici UML e codice in A</i>	Roetta Marco <i>Amministratore di Progetto</i>	2015-05-08	v4.0.1
<i>Approvazione documento</i>	Cusinato Giacomo <i>Responsabile di Progetto</i>	2015-04-09	v4.0.0
<i>Verifica del documento</i>	Tesser Paolo <i>Verificatore</i>	2015-04-08	v3.1.0
<i>Inserimento strumento per la verifica Cloc in 3.2.4.3</i>	Roetta Marco <i>Amministratore di Progetto</i>	2015-04-07	v3.0.5
<i>Aggiunti strumenti di test in Processi di Verifica: Wercker, Karma, Jasmine, Protractor, Testbed GAE</i>	Roetta Marco <i>Amministratore di Progetto</i>	2015-04-07	v3.0.4
<i>Inserita appendice Configurazione File Python in C</i>	Roetta Marco <i>Amministratore di Progetto</i>	2015-04-06	v3.0.3
<i>Inserita appendice Configurazione File AngularJS in B</i>	Roetta Marco <i>Amministratore di Progetto</i>	2015-04-03	v3.0.2
<i>Stesa sezione codifica e definizione prodotto in Processi di Sviluppo (Primari)</i>	Roetta Marco <i>Amministratore di Progetto</i>	2015-04-01	v3.0.1
<i>Approvazione documento</i>	Santacatterina Luca <i>Responsabile di Progetto</i>	2015-03-02	v3.0.0
<i>Verifica del documento</i>	Faccin Nicola <i>Verificatore</i>	2015-02-26	v2.1.0
<i>Inserito paragrafo sulle norme di codifica dei processi di sviluppo</i>	Carnovalini Filippo <i>Amministratore di Progetto</i>	2015-02-25	v2.0.4
<i>Inserito paragrafo sul formalismo di classificazione dei test</i>	Carnovalini Filippo <i>Amministratore di Progetto</i>	2015-02-24	v2.0.3
<i>Inserito attività riguardante la gestione della rotazione dei ruoli di progetto</i>	Carnovalini Filippo <i>Amministratore di Progetto</i>	2015-02-23	v2.0.2
<i>Inseriti diagrammi per le procedure dei processi primari</i>	Carnovalini Filippo <i>Amministratore di Progetto</i>	2015-02-19	v2.0.1
<i>Approvazione documento</i>	Ceccon Lorenzo <i>Responsabile di Progetto</i>	2015-02-06	v2.0.0
<i>Verifica del documento</i>	Tesser Paolo <i>Verificatore</i>	2015-02-05	v1.1.0
<i>Redazione attività e norme di codifica e progettazione</i>	Faccin Nicola <i>Amministratore di Progetto</i>	2015-02-04	v1.0.4
<i>Aggiunte strumento di chat Slack</i>	Faccin Nicola <i>Amministratore di Progetto</i>	2015-02-03	v1.0.3
<i>Aggiunte attività di formazione nei processi organizzativi</i>	Faccin Nicola <i>Amministratore di Progetto</i>	2015-02-02	v1.0.2
<i>Aggiornata struttura repository per le presentazioni</i>	Faccin Nicola <i>Amministratore di Progetto</i>	2015-02-01	v1.0.1

<i>Approvazione del documento</i>	Cusinato Giacomo <i>Responsabile di Progetto</i>	2014-12-23	v1.0.0
<i>Verifica del documento</i>	Roetta Marco <i>Verificatore</i>	2014-12-20	v0.2.0
<i>Correzione del documento</i>	Santacatterina Luca <i>Amministratore</i>	2014-12-19	v0.1.1
<i>Verifica del documento</i>	Roetta Marco <i>Verificatore</i>	2014-12-19	v0.1.0
<i>Completata stesura capitolo Processi Organizzativi</i>	Tesser Paolo <i>Amministratore</i>	2014-12-18	v0.0.8
<i>Completata stesura capitolo Processi di Supporto</i>	Santacatterina Luca <i>Amministratore</i>	2014-12-16	v0.0.7
<i>Iniziata stesura capitolo Processi Organizzativi</i>	Tesser Paolo <i>Amministratore</i>	2014-12-12	v0.0.6
<i>Completata stesura capitolo Processi Primari</i>	Tesser Paolo <i>Amministratore</i>	2014-12-11	v0.0.5
<i>Inizio stesura capitolo Processi Primari</i>	Tesser Paolo <i>Amministratore</i>	2014-12-09	v0.0.4
<i>Iniziata stesura capitolo Processi di Supporto</i>	Santacatterina Luca <i>Amministratore</i>	2014-12-09	v0.0.3
<i>Stesura capitolo Introduzione</i>	Santacatterina Luca <i>Amministratore</i>	2014-12-08	v0.0.2
<i>Creazione struttura documento</i>	Tesser Paolo <i>Amministratore</i>	2014-12-08	v0.0.1

Indice

1	Introduzione	1
1.1	Scopo del documento	1
1.2	Scopo del prodotto	1
1.3	Glossario	1
1.4	Riferimenti	1
1.4.1	Informativi	1
1.4.2	Normativi	2
2	Processi Primari	3
2.1	Processo di fornitura	3
2.1.1	Attività	3
2.1.1.1	Accettazione	3
2.1.1.1.1	Discussione e scelta del capitolo	3
2.1.1.1.2	Studio di Fattibilità	3
2.1.1.2	Preparazione della risposta	3
2.1.1.2.1	Definizione e preparazione della proposta	3
2.1.1.3	Pianificazione	4
2.1.1.3.1	Scelta del modello di ciclo di vita	4
2.1.1.3.2	Sviluppo e documentazione del Piano di Progetto	4
2.2	Processo di sviluppo	4
2.2.1	Attività	4
2.2.1.1	Analisi dei requisiti	4
2.2.1.2	Progettazione	4
2.2.1.2.1	Specifica Tecnica	4
2.2.1.2.2	Definizione di prodotto	5
2.2.1.3	Codifica	5
2.2.2	Procedure	5
2.2.2.1	Inserimento requisiti nella base di dati	5
2.2.2.2	Inserimento casi d'uso nella base di dati	6
2.2.2.3	Aggiunta template di struttura file in WebStorm	6
2.2.3	Norme	10
2.2.3.1	Classificazione requisiti	10
2.2.3.2	Classificazione casi d'uso	10
2.2.3.3	Codifica file	11
2.2.3.4	Nomi e norme di codifica	11
2.2.3.5	Ricorsione	12
2.2.4	Strumenti	12
2.2.4.1	RequirementsTool	12
2.2.4.2	PhpStorm	12
2.2.4.3	PyCharm	12
2.2.4.4	WebStorm	12
2.2.4.5	Google App Engine	12
2.2.4.6	Yeoman	12

3 Processi di Supporto	14
3.1 Processo di documentazione	14
3.1.1 Attività	14
3.1.1.1 Documentazione	14
3.1.2 Procedure	14
3.1.2.1 Gestione dei documenti	14
3.1.2.1.1 Creazione di un nuovo documento	14
3.1.2.1.2 Avanzamento di un documento	14
3.1.2.2 Gestione del glossario	14
3.1.2.2.1 Inserimento di un termine	15
3.1.2.2.2 Eliminazione di un termine	15
3.1.2.2.3 Inserimento termine nei documenti	15
3.1.3 Norme	15
3.1.3.1 Progettazione e sviluppo dei documenti	15
3.1.3.2 Versionamento	15
3.1.3.3 Template	16
3.1.3.4 Struttura dei documenti	16
3.1.3.4.1 Prima pagina	16
3.1.3.4.2 Diario delle revisioni	17
3.1.3.4.3 Indici	17
3.1.3.4.4 Formattazione di una pagina	17
3.1.3.5 Suddivisione sezioni documenti	17
3.1.3.5.1 Studio di Fattibilità	17
3.1.3.5.2 Norme di Progetto	17
3.1.3.5.3 Piano di Progetto	17
3.1.3.5.4 Piano di Qualifica	18
3.1.3.5.5 Analisi dei Requisiti	18
3.1.3.5.6 Specifica Tecnica	18
3.1.3.5.7 Definizione di Prodotto	18
3.1.3.5.8 Glossario	18
3.1.3.6 Norme tipografiche	18
3.1.3.6.1 Stili di testo	18
3.1.3.6.2 Punteggiatura	18
3.1.3.6.3 Composizione del testo	19
3.1.3.6.4 Formati ricorrenti	19
3.1.3.7 Componenti grafiche	19
3.1.3.7.1 Immagini	20
3.1.3.7.2 Diagrammi	20
3.1.4 Strumenti	20
3.1.4.1 Script	20
3.2 Processo di verifica	21
3.2.1 Attività	21
3.2.1.1 Analisi statica	21
3.2.1.2 Analisi dinamica	21
3.2.1.3 Gestione anomalie	22
3.2.1.4 Test	22
3.2.1.4.1 Test di unità	22
3.2.1.4.2 Test di integrazione	22
3.2.1.4.3 Test di sistema	22
3.2.1.4.4 Test di regressione	22

3.2.1.4.5	Test di validazione	22
3.2.1.5	Tracciamento	23
3.2.2	Procedure	23
3.2.2.1	Procedure d'assegnazione delle anomalie	23
3.2.2.2	Configurazione sistema di Continuous Integration . .	23
3.2.3	Norme	26
3.2.3.1	Priorità risoluzione anomalie	26
3.2.4	Strumenti	26
3.2.4.1	Correzione ortografica	26
3.2.4.2	Calcolo indice Gulpease	26
3.2.4.3	Cloc	26
3.2.4.4	Wercker	26
3.2.4.5	Karma	26
3.2.4.6	Jasmine	26
3.2.4.7	Protractor	27
3.2.4.8	Python unittest	27
3.2.4.9	GAE testbed	27
4 Processi Organizzativi		28
4.1	Gestione dei processi	28
4.1.1	Attività	28
4.1.1.1	Gestione delle Comunicazioni	28
4.1.1.1.1	Mail	28
4.1.1.1.2	Comunicazioni interne	28
4.1.1.1.3	Comunicazioni esterne	28
4.1.1.2	Gestione delle Riunioni	28
4.1.1.2.1	Riunioni interne	28
4.1.1.2.2	Riunioni esterne	29
4.1.1.3	Gestione del sistema di ticketing	29
4.1.1.4	Gestione delle Milestone	30
4.1.1.5	Gestione dei task	30
4.1.1.6	Gestione dello svolgimento dei task	30
4.1.1.7	Gestione delle change request	30
4.1.1.8	Gestione della commit	30
4.1.1.8.1	Meccanismo di aggancio commit-task	31
4.1.1.9	Gestione della rotazione dei ruoli	31
4.1.2	Procedure	31
4.1.2.1	Procedura di generazione di una milestone	31
4.1.2.2	Procedura d'assegnazione dei task	31
4.1.2.3	Procedura per il reperimento dell'id del task	32
4.1.2.4	Procedura di svolgimento dei task assegnati	32
4.1.2.5	Procedura di svolgimento di una change request . .	34
4.1.2.6	Procedura di rilevazione dei rischi	36
4.1.3	Norme	36
4.1.3.1	Ruoli di Progetto	36
4.1.3.1.1	Responsabile di Progetto	36
4.1.3.1.2	Amministratore	38
4.1.3.1.3	Analista	38
4.1.3.1.4	Progettista	38
4.1.3.1.5	Programmatore	38
4.1.3.1.6	Verificatore	39

4.1.3.2	Formato dei task	39
4.1.3.3	Formato delle change request	40
4.1.4	Strumenti	40
4.1.4.1	Asana	40
4.1.4.2	Astah	40
4.1.4.3	ProjectLibre	41
4.1.4.4	NetSons	41
4.1.4.5	Skype	41
4.1.4.6	WhatsApp	41
4.1.4.7	Slide	41
4.1.4.8	Slack	41
4.2	Gestione delle infrastrutture	42
4.2.1	Attività	42
4.2.1.1	Gestione del Repository	42
4.2.1.2	Gestione dei Git Hooks	42
4.2.1.3	Gestione del template del messaggio di commit	42
4.2.2	Procedure	43
4.2.2.1	Installazione e manutenzione Git Hooks	43
4.2.2.2	Installazione e manutenzione messaggio di commit	43
4.2.3	Norme	43
4.2.3.1	Repository	43
4.2.3.1.1	Nomi dei file in doc_BDSM_App	43
4.2.3.1.2	Struttura di doc_BDSM_App	46
4.2.3.1.3	Nomi dei file in src_BDSM_App_client	46
4.2.3.1.4	Struttura di src_BDSM_App_client	46
4.2.3.1.5	Nomi dei file in src_BDSM_App_server	47
4.2.3.1.6	Struttura di src_BDSM_App_server	47
4.2.3.1.7	Modello di sviluppo	47
4.2.3.1.8	Template messaggio di commit	47
4.2.4.1	Strumenti	48
4.2.4.1.1	Git	48
4.2.4.1.2	GitHub	48
4.2.4.1.3	Git Hooks	48
4.2.4.1.4	Google Drive	48
4.2.4.1.5	Sistema Operativo	48
4.3	Formazione dei membri del team	49
4.3.1	Attività	49
4.3.1.1	Incontri con il proponente	49
A	Lista di controllo	50
B	Configurazione File AngularJS	51
B.1	View	51
B.2	Page Controller	51
B.3	Route Controller	52
B.4	Services Model	52
C	Configurazione File Python	55
C.1	Configurazione costrutto switch	55
C.2	Nome variabili	55

Elenco delle figure

1	Diagramma di attività - procedura d'inserimento dei requisiti	7
2	Diagramma di attività - procedura d'inserimento dei casi d'uso	8
3	Diagramma di attività - procedura di aggiunta template in WebStorm	9
4	Diagramma di attività - procedura assegnazione anomalie	24
5	Diagramma di attività - procedura configurazione continuous integration	25
6	Diagramma di attività - procedura d'assegnazione di un task	33
7	Reperimento dell'id del task	34
8	Diagramma di attività - procedura di svolgimento dei task assegnati	35
9	Diagramma di attività - procedura di svolgimento di una change request	37
10	Template Task su Asana	40
11	Template Change Request su Asana	41
12	Diagramma di attività - installazione/aggiornamento dei Git Hooks	44
13	Diagramma di attività - installazione/aggiornamento del template del messaggio per la commit	45

1 Introduzione

1.1 Scopo del documento

Questo documento vuole definire tutte le norme che i membri del gruppo *MashUp* dovranno rispettare durante lo svolgimento del progetto BDSMApp.

Tutti i membri sono tenuti a leggere il documento e a seguire rigorosamente le norme descritte.

Ogni membro del gruppo dovrà segnalare all'*Amministratore di Progetto* qualsiasi richiesta per la modifica o l'introduzione di nuove norme.

Successivamente l'*Amministratore di Progetto* coinvolgerà i membri del gruppo per discuterne e avrà a disposizione la possibilità di approvare o negare la richiesta. Tutte le norme raccolte dal seguente documento si riferiscono al progetto BDSMApp.

Le seguenti norme permetteranno di ottenere una maggiore uniformità dei documenti prodotti, porteranno ad un incremento dell'efficienza e della efficacia e ridurranno la possibilità di commettere errori.

In particolare si vuole definire:

- identificazione ruolo e mansione per ogni singolo componente del gruppo;
- strumenti e metodi per la comunicazione interna ed esterna al gruppo;
- strumenti e metodi per la stesura di tutti i documenti;
- tempi e modalità di lavoro per ogni singola fase;
- ambienti di sviluppo e tools di versionamento.

1.2 Scopo del prodotto

Lo scopo del prodotto è di creare una nuova infrastruttura che permetta di interrogare Big Data recuperati dai social network, quali: Facebook, Twitter, Instagram. L'applicazione sarà composta da due parti:

- consultazione e interrogazione con interfaccia web per utente;
- servizi web REST interrogabili.

1.3 Glossario

Al fine di evitare ogni ambiguità relativa al linguaggio usato nei documenti viene allegato il *Glossario v3.0.0*. Esso ha lo scopo di definire ed analizzare tutti i termini tecnici del progetto e di fugare eventuali ambiguità fornendo un'accurata descrizione. Tutte le occorrenze di tali termini nei documenti verranno contrassegnate con una "G" a pedice.

1.4 Riferimenti

1.4.1 Informativi

- **Asana:** <http://asana.com>
- **BDSMApp:** <http://bit.ly/BDSMapp>
- **GitHub:** <http://github.com>

- **Yeoman:** <http://yeoman.io/>
- **Inspection e Walkthrough:** <http://www.math.unipd.it/~tullio/IS-1/2004/Approfondimenti/Inspections-Walkthroughs.pdf>
- **Amministrazione di progetto:** <http://www.math.unipd.it/~tullio/IS-1/2014/Dispense/P06.pdf>
- **Costrutto switch:** <http://bytebaker.com/2008/11/03/switch-case-statement-in-python/>
- **Unit Test Angular:** <https://docs.angularjs.org/guide/unit-testing>
- **Unit Test Python:** <https://docs.python.org/3/library/unittest.html>
- **Unit Test Python GAE:** <https://cloud.google.com/appengine/docs/python/tools/localunittesting>
- **Cloc:** <http://cloc.sourceforge.net/>

1.4.2 Normativi

- **ISO 8601:** http://it.wikipedia.org/wiki/ISO_8601
- **Branching Git Flow:** <http://nvie.com/posts/a-successful-git-branching-model/>

2 Processi Primari

Nella sezione dei Processi Primari sono descritti i processi e le attività che servono alle parti prime durante il ciclo di vita del software.

Queste parti sono l'acquirente, il fornitore, lo sviluppatore, l'operatore e il manutentore del prodotto software.

Il gruppo *MashUp* si occuperà solo dei Processi di Fornitura e di Sviluppo. Il Processo di Acquisizione spetterà al proponente_G e al committente_G del capitolato scelto, mentre il Processo di Manutenzione non potrà essere attuato per vincoli temporali. Nonostante questo l'applicativo BDSMApp e la documentazione fornita con esso dovranno garantire la possibilità futura di procedere con le attività presenti in questo processo a discrezione del proponente_G.

2.1 Processo di fornitura

2.1.1 Attività

2.1.1.1 Accettazione

2.1.1.1.1 Discussione e scelta del capitolato Il *Responsabile di Progetto* ha il compito di organizzare gli incontri iniziali per permettere ai componenti del gruppo di discutere e esaminare i capitolati selezionati dal committente_G.

Dalle riunioni emergerà la scelta effettuata dal team.

Le valutazioni che hanno portato a prendere questa decisione verranno documentate in dei verbali.

2.1.1.1.2 Studio di Fattibilità Agli *Analisti* spetta il compito di redigere lo *Studio di Fattibilità v1.0.0*, basandosi sui verbali citati in precedenza. Per realizzare il documento dovranno essere presi in considerazione i seguenti punti per ogni capitolato, con maggiore dettaglio per quello preso in carico.

- **dominio tecnologico e applicativo:** si valutano le conoscenze attuali del team riguardo le tecnologie richieste;
- **interesse strategico:** si valuta quanto interesse abbia il team ad apprendere gli strumenti che serviranno a realizzare il capitolato in questione;
- **rischi possibili:** si analizzano i possibili punti critici in merito anche alle valutazioni effettuate nei punti precedenti.

2.1.1.2 Preparazione della risposta

2.1.1.2.1 Definizione e preparazione della proposta I membri del gruppo *MashUp* dovranno redigere i seguenti documenti:

- *Studio di Fattibilità v1.0.0*
- *Analisi dei Requisiti v4.0.0*
- *Piano di Progetto v5.0.0*
- *Piano di Qualifica v4.0.0*

- *Norme di Progetto v5.0.0*

In allegato a essi si dovrà consegnare una *lettera di presentazione*.

Starà quindi al committente_G e al proponente valutare l'offerta ricevuta e decidere se accettare o meno la proposta fatta.

2.1.1.3 Pianificazione

2.1.1.3.1 Scelta del modello di ciclo di vita Il *Responsabile di Progetto* avrà il compito di scegliere, se non indicato dal proponente_G, il modello di ciclo di vita adatto per lo sviluppo del prodotto richiesto.

2.1.1.3.2 Sviluppo e documentazione del Piano di Progetto Il *Responsabile di Progetto* dovrà eseguire una serie di compiti per delineare i lavori che i membri del team dovranno andare ad eseguire, calcolandone i costi e le tempistiche. Il tutto verrà documentato nel *Piano di Progetto v5.0.0*.

2.2 Processo di sviluppo

2.2.1 Attività

2.2.1.1 Analisi dei requisiti Dopo avere redatto lo *Studio di Fattibilità*, agli *Analisti* spetterà il compito di scrivere il documento *Analisi dei Requisiti v4.0.0*. L'obiettivo sarà quello di cercare e produrre dei requisiti a partire dalle informazioni a disposizione del team.

Le risorse che potranno essere utilizzate per questo scopo provengono dal capitolo d'appalto e dagli incontri con il proponente_G e con il committente_G.

2.2.1.2 Progettazione

2.2.1.2.1 Specifica Tecnica Compito dei *Progettisti* sarà descrivere la progettazione ad alto livello dell'architettura dell'applicazione e dei singoli componenti nella *Specifica Tecnica v2.0.0*. Devono inoltre progettare gli opportuni test di integrazione.

Diagrammi UML_G

Si utilizzerà il linguaggio UML_G per definire:

- Diagrammi delle classi;
- Diagrammi dei package_G;
- Diagrammi delle attività;
- Diagrammi di sequenza;

Per la stesura, verranno seguite le seguenti convenzioni:

- **Standard:** standard UML_G 2.0;
- **Lingua:** la lingua utilizzata è l'inglese.

Design pattern_G

La descrizione dei design pattern_G sarà a carico dei *Progettisti* che sceglieranno quelli più adatti al contesto dell'applicativo per migliorarne l'efficienza. Ogni uno di questi sarà accompagnato da una breve descrizione e da un diagramma che ne esemplificano il funzionamento e la struttura.

Tracciamento componenti

Ogni requisito deve essere tracciato al componente che lo soddisfa. In questo modo sarà possibile misurare il progresso nell'attività di progettazione e garantire che ogni componente soddisfi almeno un requisito.

Test di integrazione

I *Progettisti* dovranno definire delle classi di verifica, necessarie per verificare che i componenti del sistema funzionino correttamente.

2.2.1.2.2 Definizione di prodotto Compito dei *Progettisti* sarà descrivere la progettazione a basso livello dell'architettura dell'applicazione nella *Definizione di Prodotto v2.0.0*, andando ad ampliare quanto descritto nella *Specifica Tecnica v2.0.0*.

Diagrammi UML_G

Si utilizzerà il linguaggio UML_G per definire:

- Diagrammi delle classi;
- Diagrammi delle attività;
- Diagrammi di sequenza;

Tracciamento classi

Ogni requisito deve essere tracciato dalla classe che lo soddisfa. In questo modo sarà possibile misurare il progresso nell'attività di progettazione e garantire che ogni classe soddisfi almeno un requisito.

Test di unità

I *Progettisti* dovranno definire i test di unità necessari a verificare che i singoli componenti del sistema funzionino nel modo previsto.

2.2.1.3 Codifica Le attività di codifica seguiranno svolte dai *Programmatori* dovranno basarsi su quanto descritto nella *Definizione di Prodotto v2.0.0* e seguire le linee guida presenti in questo documento.

2.2.2 Procedure

2.2.2.1 Inserimento requisiti nella base di dati Tramite la base di dati creata appositamente è possibile poter inserire nuovi requisiti (figura 1), l'*Analista* deve eseguire le seguenti procedure ordinatamente:

1. Accedere alla base usando il link <http://gestionale.mashup-unipd.it> ed inserire le credenziali fornite;
2. recarsi nella pagina "Gestione requisiti";
3. nella colonna di sinistra selezionare in quale punto dell'albero inserire il nuovo requisito;

4. cliccare con il tasto destro nel punto identificato e selezionare dal menù a tendina "Nuovo";
5. compilare il form_G con i dati richiesti;
6. premere salva alla fine del lavoro.

2.2.2.2 Inserimento casi d'uso nella base di dati Tramite la base di dati creata appositamente è possibile poter inserire nuovi casi d'uso (figura 3), l'*Analista* deve eseguire le seguenti procedure ordinatamente:

1. accedere alla base usando il link <http://gestionale.mashup-unipd.it> ed inserire le credenziali fornite;
2. recarsi nella pagina "Gestione casi d'uso";
3. nella colonna di sinistra selezionare in quale punto dell'albero inserire il nuovo caso d'uso;
4. cliccare con il tasto destro nel punto identificato e selezionare dal menù a tendina "Nuovo";
5. compilare il form_G con i dati richiesti;
6. premere salva alla fine del lavoro.

2.2.2.3 Aggiunta template di struttura file in WebStorm Di seguito vengono elencati i passi che ogni *Programmatore* dovrà compiere la prima volta che andrà a lavorare sul codice relativo al client dell'applicazione.

1. aprire l'applicativo WebStorm;
2. aprire il pannello delle preferenze;
3. selezionare la voce **File and Code Templates**;
4. premere sul tasto + in verde che dice "Create Template_G";
5. dare un nome significativo a seconda del tipo che si vuole creare;
6. copiare uno dei template riportati nell'appendice B;
7. salvare le nuove aggiunte;
8. ripetere dal passo 4 finché non saranno terminati i template_G da inserire.

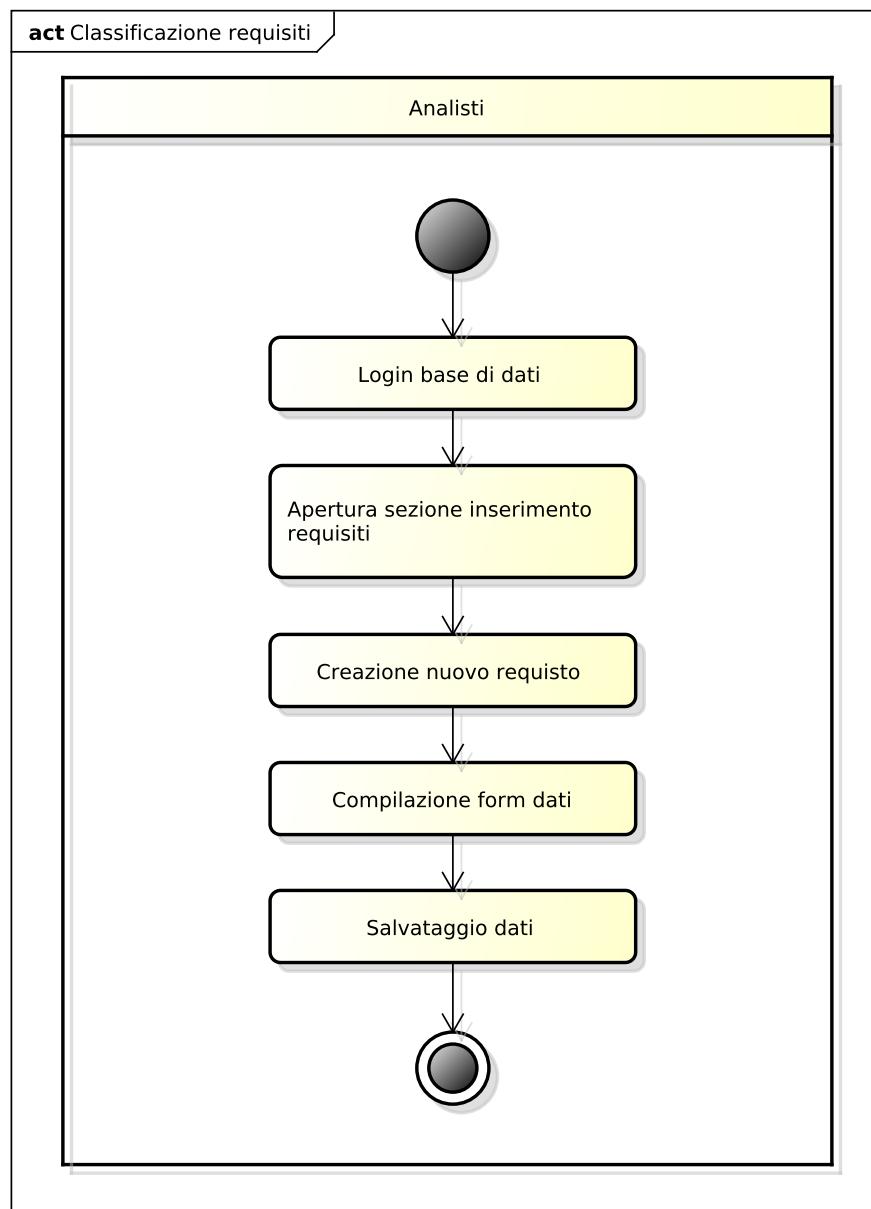


Figura 1: Diagramma di attività - procedura d'inserimento dei requisiti

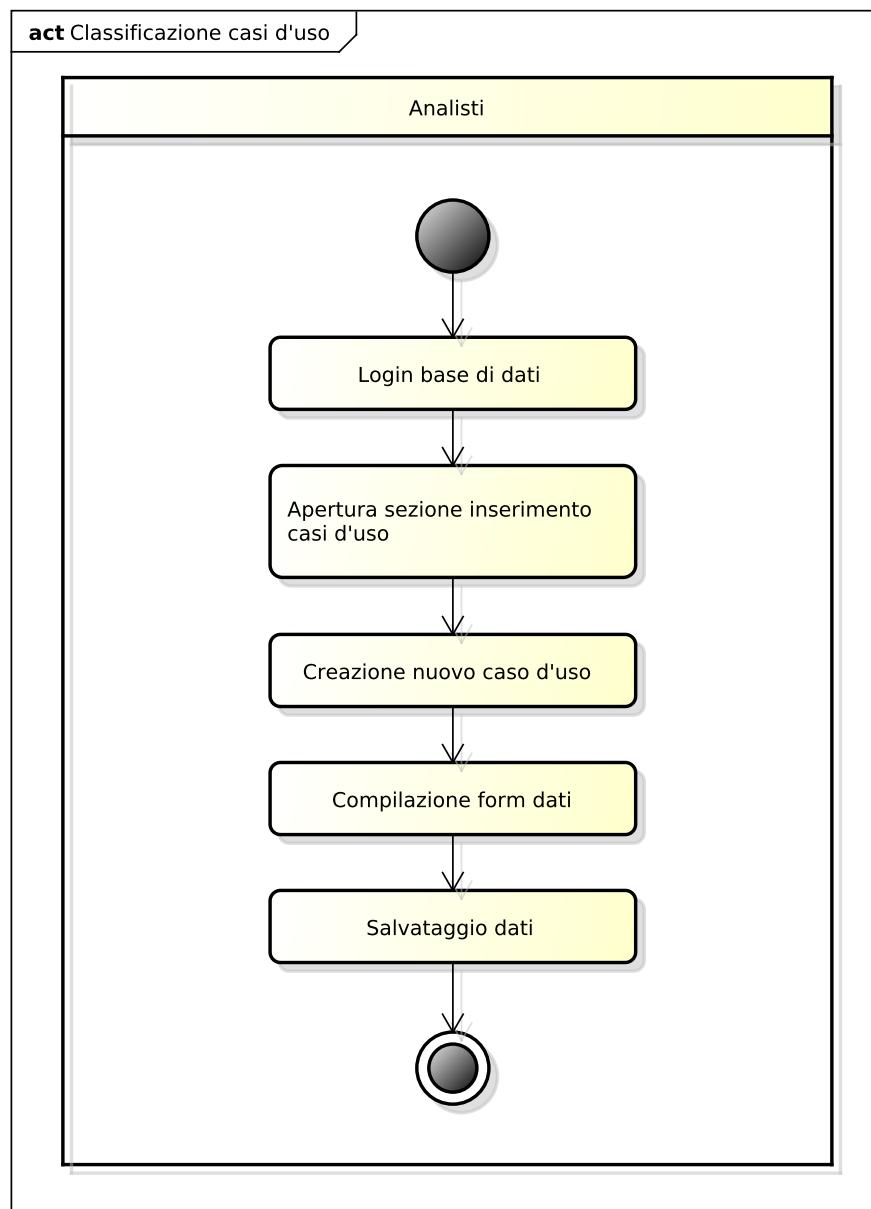


Figura 2: Diagramma di attività - procedura d'inserimento dei casi d'uso

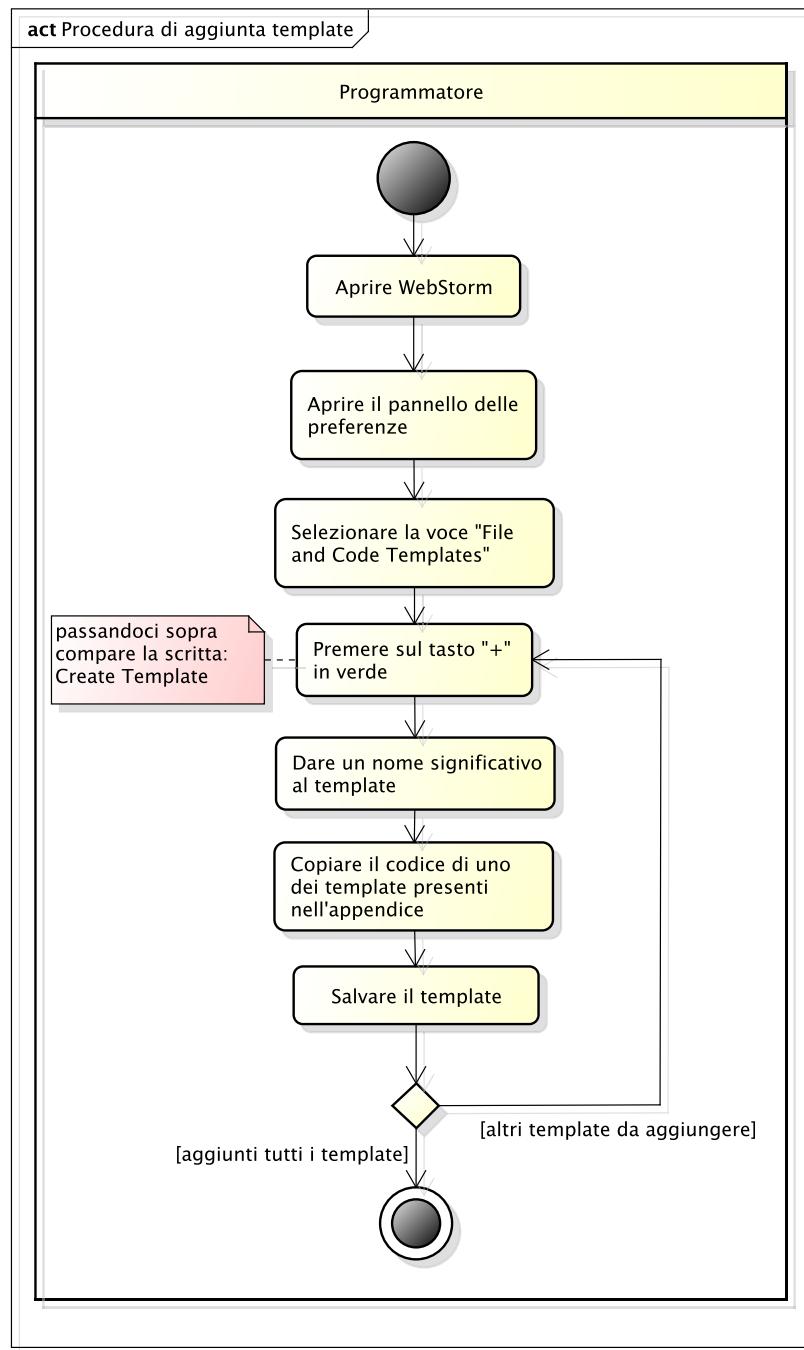


Figura 3: Diagramma di attività - procedura di aggiunta template in WebStorm

2.2.3 Norme

2.2.3.1 Classificazione requisiti I requisiti prodotti dovranno essere classificati a seconda del tipo e dell'importanza, seguendo la seguente notazione:

$$R[Importanza][Tipo][Codice]$$

- **Importanza:** i valori che può assumere sono:
 - O: requisito obbligatorio;
 - D: requisito desiderabile;
 - F: requisito facoltativo(opzionale).
- **Tipo:** i valori che può assumere sono:
 - F: requisito funzionale;
 - P: requisito prestazionale;
 - Q: requisito di qualità;
 - V: requisito di vincolo.
- **Codice:** è il codice gerarchico e univoco del vincolo espresso nella forma X.X.X dove X è un valore numerico.

Ogni requisito inoltre dovrà contenere le seguenti informazioni:

- **Descrizione:** descrizione del requisito il meno possibile ambigua;
- **Fonte:** la scelta può ricadere tra:
 - **capitolato:** è il requisito che viene trovato dalle specifiche del capitolato di appalto;
 - **interno:** è il requisito che viene trovato dagli *Analisti* durante un'analisi più approfondita;
 - **caso d'uso:** è il requisito che viene trovato a partire da uno o più casi d'uso. In tal caso andrà specificato il codice del caso d'uso a cui si riferisce;
 - **verbale:** è il requisito che viene trovato dagli incontri con il proponente_G o da incontri interni tra i membri del gruppo.

2.2.3.2 Classificazione casi d'uso I casi d'uso devono essere divisi in ordine gerarchico secondo il seguente schema:

$$UC[Codice]$$

- **Codice:** è il codice gerarchico e univoco per identificare ogni caso d'uso.

Per ogni caso d'uso dovranno essere presenti anche le seguenti informazioni:

- **titolo:** fornire un titolo riassuntivo dell'operazione che il caso d'uso intende modellare;
- **descrizione:** fornire una breve descrizione del caso d'uso che sia il meno ambigua possibile;

- **attori principali:** elenco degli attori principali coinvolti nel caso d'uso;
- **attori secondari:** elenco degli attori secondari coinvolti nel caso d'uso;
- **scenari principali:** descrizione dei possibili scenari principali;
- **scenari alternativi:** descrizione dei possibili scenari secondari;
- **pre-condizioni:** condizione di partenza sempre vera all'inizio del caso d'uso;
- **flusso degli eventi:** ordine di esecuzione dei casi d'uso figli;
- **inclusioni:** spiegazione di tutte le inclusioni se presenti;
- **estensioni:** spiegazione di tutte le estensioni se presenti;
- **generalizzazioni:** spiegazione di tutte le generalizzazioni, se presenti;;
- **post-condizioni:** condizione finale sempre vera alla fine dell'esecuzione del caso d'uso.

Alcuni dei campi potranno essere assenti nel caso dovessero risultare inutilizzati.

2.2.3.3 Codifica file Tutti i file che vengono creati, contenenti sia codice sia documentazione, devono essere codificati tramite UTF_G-8 senza BOM_G.

Se verrà a verificarsi la necessità di un cambiamento il *Responsabile di Progetto* ne dovrà approvare le modifiche.

2.2.3.4 Nomi e norme di codifica Il codice dovrà seguire determinate linee guida a seconda del linguaggio.

- **Python:** Google Python Style Guide <https://google-styleguide.googlecode.com/svn/trunk/pyguide.html>;
- **Javascript:** Google JavaScript Style Guide http://google-styleguide.googlecode.com/svn/trunk/javascriptguide.xml?showone=Method_and_property_definitions#Method_and_property_definitions;
- **AngularJS:** Google AngularJS Style Guide <https://google-styleguide.googlecode.com/svn/trunk/angularjs-google-style.html>.
- ci dovrà essere un header su ciascun file che conterrà le seguenti informazioni:
 - percorso del file e nome del file;
 - cognome e nome dell'autore;
 - data di creazione;
 - email dell'autore;
 - dovranno inoltre essere inserite per ogni modifica che verrà effettuata: la successiva versione generata dall'avanzamento, la data, una breve descrizione e l'autore.

Esempio relativo ai file in Python, ma che dovrà essere simile anche per i file javascript_G:

```

# File: nome_del_file
# Author: Cognome Nome
# Email: cognome.nome@mashup-unipd.it
# Date: AAAA-MM-DD
# Modify:
# =====
# Version      Date        Author          Description
# =====
# x.y.z        AAAA-MM-DD   Cognome Nome  Description
# -----

```

- i nomi delle variabili dovranno essere in inglese;
- i commenti dovranno essere in italiano

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

2.2.4 Strumenti

2.2.4.1 RequirementsTool Il gruppo ha creato un applicativo web che gestisce l'immagazzinamento e il tracciamento dei requisiti e delle fonti attraverso l'inserimento in appositi form_G.

Il servizio viene offerto nel servizio di hosting del team e ci si può accedere attraverso le API_G Key di Asana_G.

L'applicativo è anche in grado di tracciare tutti i task_G che sono assegnati al membro loggato e consentire la gestione del tempo di esecuzione di ciascuno di essi.

Lo strumento è scritto in PHP.

2.2.4.2 PhpStorm L'IDE JetBrains PhpStorm edizione professional, disponibile tramite licenza studentesca, viene utilizzato per scrivere il codice PHP relativo all'applicativo descritto nella sezione 2.2.4.1

2.2.4.3 PyCharm L'IDE_G JetBrains PyCharm edizione professional, disponibile tramite licenza studentesca, viene utilizzato per scrivere codice in Python relativo al back-end dell'applicativo.

2.2.4.4 WebStorm L'IDE_G JetBrains WebStorm edizione professional, disponibile tramite licenza studentesca, viene utilizzato per il front-end, quindi per il codice in Javascript e HTML

2.2.4.5 Google App Engine È una piattaforma cloud per lo sviluppo e l'hosting di applicazioni, sarà utilizzata principalmente per il back-end.

2.2.4.6 Yeoman Al fine di creare la struttura di base per l'applicativo web front-end verrà usato il software di scaffolding Yeoman, con il generatore generator-angular. Verranno selezionate tra le opzioni proposte di non utilizzare Sass, di includere Bootstrap e tutti i moduli presenti di AngularJS_G.

Questo strumento fornisce già dei sistemi incorporati per gestire anche tutta la parte

relativa ai test dell'applicazione che verranno spiegati alla sezione 3.2.4 come Karma, Jasmine e PhantomJS.

3 Processi di Supporto

3.1 Processo di documentazione

3.1.1 Attività

3.1.1.1 Documentazione Con il processo di documentazione, il gruppo *MashUp* intende riportare tutte le informazioni acquisite durante il ciclo di vita del software. In particolare si andranno ad identificare gli standard per la creazione e stesura dei documenti, nonché tutti i processi atti a rendere i documenti formalmente corretti.

3.1.2 Procedure

3.1.2.1 Gestione dei documenti Per ogni documento che il *Responsabile di Progetto* ritiene utile essere steso dovrà essere generato con appositi tools messi a disposizione. Essi permettono di mantenere un certo ordine e rigore logico.

3.1.2.1.1 Creazione di un nuovo documento Ogni qualvolta il *Responsabile di Progetto* voglia creare un nuovo documento dovrà eseguire i seguenti passi:

1. posizionarsi all'interno della cartella *documenti* presente nel repository_G;
2. invocare il comando *make nome_del_documento*.

Al termine dell'esecuzione dello script sarà presente una nuova cartella già rinominata correttamente con all'interno presenti tutti i file utili per l'inizio della stesura del nuovo documento.

3.1.2.1.2 Avanzamento di un documento Alla fine di ogni avanzamento il documento dovrà essere sottoposto a molteplici procedure. Essi possono essere così definite:

1. l'assegnatario dovrà effettuare l'upload del documento all'interno del branch predisposto presente nel repository_G;
2. l'assegnatario segnerà come completato il ticket_G assegnatogli;
3. il *Verificatore* riceverà automaticamente una mail con la conferma dell'avvenuta conclusione;
4. il *Verificatore* assegnerà un ticket_G per la risoluzione de problema al redattore del documento;
5. qualora il documento sia corretto si procederà alla chiusura del ticket_G, avviando automaticamente il *Responsabile di Progetto*, che a sua volta provvederà a contrassegnare il documento come approvato. Altrimenti il *Verificatore* assegnerà nuovi ticket al redattore. In seguito il redattore provvederà a eseguire di nuovo tutti i passi del seguente elenco;

3.1.2.2 Gestione del glossario Ogni membro, a seconda di chi il *Responsabile di Progetto* riterrà opportuno, potrà andare a redigere il glossario.

3.1.2.2.1 Inserimento di un termine In un apposito documento creato su Google Drive, ogni componente che durante la stesura di un documento andrà ad imbattersi in parole che possono risultare ambigue al lettore dovrà inserire quel termine nel documento.

3.1.2.2.2 Eliminazione di un termine L'incaricato alla stesura del *Glossario v3.0.0* andrà a prendere i termini presenti nei documenti in Google Drive e ad inserirli in ordine alfabetico. Una volta inserito il vocabolo provvederà alla sua rimozione da Google Drive.

3.1.2.2.3 Inserimento termine nei documenti Una volta terminata la stesura del *Glossario v3.0.0* l'*Amministratore di Progetto* provvederà ad eseguire gli script necessari all'inserimento dei termini ambigui seguendo la seguente procedura da terminale:

1. dalla root del repository `G` entrare nella cartella `script` ed invocare il comando:

make glossterm

2. ritornare alla root ed entrare nella cartella `documenti`. Invocare da li il comando:

make gloss

Una volta terminata l'esecuzione dell'ultimo script i documenti finali saranno stati generati e l'*Amministratore di Progetto* avrà terminato il suo compito.

3.1.3 Norme

3.1.3.1 Progettazione e sviluppo dei documenti Per ogni documento che si redige si dovrà rispettare un ben preciso layout di sviluppo. Si raccolgono di seguito le informazioni utili per la composizione delle prime pagine, le sezioni e le norme tipografiche da utilizzare in fase di scrittura.

3.1.3.2 Versionamento Per ogni documento sviluppato deve essere presente la versione di avanzamento prevista. La classificazione di versionamento deve rispettare le seguenti norme:

X.Y.Z

dove:

- **X**: numero intero che identifica la versione di rilascio ogni qual volta si prepari il documento per una nuova revisione. Ogni incremento di una unità provoca l'azzeramento delle variabili Y e Z;
- **Y**: numero intero che identifica la versione dopo aver passato la fase di verifica. Ogni incremento di una unità provoca l'azzeramento della variabile Z;
- **Z**: numero intero che identifica la versione modificata del documento da parte del redattore. Ogni modifica effettuata provoca l'incremento di una unità.

3.1.3.3 Template Al fine di ottenere dei documenti formattati correttamente si è predisposto un template_G in L^AT_EX da seguire scrupolosamente. Esso deve essere utilizzato come linea guida per lo sviluppo dei documenti. Ogni tipologia di documento, prima pagina, indici, contenuti interni, lettere hanno uno stile definito in maniera univoca. Esso ne permette una gestione capillare ed omogenea.

Verrà definita inoltre un intestazione per ogni file *.tex* nel modo seguente:

```
% =====
% File: nome_del_documento.tex
% Description: Definisce il capitolo relativo a ...
% Created: aaaa-mm-gg
% Author: Cognome Nome
% Email: cognome.nome@mashup-unipd.it
% =====
% Modification History:
% Version Modifier Date Change Author
% 0.0.1 aaaa-mm-gg Descrizione del cambiamento Cognome Nome
```

3.1.3.4 Struttura dei documenti Per facilitare la stesura e la gestione si è provvisto alla creazione di un cartella per ogni documento, essa ha come titolo il nome del documento.

Nella root della cartella è presente il file che gestisce il contenuto della prima pagina. Esso ha il compito di includere tutti gli stili e a linkare tutte le sezioni utili alla costruzione del documento.

All'interno della cartella *content* sono presenti tutti i file, divisi per capitolo, contenenti il testo dell'intero documento.

Le cartelle *diagrams* e *images* hanno il compito di contenere i relativi diagrammi ed immagini usati nei singoli documenti.

Nella cartella *doc_to_modify* sono presenti due file, il primo *content_files.tex* raggruppa tutti i sotto capitoli e li ordina in base alle scelte dell'utente. Il secondo file *history.tex* contiene la storia della creazione e modifica apportata al documento.

3.1.3.4.1 Prima pagina La prima pagina contiene un elenco delle caratteristiche fondamentali del documento quali:

- logo del progetto;
- logo, nome ed email del gruppo;
- nome del documento;
- versione del documento;
- data redazione del documento;
- cognome e nome dei redattori del documento;
- cognome e nome dei verificatori del documento;
- cognome e nome dell'approvatore del documento;
- lista di distribuzione del documento;

- uso interno o esterno del documento;
- sommario contenente una breve descrizione del documento.

3.1.3.4.2 Diario delle revisioni Successivamente nella seconda pagina è presente l'elenco delle revisioni apportate al documento. Esso risulta utile per il tracciamento in fase di elaborazione del documento. Permette di conoscere l'autore delle rispettive sezioni. L'elenco è formato da:

- descrizione della modifica apportata al documento;
- cognome, nome e ruolo dell'autore della modifica;
- data della modifica;
- versione del documento dopo la modifica.

3.1.3.4.3 Indici Subito dopo il *Diario delle revisioni* è presente l'indice generale del documento. Esso è reperibile in tutti i documenti ad esclusione del *Glossario*. Nella parte iniziale l'indice fa riferimento ai capitoli e sotto capitoli. Se presenti immagini e/o tabelle, in automatico è possibile trovare anche il relativo indice associato.

3.1.3.4.4 Formattazione di una pagina Tutti i documenti descrittivi sono formati da una intestazione e da un piè di pagina ben definito.

Sono presente due bande di colore grigio chiaro agli estremi del documento.

Nella parte alta del documento a sinistra si trova il logo ed il nome del gruppo, mentre nella parte destra il titolo del capitolo principale.

Nella parte inferiore del documento partendo da sinistra è possibile trovare il nome del documento con la relativa versione.

Nella parte destra è presente il numero di pagina formato da "Pagina X di Y", dove con "X" si indica la pagina corrente e con "Y" il numero totale di pagine presenti ad esclusione della prima pagina, del diario delle revisioni e degli indici.

3.1.3.5 Suddivisione sezioni documenti

3.1.3.5.1 Studio di Fattibilità Il documento raccoglie le motivazioni e le considerazioni che il gruppo ha elaborato per l'accettazione o meno del progetto da sviluppare. All'interno del documento può essere presente un'analisi che ha portato all'esclusione di altri capitolati.

Il documento è di tipo interno.

3.1.3.5.2 Norme di Progetto Il documento raccoglie tutte le attività, norme, procedure e strumenti che il gruppo adotterà durante lo sviluppo del progetto se esso verrà concesso.

Il documento è di tipo interno.

3.1.3.5.3 Piano di Progetto Il documento specifica la pianificazione impiegata per lo svolgimento del progetto. Sono contenute tutte le attività con le relative tempistiche e i relativi rischi. Sono inoltre presenti le specifiche di ogni risorsa impiegata;

Il documento è di tipo esterno.

3.1.3.5.4 Piano di Qualifica Il documento specifica le strategie applicate al capitolato per ottenere gli obiettivi di qualità. Sono inoltre presenti le attività di verifica e pianificazione con i relativi test che si andranno a sviluppare.

Il documento è di tipo esterno.

3.1.3.5.5 Analisi dei Requisiti Il documento identifica e descrive i requisiti necessari all'implementazione del progetto. Saranno inoltre integrati i casi d'uso ed i requisiti utili allo sviluppo del progetto. Saranno altresì presenti i diagrammi grafici di interazione tra utenti e sistema.

Il documento è di tipo esterno.

3.1.3.5.6 Specifica Tecnica Il documento identifica una prima progettazione ad altro livello del sistema che si andrà a sviluppare. Saranno individuati e dettagliatamente specificati i design pattern utilizzati nel progetto.

Il documento è di tipo esterno.

3.1.3.5.7 Definizione di Prodotto Il documento identifica dettagliatamente il prodotto che si va a sviluppare. Tale documento sarà utilizzato dai programmati per poter sviluppare il software.

Il documento è di tipo esterno.

3.1.3.5.8 Glossario Il documento raccoglie tutti i termini usati nella documentazione e ne riporta una definizione più descrittiva. Il documento è di tipo esterno.

3.1.3.6 Norme tipografiche Tutti i documenti devono rispettare le seguenti norme tipografiche, utili per mantenere i documenti omogenei tra loro.

3.1.3.6.1 Stili di testo Molti stili di testo sono già stati definiti ed inclusi in appositi file. L^AT_EX permette così un forte controllo sulla tipografia, restringendone i campi d'applicazione solamente quando necessario.

- **Colori:** il testo colorato è previsto solamente nei link. Essi sono di colore blu;
- **Corsivo:** il testo corsivo deve essere utilizzato ogni volta si voglia dare enfasi ad una determinata parola/frase;
- **Grassetto:** il testo grassetto deve essere utilizzato nella stesura della prefazione degli elenchi non ordinati e ai soli titoli dei capitoli. Può essere utilizzato nel testo ma con moderazione solamente quanto si intende focalizzare un argomento;
- **Maiuscolo:** l'uso del testo maiuscolo è riservato solamente agli acronimi;
- **Sottolineato:** non è previsto l'uso del testo sottolineato.

3.1.3.6.2 Punteggiatura L'utilizzo della punteggiatura deve essere coerente per tutti i testi scritti in L^AT_EX. Si impone così:

- **Maiuscole:** il primo carattere di testo maiuscolo è consentito solamente dopo il punto, punto di domanda, punto esclamativo o nell'intestazione di ogni elenco;

- **Numeri:** i numeri dovranno rispettare lo standard SI/ISO_G 31-0
- **Parentesi:** nel testo è ammesso solamente l'uso di parentesi tonde. Il testo racchiuso tra parentesi non deve mai iniziare o finire con alcun carattere di spaziatura;
- **Punteggiatura:** prima di ogni simbolo di punteggiatura *non deve* essere presente alcun carattere di spaziatura. L'uso del punto servirà solamente per terminare una discussione;
- **Spazi:** non dovranno mai essere presenti spazi doppi, o spazi per la tabulazione.

3.1.3.6.3 Composizione del testo

- **Elenchi non numerati:** ogni singolo elemento deve iniziare con un testo scritto in grassetto e con la prima lettera maiuscola. Ogni singolo punto dell'elenco deve terminare con il punto e virgola. Solamente alla fine è obbligo l'uso del punto;
- **Elenchi numerati:** ogni singolo elemento deve iniziare con il testo scritto in minuscolo. Ogni singolo punto dell'elenco deve terminare con il punto e virgola. Solamente alla fine è obbligo l'uso del punto;
- **Note a piè di pagina:** ogni singola nota a piè di pagina deve iniziare con lettera maiuscola. Tutte le note a piè di pagina devono terminare obbligatoriamente con il punto e virgola;
- **Pedice “G”:** per tutti gli acronimi e/o termini particolari presenti nei documenti e riportati nel *Glossario*, deve essere presente la notazione tramite pedice “G”.

3.1.3.6.4 Formati ricorrenti

- **Nomi propri:** tutti nomi propri devono essere espressi nella forma “Cognome Nome”;
- **Date:** ogni data deve essere scritta come indicato dallo standard ISO_G 8601:

AAAA – MM – GG

dove:

- **AAAA:** rappresenta l'anno scritto in numero ed in modalità estesa con tutti e quattro i caratteri;
- **MM:** rappresenta il mese scritto in numero ed in modalità estesa con tutti e due i caratteri;
- **GG:** rappresenta il giorno scritto in numero ed in modalità estesa con tutti e due i caratteri.

- **Riferimenti ai documenti:** ogni riferimento ai documenti deve presentare il nome del documento e la sua attuale revisione;

3.1.3.7 Componenti grafiche

3.1.3.7.1 Immagini Ogni immagine inclusa nei documenti deve essere in formato PDF, meglio se in formato vettoriale in quanto riduce lo spazio occupato e migliora la scalabilità.

Ogni immagine deve essere salvata nell'apposita directory *images* messa a disposizione.

Assieme all'immagine deve essere fornita una descrizione ed una parola chiave per poter riportare nel testo la figura di riferimento.

Il numero viene assegnato automaticamente dal software L^AT_EX.

3.1.3.7.2 Diagrammi Ogni diagramma riportato nei documenti deve essere in formato PDF, meglio se in formato vettoriale in quanto riduce lo spazio occupato e migliora la scalabilità.

Ogni diagramma deve essere salvato nell'apposita directory *diagrams* messa a disposizione.

Assieme al diagramma deve essere fornita una descrizione ed una parola chiave per poter riportare nel testo il diagramma di riferimento.

Il numero viene assegnato automaticamente dal software L^AT_EX.

3.1.4 Strumenti

Per semplificare ed automatizzare il lavoro sono stati creati alcuni script che verranno lanciati o automaticamente durante una determinata azione sul repository_G o in maniera manuale dai membri del team quando necessario.

3.1.4.1 Script

- **Generazione struttura documento:** il makefile predisposto per la nuova generazione dei documenti ha il ruolo di effettuare una copia del template_G di documento ed assegnargli il nome;
- **Compilazione documento:** per la compilazione dei documenti è stato sviluppato un makefile che procede alla compilazione completa del file L^AT_EX e ne verifica tramite il software GNU Aspell la correttezza ortografica.
Alla fine della compilazione elimina tutti i file temporanei e procede all'apertura del file pdf;
- **Inserimento notazioni termini di glossario:** lo script provvede alla duplicazione della cartella relativa ai documenti e analizza documento per documento, inserendo la notazione "G" in pedice per tutti i termini presenti nel glossario.
Alla fine si crea automaticamente una nuova cartella contenente tutti i documenti compilati;
- **Calcolo indice di Gulpease:** tramite lo script di parsing automatico avviato sul documento è possibile ottenere l'indice di Gulpease all'interno di un file di testo separato;
- **Controllo ortografico:** sul makefile per la compilazione dei file L^AT_EX è presente l'integrazione del software GNU Aspell per il controllo ortografico che blocca la compilazione e lancia una eccezione specificando il termine da modificare.

3.2 Processo di verifica

Con il processo di verifica si vuole introdurre una approfondita analisi del rispetto dei requisiti fissati prima dell'inizio del progetto.

Tale tecnica permette di aumentare il fattore efficienza/efficacia e di ridurre il tempo impiegato nell'analisi.

Il processo prevede una prima fase di pianificazione e successivamente di verifica.

3.2.1 Attività

3.2.1.1 Analisi statica L'attività di analisi statica è una ben precisa tecnica di verifica, applicabile sia a documenti sia a codice sorgente. Essa viene applicata durante tutto il ciclo di sviluppo del progetto. Ha come obiettivo la ricerca delle anomalie. Un volta individuate possono essere così applicate:

- **Walkthrough:** il seguente metodo, ricerca all'interno del testo e del codice, tutte le possibili anomalie. L'analisi si basa sulla lettura di tutto il testo.

Dopo aver trovato le varie anomalie si portano in discussione con il redattore del documento e si cerca una risoluzione ottimale.

Tale tecnica può essere usata all'inizio del progetto, in quanto non si ha una visione generale del documento/codice che si sta sviluppando. Permette così ai verificatori di preparare una *checklist*.

Come descritto inizialmente, tale tecnica risulta la più adeguata. Solamente dopo aver stilato una *lista di controllo*, si potrà passare ad una analisi più specifica, usando l'attività di analisi di tipo *inspection*;

- **Inspection:** il seguente metodo si basa sulla lettura mirata del documento/-codice per analizzarne i difetti. Sarà così possibile usare la lista di controllo per verificare la presenza di errori. Con l'acquisizione di nuove capacità si potrà così ampliare la lista di controllo in modo tale da rendere l'operazione più efficiente/efficace.

3.2.1.2 Analisi dinamica L'attività di analisi dinamica si può applicare solamente a livello software. Può essere applicata all'interno software o per semplicità anche ad una singola porzione di esso.

L'attività prevede l'esecuzioni di test automatici programmati precedentemente, essi servono a verificarne il corretto funzionamento. Essendo uno strumento automatico di verifica, in caso di anomalie il sistema invierà delle notifiche per identificare chiaramente il problema.

Tutti i test applicati si intendono ripetibili, ovvero devono poter essere applicati in qualsiasi momento e ripetuti per tutta la durata del software.

Ogni test deve avere delle caratteristiche ben precise da rispettare, ovvero:

- **Ambiente:** deve essere riportato necessariamente l'ambiente sia software sia hardware in cui il sistema dovrà eseguire il test. Deve necessariamente essere specificato lo stato iniziale del sistema;
- **Specifiche:** devono essere riportati necessariamente i dati in input ed output. Questi serviranno per poter effettuare un test di congruenza;
- **Procedure:** possono essere specificati ulteriori istruzioni per l'esecuzione dei test. Inoltre possono essere riportate istruzioni sulla corretta lettura dei risultati.

3.2.1.3 Gestione anomalie Il *Verificatore*, durante le attività di controllo, nel caso si riscontrassero anomalie o discordanze normative avrà il compito di notificarle all’assegnatario del task_G sul quale sta effettuando la verifica.

Nella sezione 3.2.2.1. è indicata la procedura per l’apertura di una nuova anomalia.

3.2.1.4 Test

3.2.1.4.1 Test di unità I test di unità verificano che ogni unità software funzioni a dovere. Attraverso questo si può verificare la correttezza di tutti i moduli base che andranno a comporre il software, limitando così gli errori di implementazione. Questo tipo di test vengono identificati con la sintassi seguente:

TU[Codice test]

3.2.1.4.2 Test di integrazione I test di integrazione verificano che i moduli verificati singolarmente funzionino anche se integrati tra di loro. Questo aiuta anche a scoprire eventuali problemi non sottolineati nei test precedenti. L’integrazione inoltre non viene fatta solo con moduli interni ma anche con componenti esterne quali librerie e framework_G.

Potranno inoltre essere utilizzate delle componenti create ad hoc che simulano i risultati dei moduli non ancora pronti, questo per poter analizzare il risultato di determinati test. Questo tipo di test vengono identificati con la sintassi seguente:

TI[Identificativo del componente]

3.2.1.4.3 Test di sistema I test di sistema consistono nella validazione_G del prodotti software. Questi saranno eseguiti solamente quando si ritiene che il prodotto è giunto ad una versione stabile. Si verifica dunque che il prodotto soddisfi tutti i requisiti. Questo tipo di test vengono identificati con la sintassi seguente:

TS[Tipo requisito][Codice requisito]

3.2.1.4.4 Test di regressione I test di regressione vengono eseguiti dopo che un componente è stato modificato. Si eseguono tutti i test fatti precedentemente per verificare che le modifiche fatte non causino problemi agli altri moduli. Grazie al tracciamento si possono ricavare dei test critici, ossia coloro che sono più a rischio in caso di modifica, e quindi da ripetere ogni volta.

3.2.1.4.5 Test di validazione I test di validazione_G coincidono con il collaudo del prodotto con il proponente_G. Se tale test è positivo allora significa che il prodotto proposto è abbastanza maturo per essere rilasciato. Questo tipo di test vengono identificati con la seguente sintassi:

TV[Tipo requisito][Codice requisito]

3.2.1.5 Tracciamento L'attività di tracciamento, eseguita dai *Verificatori*, prevede il controllo della corrispondenza dei requisiti definiti nelle fasi precedenti. Inoltre ogni requisito deve essere tracciato al componente che lo soddisfa, in modo da garantire che ogni requisito venga soddisfatto. Questo consente anche di misurare il progresso nell'attività di progettazione.

3.2.2 Procedure

3.2.2.1 Procedure d'assegnazione delle anomalie L'assegnazione delle anomalie riscontrate da parte del *Verificatore* deve avvenire tramite l'uso della piattaforma Asana_G.

Esistono due tipi di sezioni sulle quali effettuare la segnalazione:

- **Bug_G Tracking Document:** sezione dedicata alle anomalie riguardante la documentazione;
- **Bug_G Tracking Source:** sezione dedicata alle anomalie riguardante il codice sorgente del software.

È necessario fare uso di uno dei due template_G messi a disposizione, presenti all'interno delle viste sopraelencate.

Per l'assegnazione si devono eseguire i seguenti punti, maggiormente schematizzati nella figura 5:

1. definizione del nuovo titolo, contenente nella parte iniziale la dicitura [bug_G];
2. descrizione esplicativa dell'errore riscontrato e punto del riscontro;
3. assegnazione al membro del gruppo che ha commesso l'errore;
4. inserimento data di scadenza per la correzione dell'anomalia;
5. assegnazione del tipo di priorità in base alle norme presenti nella sezione 3.2.3.1.

Il sistema automaticamente provvederà alla notifica del nuovo task_G attraverso l'invio di una mail.

3.2.2.2 Configurazione sistema di Continuous Integration Ogni membro del gruppo dovrà seguire la seguente procedura per interfacciarsi con il sistema adottato di Continuous Integration 3.2.4.4 configurato dall'*Amministratore di Progetto*.

- accedere al sito <http://wercker.com/> tramite l'account di GitHub;
- dal menù in alto selezionare la voce “Create” e scegliere “Application”;
- selezionare GitHub_G come provider di Git_G;
- selezionare il repository_G che interessa (src_BDSM_App_client e src_BDSM_App_server);
- premere il pulsante “Join App”;
- ripetere la procedura per entrambi i repository_G citati al punto 4.

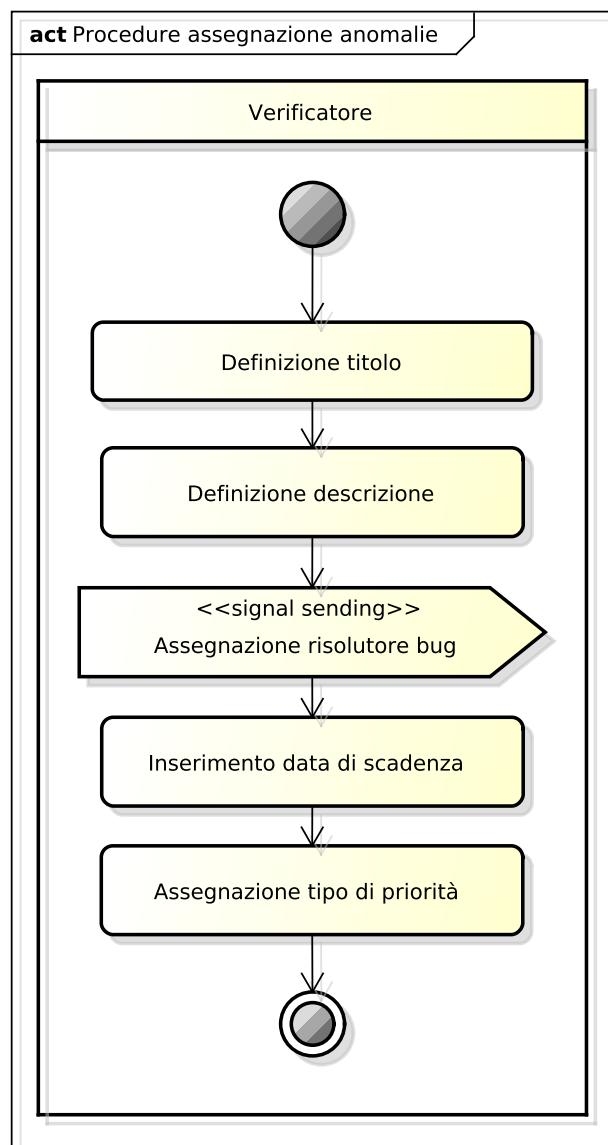


Figura 4: Diagramma di attività - procedura assegnazione anomalie

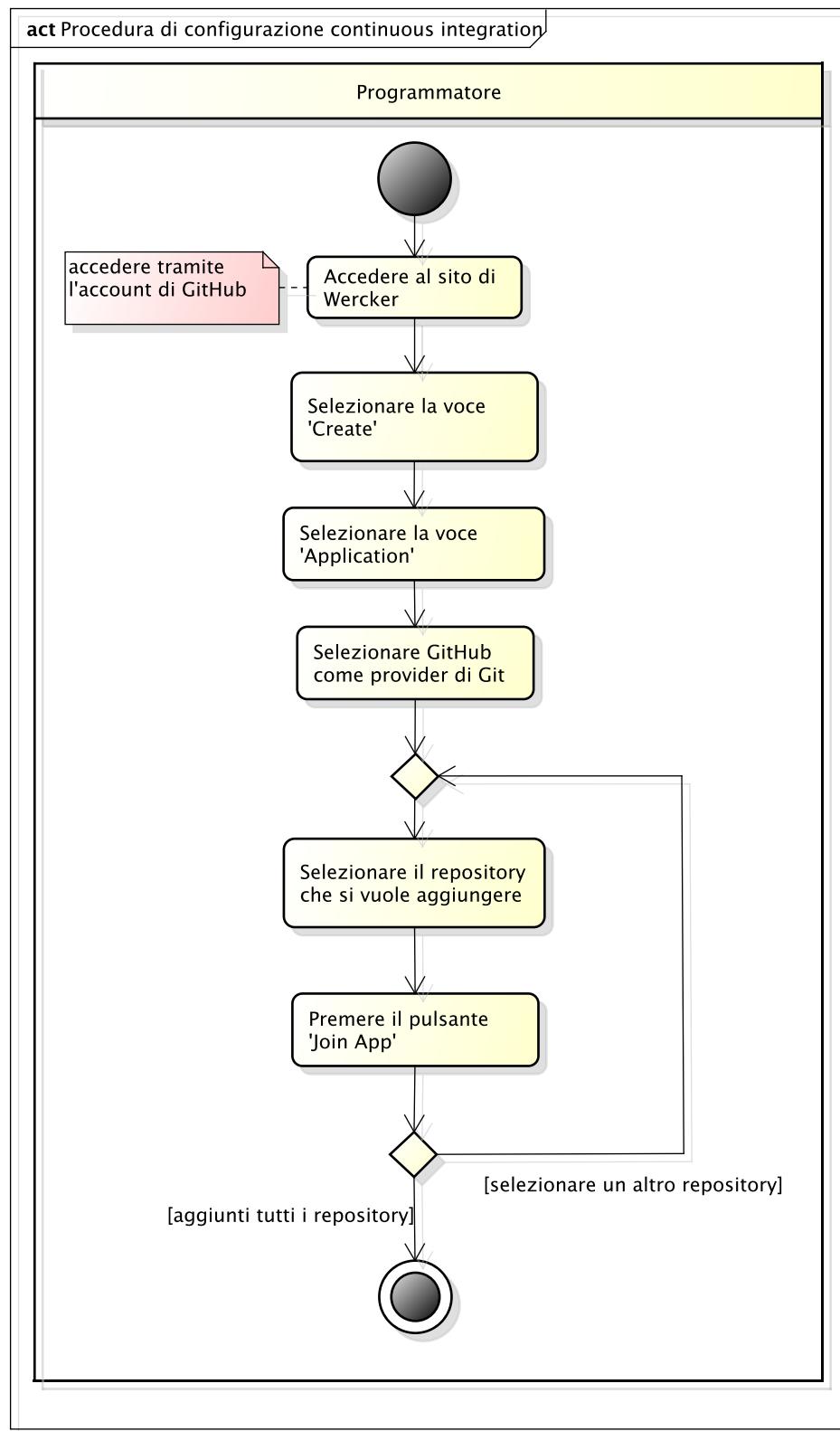


Figura 5: Diagramma di attività - procedura configurazione continuous integration

3.2.3 Norme

3.2.3.1 Priorità risoluzione anomalie Ogni anomalia riportata nel sistema deve avere una priorità di esecuzione. Di seguito vengono definite le possibili scelte:

- **P0**: priorità elevata, la risoluzione deve avvenire entro le 24 ore dalla notifica;
- **P1**: priorità media, la risoluzione deve avvenire entro 3 giorni dalla notifica;
- **P2**: priorità normale, la risoluzione deve avvenire entro 7 giorni dalla notifica;
- **P3**: priorità bassa, la risoluzione deve avvenire entro 2 settimane dalla notifica;

3.2.4 Strumenti

3.2.4.1 Correzione ortografica Per la parte di correzione ortografica, si sono scelti di usare due strumenti di analisi. Il primo risiede all'interno dell'editor Tex-Maker, offrendo le funzionalità di correttore ortografico per i file L^AT_EX. Esso ha dei limiti, in quanto usa un normale dizionario presente all'interno del sistema operativo e può essere utile per gli errori più grossolani di ortografia.

Per ovviare al problema, si è scelto di implementare ulteriormente un controllo più accurato tramite il software *GNU Aspell*. Esso permette una migliore integrazione con i file L^AT_EX e permette l'aggiunta centralizzata dei termini di più difficile comprensione.

3.2.4.2 Calcolo indice Gulpease Attraverso i file di testo, generati in automatico per ogni documento dallo script elencato nella sezione 3.1.4.1, l'*Amministratore di Progetto* potrà confrontarne la conformità con le metriche previste dal *Piano di Qualifica v4.0.0*.

3.2.4.3 Cloc Cloc è lo strumento che viene utilizzato per il calcolo di alcune metriche stabile del *Piano di Qualifica v4.0.0* come il conteggio delle linee di commento e quelle di codice effettivo.

3.2.4.4 Wercker Wercker è lo strumento scelto per le attività di Continuous Integration. È disponibile come applicazione cloud, non necessitando quindi di essere scaricato in locale. Questo consente di non dover provvedere a possibili diverse configurazioni nelle macchine dei diversi membri del gruppo. Sarà compito dell'*Amministratore di Progetto* impostare inizialmente i settaggi necessari a far funzionare l'applicativo.

3.2.4.5 Karma Karma è uno strumento che permette di eseguire del codice JavaScript_G in più browser o di testarlo in diverse maniere e in diversi momenti.

Viene configurato direttamente quando si installa Yeoman attraverso il generatore di angular utilizzato.

3.2.4.6 Jasmine Jasmine è un framework_G JavaScript_G di sviluppo basato sul comportamento che viene utilizzato per testare l'applicazione, in particolare per programmare i “test di unità”. Fornisce delle funzioni che aiutano a strutturare i test e che permettono di fare delle asserzioni.

Viene configurato direttamente quando si installa Yeoman attraverso il generatore di angular utilizzato.

3.2.4.7 Protractor Protractor è un framework JavaScript di test “end-to-end (e2e)” per le applicazioni in AngularJS che permette di simulare le interazioni dell’utente. Utilizza Jasmine 3.2.4.6 per la sintassi dei test. Viene consigliato l’uso di questo strumento al posto del deprecato **Angular Scenario Runner**.

Non è presente di default negli strumenti impostati da Yeoman. Verrà quindi configurato dall’*Amministratore di Progetto* prima che gli altri componenti vadano a codificare questo tipo di test.

3.2.4.8 Python unittest `unittest` è un modulo integrato in Python che fornisce un framework \mathbf{G} per effettuare i test di unità di tutti i moduli Python presenti nell’applicativo. Tra le diverse cose, `unittest` fornisce una classe base su cui definire diversi test e metodi per avviare il testing dell’applicativo.

3.2.4.9 GAE testbed `testbed` è un modulo Python fornito da Google che permette di testare in locale le componenti che saranno ospitati in Google App Engine \mathbf{G} fornendo dei servizi `stub \mathbf{G}` , dei metodi che simulano i vari servizi offerti da Google Cloud Platform. In quanto tale modulo necessita che l’SDK \mathbf{G} di Google App Engine sia presente nel path di sistema, sarà compito dell’*Amministratore di Progetto* configurarne l’utilizzo in modo tale che rimanga uniforme per tutte le macchine che lo utilizzeranno.

4 Processi Organizzativi

4.1 Gestione dei processi

4.1.1 Attività

4.1.1.1 Gestione delle Comunicazioni

4.1.1.1.1 Mail Ogni membro del gruppo avrà una mail personale creata grazie all'acquisizione di un dominio web sul servizio di hosting NetSons.

Il formato dell'indirizzo dovrà essere del tipo esposto di seguito e servirà per registrarsi ad ogni servizio che il team andrà ad utilizzare:

cognome.nome@mashup-unipd.it

4.1.1.1.2 Comunicazioni interne Le comunicazione interne, prettamente informali riguardanti la logistica, verranno gestite tramite un gruppo su WhatsApp denominato MashUp.

Quelle formali interne verranno gestite attraverso l'applicativo Slack che consente delle chat di gruppo. Stiamo cercando di sostituire l'utilizzo di WhatsApp con quest'ultimo per qualsiasi tipo di comunicazione, allo scopo di convogliare il tutto in un unico servizio. Ogni componente dovrà registrarsi tramite dominio <https://mashup-unipd.slack.com> per accedere alle comunicazioni del team.

Le norme e le procedure relative a questo servizio verranno trattate in dettaglio nella sezione 4.1.1.5 e in quella 4.1.4.1.

Se si necessita di un interazione vocale con gli altri membri, qualora non fossero presenti nello stesso luogo, verrà utilizzata l'applicazione Skype.

4.1.1.1.3 Comunicazioni esterne Le comunicazione esterne vengono effettuate dal *Responsabile di Progetto* in quanto rappresenta il gruppo *MashUp*.

Egli, attraverso la seguente mail, manterrà i contatti con il proponente_G e con il committente_G. In caso lo ritenga necessario, girerà tali messaggi agli altri membri del team.

info@mashup-unipd.it

È stato inoltre stabilito, insieme al proponente, che l'interazione con lui, qualora non potesse avvenire tramite un incontro esterno, specificato nella sezione 4.1.1.2.2, possa avvenire tramite videochiamata in Skype e condividendo parte dei documenti del progetto.

4.1.1.2 Gestione delle Riunioni

4.1.1.2.1 Riunioni interne Il *Responsabile di Progetto* ha il compito di convocare le riunioni interne al team. Dovrà quindi informare i componenti tramite le metodologie viste nella sezione 4.1.1.1.2.

Per ogni nuovo incontro dovranno essere specificati la data, l'ora, il luogo, il proponente_G e la motivazione che lo hanno reso necessario.

Ad ogni membro del gruppo è consentito chiedere la convocazione di una riunione

interna. Il *Responsabile di Progetto*, una volta valutati i motivi e la necessità di tale incontro, provvederà ad organizzarlo secondo le norme viste in precedenza.

4.1.1.2.2 Riunioni esterne Il *Responsabile di Progetto* ha il compito di concordare la data, l'ora e il luogo dell'incontro con il proponente o con il committente attraverso il meccanismo visto nella sezione 4.1.1.3.

Una volta trovato l'accordo dovrà notificarlo agli altri membri secondo i metodi presenti nella sezione 4.1.1.2.

Ad ogni membro del gruppo è consentito chiedere la convocazione di una riunione esterna. Il *Responsabile di Progetto*, oltre ad accertarsi dei motivi e delle necessità di tale incontro, dovrà garantire che siano presenti almeno due componenti del team. Sarà compito di uno dei presenti, delegato di volta in volta, redigere il verbale dell'incontro avvenuto.

4.1.1.3 Gestione del sistema di ticketing Il sistema scelto per gestire i task_G da assegnare ai membri del team è Asana_G.

In esso si possono creare diverse viste per diversificare i task_G a seconda dell'ambito per il quale vengono aperti. Le viste presenti sono:

- **Bug_G Tracking Document:** vista dedicata ai task_G necessari alla risoluzione di problemi legati ai documenti;
- **Bug_G Tracking Source:** vista dedicata ai task_G necessari alla risoluzione di problemi legati al codice sorgente;
- **Change Request Document:** vista dedicata ai task_G aperti per proporre dei cambiamenti/miglioramenti al *Responsabile di Progetto* legati ai documenti;
- **Change Request Source:** vista dedicata ai task_G aperti per proporre dei cambiamenti/miglioramenti al *Responsabile di Progetto* legati al codice sorgente;
- **Fasi di avanzamento:** vista dedicata ai task_G che servono per soddisfare le milestone_G fissate nel *Piano di Progetto v5.0.0* per portare a compimento il progetto scelto;
- **Presentazioni:** vista dedicata ai task_G che servono a portare a compimento le presentazioni che il gruppo *MashUp* deve affrontare in corrispondenza di ogni revisione di progetto decisa dal committente_G;
- **Revisioni:** vista dedicata ai task_G che servono per soddisfare le revisioni d'avanzamento decise dal committente_G allo scopo di verificare lo sviluppo del progetto;
- **Stesura Documenti:** vista dedicata ai task_G che servono nell'avanzamento della redazione dei documenti di progetto.

Ogni task_G può appartenere a più viste. Sarà compito del *Responsabile di Progetto* valutare se i task che apre possano essere presenti in diverse parti per facilitarne il controllo. I task invece che andranno ad aprire gli altri membri saranno collocati in un'unica vista come esposto nelle procedure.

4.1.1.4 Gestione delle Milestone Il *Responsabile di Progetto* deve pianificare i punti di controllo che il team deve raggiungere e che richiedono il soddisfacimento di alcune attività per poterli superare con successo.

Una milestone_G in Asana_G è semplicemente un task_G particolare generato inserendo il carattere ":" alla fine del titolo. Questo creerà in automatico una rappresentazione grafica diversa rispetto ai normali task.

Possono essere inserite solo nelle seguenti viste:

- **Fasi di avanzamento**
- **Presentazioni**
- **Revisioni**

4.1.1.5 Gestione dei task Il *Responsabile di Progetto* deve pianificare e assegnare i compiti che i singoli membri dovranno andare a eseguire. Per farlo viene usato l'applicazione web Asana_G che consente di creare facilmente i task_G.

Permette inoltre un buon monitoraggio dell'andamento globale del progetto, offrendo grafici che illustrano il rapporto tra i ticket_G completati e quelli ancora da terminare in relazione anche alle milestone_G fissate dal team.

4.1.1.6 Gestione dello svolgimento dei task Ogni membro del team è tenuto ad accettare tutti i task_G che il *Responsabile di Progetto* pianificherà di assegnarli. Se per dei motivi eccezionali l'assegnatario non potesse svolgere il compito indicato dovrà renderlo noto nei modi previsti nella sezione 4.1.1.1.2 al *Responsabile di Progetto*, il quale, una volta valutata le motivazioni, dovrà provvedere a trovare un nuovo destinatario del task.

Qualora il *Responsabile di Progetto* non ricevesse alcuna notifica entro le prime 24 ore riterrà automaticamente accettato il task_G dall'assegnatario.

4.1.1.7 Gestione delle change request Ogni membro del team può proporre dei cambiamenti o degli avanzamenti da apportare. Il suggerimento dovrà essere motivato adeguatamente in modo che il *Responsabile di Progetto* non debba preoccuparsi di ricercare le fonti che forniscono l'effettiva validità di questo mutamento. Una volta verificata o meno la necessità dell'idea, il *Responsabile di Progetto* approverà o respingerà la proposta fatta.

Qualora la proposta fosse accettata il *Responsabile di Progetto* avrà il compito di pianificare le attività che serviranno ad apportare il cambiamento.

4.1.1.8 Gestione della commit L'operazione di commit_G deve essere il più atomica possibile comprendendo il minore numero di file per volta. Oltre a dare una migliore visione di ciò che si sta facendo e di dove si sta lavorando garantisce anche la probabilità di avere meno conflitti all'interno del repository_G.

Per fornire al *Responsabile di Progetto* un più ampio quadro di come stanno procedendo i singoli compiti viene utilizzato un servizio web offerto da GitHub_G.

Questo permette di mostrare, nei commenti dei task_G di Asana_G, il messaggio inserito durante la commit_G una volta effettuata l'operazione di push sul server.

Fornisce quindi il tracciamento del lavoro che viene svolto per portare a compimento il task_G in questione. Il servizio viene attivato sia in doc_BDSM_App sia in src_BDSM_App.

4.1.1.8.1 Meccanismo di aggancio commit-task È stato creato un account generale su Asana_G con user name “MainMashUp” che fornisce l’API_G_Key necessaria a configurare il servizio senza che esso dipenda dall’account di un membro in particolare.

Per permettere il tracciamento bisogna usare l’identificato del task_G di cui vogliamo tracciare i messaggi di commit_G.

L’id si può reperire nel modalità illustrate dalla figura 7 e va inserito all’interno del messaggio nelle modalità indicate nella sezione 4.2.3.1.8.

4.1.1.9 Gestione della rotazione dei ruoli Lo sviluppo di un progetto necessita della collaborazione di diverse persone che andranno a ricoprire incarichi che rappresentano delle figure aziendali. Queste figure sono presenti nella sezione 4.1.3.1. Viene garantito che ogni componente del gruppo *MashUp* dovrà ricoprire, almeno una volta, tutti i ruoli. Questa rotazione potrebbe generare dei problemi dovuti a conflitti d’interesse in quanto lo stesso soggetto che svolge un compito potrebbe ritrovarsi a verificare l’operato svolto in precedenza.

Un’attenta pianificazione da parte del *Responsabile di Progetto* dovrà preoccuparsi di non fare avvenire queste situazioni.

Il compito di garantire che non siano stati fatti errori spetta al *Verificatore*, il quale, una volta riscontrata una incongruenza, dovrà notificarla al *Responsabile di Progetto* che avrà il compito di risolverla.

Il ruoli verranno ruotati in base a quanto stabilito nel *Piano di Progetto v5.0.0* a seconda della fase di avanzamento in cui si trova il team. In base a quelle indicazioni il *Responsabile di Progetto* deciderà a chi fare svolgere i diversi compiti. Ad ogni membro, siccome durante ciascuna fase può ricoprire più ruoli, potranno essere assegnati task_G che riguardano campi diversi. Deve essere però garantito che i ticket_G, nell’arco di 3 giorni, riguardino lo stesso ruolo per garantire una migliore linearità e coerenza nel lavoro. Quest’ultima norma non è vincolante in caso ci sia l’esigenza immediata di svolgere un lavoro che nessun altro membro riesca ad eseguire.

4.1.2 Procedure

4.1.2.1 Procedura di generazione di una milestone Il *Responsabile di Progetto* dovrà eseguire i seguenti passi per generare una milestone e potrà farlo solo entrando nelle viste elencate nella sezione 4.1.1.4:

1. definire il titolo della milestone_G;
2. inserire alla fine del titolo il carattere “:”;
3. definire la data in cui verrà effettuato il controllo del raggiungimento degli obiettivi prefissati;
4. inserire una descrizione generale sulla natura del controllo che verrà apportato e sui macro obiettivi da raggiungere.

I task_G che serviranno per soddisfare gli obiettivi prefissati dovranno essere collocati sotto la milestone_G creata.

4.1.2.2 Procedura d’assegnazione dei task Il *Responsabile di Progetto* dovrà eseguire i seguenti passi, riassunti in seguito nella figura 6.

Per effettuare un’assegnazione si deve servire del template illustrato nella figura 10 disponibile nella vista “Fasi di avanzamento” di Asana.

1. definire il titolo del task secondo le norme presenti nella sezione 4.1.3.2;
2. se necessario, fornire una descrizione più esaustiva del compito da svolgere;
3. definire la data di scadenza entro la quale il task_G dovrà essere portato a termine;
4. indicare il verificatore;
5. aggiungere tra i followers del task_G l’utente: “MainMashUp”;
6. selezionare l’assegnatario del task_G. Questa assegnazione genererà in automatico una notifica al membro scelto;
7. se si dovesse verificare il caso in cui l’assegnatario rifiutasse il task_G assegnato e il *Responsabile di Progetto* accertasse questa impossibilità nel suo svolgimento, dovrà eseguire di nuovo il passo precedente.

4.1.2.3 Procedura per il reperimento dell’id del task Ogni membro che vorrà recuperare l’identificativo del task_G su cui sta lavorando dovrà:

1. entrare su Asana_G;
2. posizionarsi sul task_G che si sta lavorando dalla sezione “MyTasks”;
3. posizionarsi sull’url e copiare tutte le cifre dopo l’ultimo slash “/”. Quello è l’id che si sta cercando.

4.1.2.4 Procedura di svolgimento dei task assegnati Il membro assegnatario del task_G, una volta ricevuta la notifica e non avendo nessuna motivazione per rifiutarlo, dovrà:

1. se il task_G ha una scadenza più immediata rispetto a quella del task sui cui si sta lavorando, sospendere lo svolgimento del task con data di scadenza più lontana e procedere con quello appena notificato, altrimenti metterlo in coda;
2. se dopo avere iniziato l’esecuzione del task_G, si riceve la notifica di uno nuovo con scadenza più immediata, si dovranno sospendere le attività che portano al compimento del task con scadenza meno ravvicinata ed eseguire nuovamente il passo precedente;
3. se per svolgere il compito si dovesse andare oltre la data di scadenza prevista, si deve impostare il tag “DELAY” dal sistema Asana o tramite i modi espressi nella sezione 4.2.3.1.8.

Questa situazione si può verificare se:

- il tempo assegnato dal *Responsabile di Progetto* è inferiore a quanto richiesto per il compimento del task_G;
- il task_G può essere completato solo quando un altro task sia portato a termine. Queste dipendenze devono verificarsi il meno possibile ed è compito del *Responsabile di Progetto* evitare ciò in fase di pianificazione;

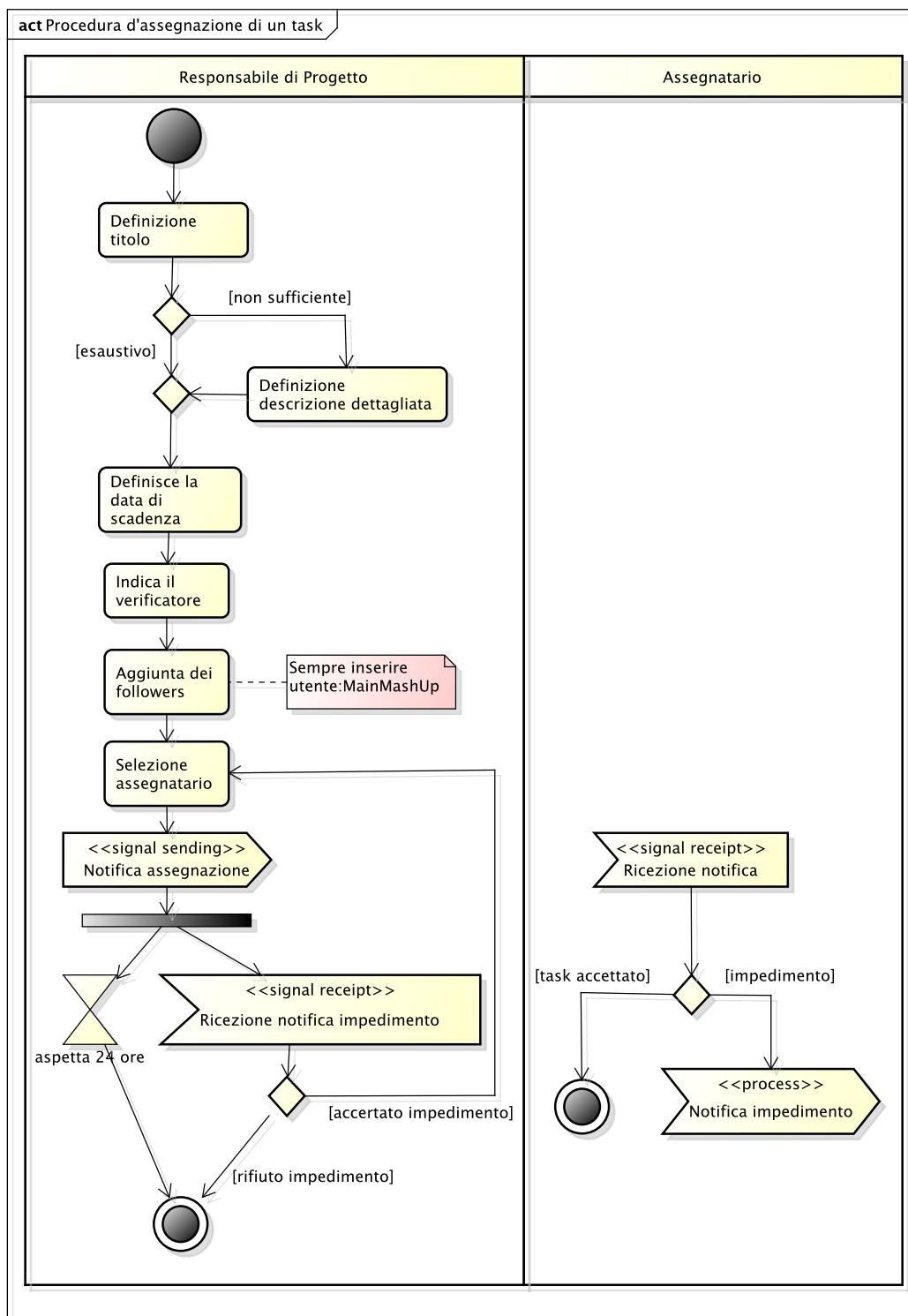


Figura 6: Diagramma di attività - procedura d'assegnazione di un task

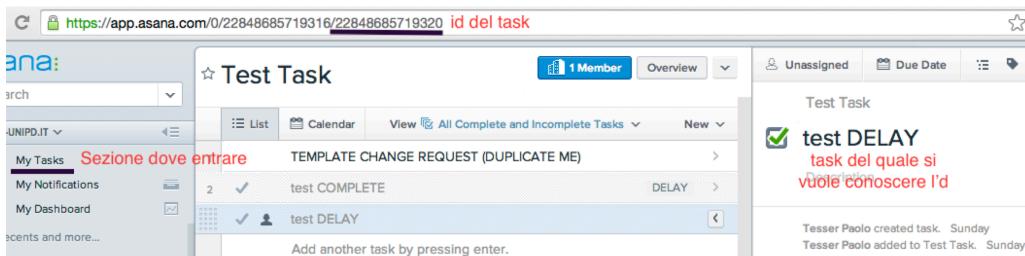


Figura 7: Reperimento dell'id del task

- l'assegnatario ha rallentamenti esterni non resi noti al *Responsabile di Progetto* in fase di pianificazione;
- il compimento del task_G necessita di competenze non in possesso dell'assegnatario.

Nei primi due casi la responsabilità è imputabile al *Responsabile di Progetto*, mentre negli ultimi due la responsabilità è del membro che devo svolgere il task_G;

4. al completamento del task, dovrà chiudere il task, spuntandolo direttamente dal sistema Asana o tramite i modi espressi nella sezione 4.2.3.1.8;
5. l'operazione illustrata al passo precedente genera in automatico una notifica verso il *Verificatore* e il *Responsabile di Progetto*;
6. ritornare al primo punto con i task_G rimanenti.

4.1.2.5 Procedura di svolgimento di una change request Ogni membro del gruppo, che intenda proporre dei cambiamenti o introdurre degli avanzamenti, è tenuto a seguire questa procedura.

Per effettuare l'apertura di una change request si deve servire del template illustrato nella figura 11 disponibile nella vista “Change Request Document” o “Change Request Source” di Asana a seconda del tipo di proposta che si vuole effettuare.

1. definire il titolo del cambiamento/avanzamento;
2. definire una descrizione dettagliata del motivo che ha portato alla proposta e le fonti che garantiscono la bontà del cambiamento/avanzamento che si vuole attuare;

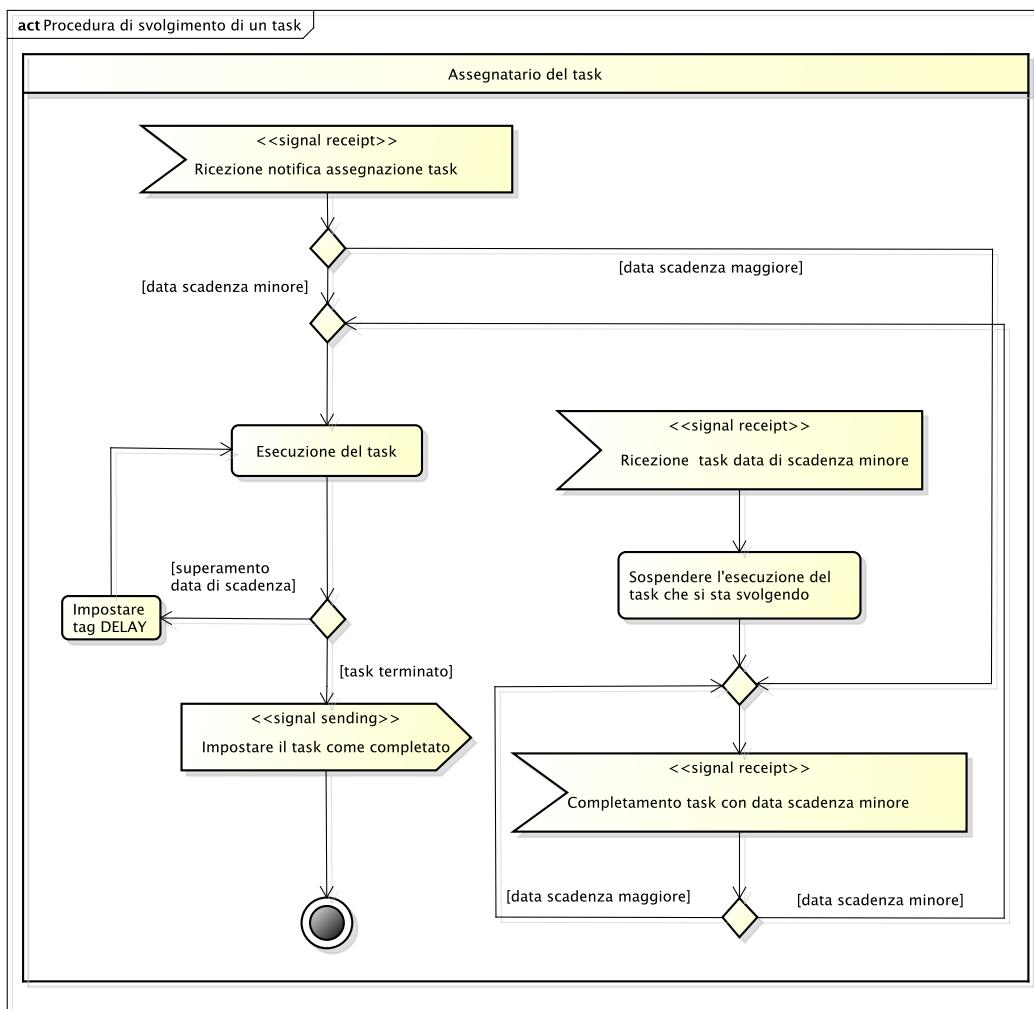


Figura 8: Diagramma di attività - procedura di svolgimento dei task assegnati

3. selezionare il *Responsabile di Progetto* in carica. Questa assegnazione genera automatico una notifica.

Il *Responsabile di Progetto*, ricevuta la notifica, dovrà:

1. verificare le fonti e accertarsi della reale necessità di apportare il cambiamento o l'avanzamento proposto;
2. accettare o rifiutare la proposta;
3. notificare la decisione presa al proponente_G;
4. se la proposta è stata accettata, pianificare le attività che serviranno per apportarla.

4.1.2.6 Procedura di rilevazione dei rischi Il *Responsabile di Progetto* ha il compito di individuare i rischi trovati nel *Piano di Progetto v5.0.0*.

Questa attività è iterativa durante tutto il progetto e necessita di un continuo monitoraggio dei rischi rilevati nel periodo iniziale. Proprio per la sua natura non finita, si può andare incontro a problemi non previsti inizialmente.

Qualora ciò avvenisse il *Responsabile di Progetto* dovrà eseguire la seguente sequenza di attività:

1. registrazione del riscontro effettivo dei rischi nel *Piano di Progetto v5.0.0*;
2. pianificazione per la gestione dei nuovi rischi trovati;
3. aggiornamento delle metodologie per fare fronte alla nuova pianificazione attuata;
4. monitoraggio dei nuovi rischi riscontrati durante lo sviluppo del progetto.

4.1.3 Norme

4.1.3.1 Ruoli di Progetto Vengono presentati i diversi ruoli, delineandone le mansioni e le responsabilità.

4.1.3.1.1 Responsabile di Progetto Il *Responsabile di Progetto* rappresenta il team e il progetto verso il committente_G e il proponente_G.

Ha quindi le seguenti responsabilità e compiti:

- pianificazione e coordinamento delle attività;
- gestione e controllo delle risorse;
- analisi e gestione dei rischi;
- approvazione dell'offerta economica;
- assicurarsi che tutte le attività svolte siano conformi alle *Norme di Progetto v5.0.0* e rispettino la pianificazione effettuata nel *Piano di Progetto v5.0.0*;
- garantire che non ci siano conflitti di interesse. A tal proposito, se il *Responsabile di Progetto* dovesse prendere parte alla stesura di qualche documento, dovrà nominare un *Responsabile di Progetto* delegato che avrà il compito di sostituirlo nell'approvazione di quei documenti;
- redigere il *Piano di Progetto v5.0.0*.

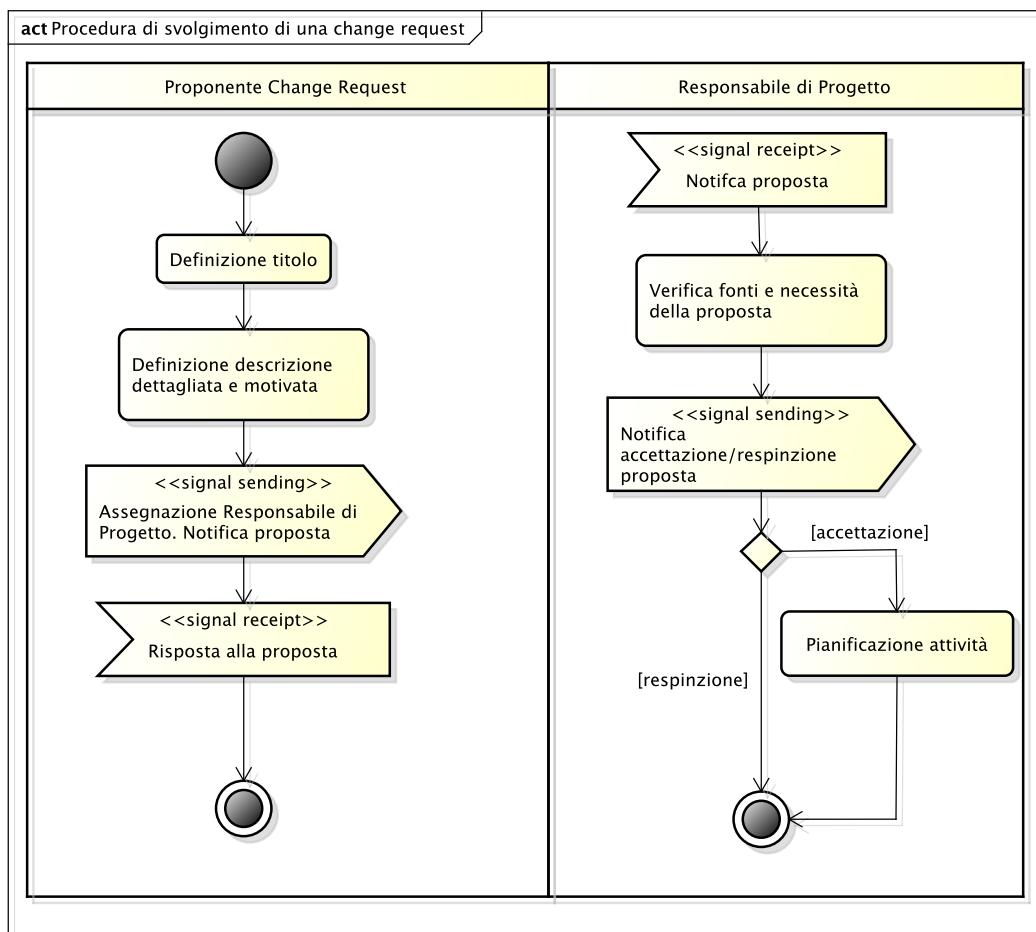


Figura 9: Diagramma di attività - procedura di svolgimento di una change request

4.1.3.1.2 Amministratore L'*Amministratore di Progetto* è il responsabile dell'ambiente di lavoro.

Ha quindi il compito di:

- ricercare e implementare strumenti che automatizzino il maggior numero di operazioni;
- gestire il versionamento del codice e della documentazione di progetto;
- fornire procedure che servono a garantire la qualità del prodotto uscente da un determinato compito;
- redigere le *Norme di Progetto v5.0.0*.

4.1.3.1.3 Analista L'*Analista* è il responsabile dell'attività di analisi.

Ha quindi il compito di:

- comprendere a pieno la natura del problema e la sua complessità;
- ricercare i requisiti che servono per realizzare il prodotto richiesto dal proponente_G;
- redigere lo *Studio di Fattibilità v1.0.0*;
- redigere l'*Analisi dei Requisiti v4.0.0*.

4.1.3.1.4 Progettista Il *Progettista* è il responsabile delle attività di progettazione.

Ha quindi il compito di:

- effettuare scelte progettuali estratte da soluzioni note e fortemente testate (design pattern_G);
- effettuare scelte progettuali che permettano al sistema di essere il più facilmente mantenibile in futuro;
- produrre una soluzione comprensibile, attuabile e motivata;
- redigere il *Piano di Qualifica v4.0.0*.

4.1.3.1.5 Programmatore Il *Programmatore* è il responsabile delle attività di codifica necessarie a portare il prodotto in uno stato che riesca a soddisfare i requisiti richiesti. Deve inoltre programmare i componenti che servono a verificare il sistema.

Ha quindi il compito di:

- scrive codice che rispetti le metriche stabilite per la sua scrittura;
- implementare lo soluzione descritte dal *Progettista*;
- implementare i test sul codice prodotto.

4.1.3.1.6 Verificatore Il *Verificatore* è il responsabile delle attività di verifica.

Ha quindi il compito di:

- controllare la conformità di ogni stadio del ciclo di vita del prodotto;
- garantire che le attività svolte siano conformi alle norme stabilito;
- redige il *Piano di Qualifica v4.0.0* per la parte che illustra l'esito e la completezza delle verifiche effettuate.

4.1.3.2 Formato dei task I task_G che si andranno ad aprire devono contenere le seguenti informazioni:

- **Titolo:** una breve descrizione del compito da eseguire. In esso all'inizio andrà riportato uno dei seguenti valori:
 - [doc]: se il task_G riguarda la stesura/modifica di un documento;
 - [tpl]: se il task_G riguarda la modifica/avanzamento sulla struttura del template_G usato per la documentazione o sulla struttura del documento stesso;
 - [repo]: se il task_G riguarda la modifica/inserimento/cancellazione di qualche elemento di utilità al repository_G;
 - [script]: se il task_G riguarda la modifica/inserimento/cancellazione di qualche script;
 - [img]: se il task_G riguarda la modifica/inserimento/cancellazione di immagini;
 - [dia]: se il task_G riguarda la modifica/inserimento/cancellazione di diagrammi/grafici;
 - [bug_G]: se il task_G riguarda la risoluzione di un bug riscontrato da un *Verificatore*.

In seguito verranno inseriti i tag relativi alle commit_G riguardanti il codice sorgente;

- **Descrizione:** qualora non fosse abbastanza esaustivo il titolo del task_G, si deve fornire una descrizione più dettagliata dell'attività da svolgere;
- **Data di scadenza:** la data entro la quale dovrà essere svolto il lavoro;
- **Assegnatario:** il membro del team che dovrà eseguire il task_G indicato;
- **Verificatore:** il membro del team che andrà ad eseguire la verifica sull'operato svolto dall'assegnatario;
- **Followers:** i membri del team che si vuole rendere partecipi sull'attività svolta. Dovrà essere sempre aggiunto l'utente "MainMashUp".

La figura 10 illustra come vengono rappresentate queste informazioni su Asana. Inoltre l'immagine viene proprio presa da uno screenshot di un template di task, copiabile dal *Responsabile di Progetto* e modificabile una volta copiato a seconda dell'esigenze riportate in precedenza.

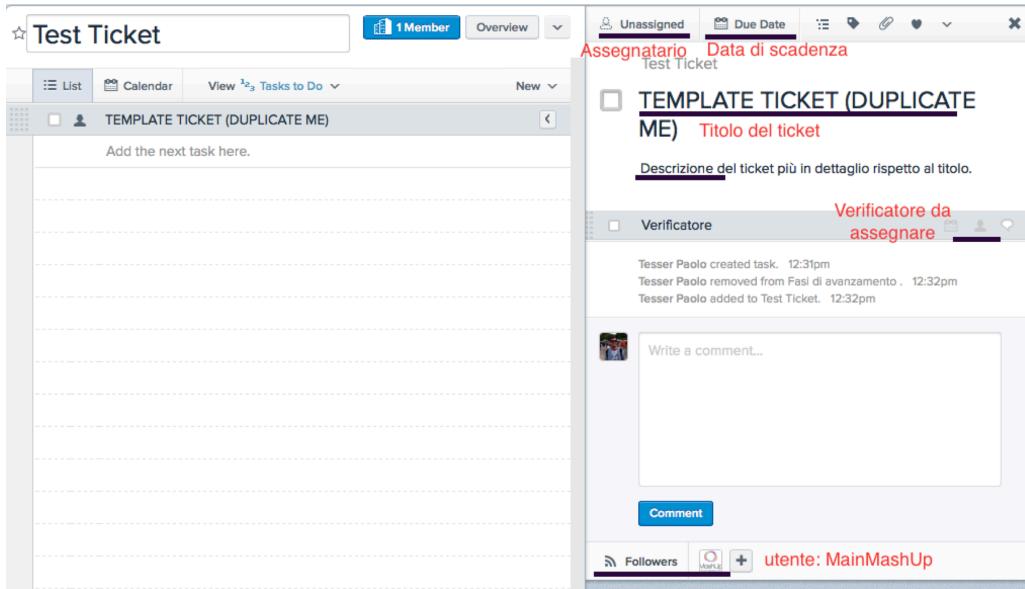


Figura 10: Template Task su Asana

4.1.3.3 Formato delle change request Le change request che si andranno ad aprire devono contenere le seguenti informazioni:

- **Titolo:** una breve descrizione di cosa si vorrebbe cambiare;
- **Descrizione:** una descrizione più dettagliata del cambiamento, motivando opportunamente la necessità riscontrata;
- **Assegnatario:** il *Responsabile di Progetto* al momento in carica secondo la rotazione dei ruoli;
- **Proponente_G:** chi è stato a proporre questo cambiamento. Sarà di fatto chi ha aperto la richiesta.

La figura 11 illustra come vengono rappresentate queste informazioni su Asana. Inoltre l'immagine viene proprio presa da uno screenshot di un template di una change request, copiabile da qualunque membro desideri proporre un cambiamento e modificabile una volta copiato a seconda dell'esigenze riportate in precedenza.

4.1.4 Strumenti

4.1.4.1 Asana Asana_G è l'applicazione web scelta per la gestione dei task_G. Registrandosi con una mail di dominio personale come quella riportata nella sezione 4.1.1.1.1 è possibile usufruire di maggiori servizi che consentono una più facile gestione del team da parte del *Responsabile di Progetto*.

4.1.4.2 Astah Astah_G è l'applicativo scelto per la creazione di grafici UML_G. La versione adottata è quella Professional, resa disponibile con una licenza gratuita per gli studenti dei corsi universitari registrandosi tramite l'indirizzo:

nome.cognome.X@studenti.unipd.it

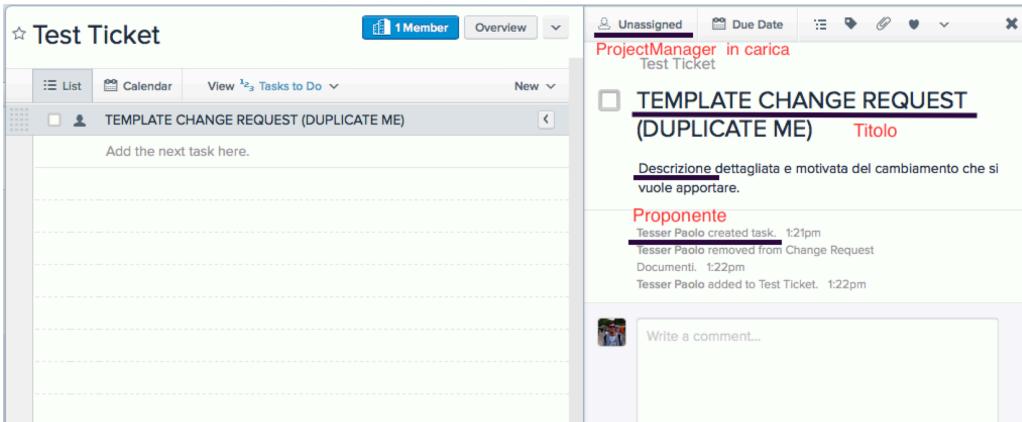


Figura 11: Template Change Request su Asana

4.1.4.3 ProjectLibre ProjectLibre_G è il prodotto scelto per la realizzazione dei diagrammi di Gaant e quelli di PERT_G.

4.1.4.4 NetSons NetSons è il servizio di hosting che il team ha deciso di adottare a scopo principalmente formativo.

Il dominio creato è:

<http://www.mashup-unipd.it>

Il servizio fornisce anche una serie di email personali che il gruppo ha deciso di utilizzare come spiegato nelle sezioni 4.1.1.1.1 e 4.1.1.1.3.

4.1.4.5 Skype Skype è l'applicativo scelto per effettuare videochiamate o chiamate VoIP tra i componenti del gruppo quando c++_G è la necessità di consultarsi o risolvere problemi e non è possibile essere presenti fisicamente nello stesso ambiente.

4.1.4.6 WhatsApp WhatsApp è l'applicativo di messaggistica scelto per le comunicazioni interne e informali al gruppo.

4.1.4.7 Slide Slide è un editor online basato sul framework_G Reveal.js_G che consente di creare e condividere presentazioni.

Il sito dell'applicazione web è: <http://slides.com> e viene utilizzato attraverso la mail del gruppo descritta nella sezione 4.1.1.1.1.

4.1.4.8 Slack Slack è un tool_G che permette la comunicazione di team, è disponibile su tutte le piattaforme sia mobile che desktop, ed andrà a sostituire progressivamente l'utilizzo di WhatsApp. Questo perché consente di avere in un unico ambiente diversi canali riguardanti temi differenti, e di scambiare messaggi con il

singolo componente. Permette inoltre l'integrazione con servizi utilizzati dal gruppo Github ed Asana_G.

4.2 Gestione delle infrastrutture

4.2.1 Attività

4.2.1.1 Gestione del Repository Il gruppo ha deciso di utilizzare tre repository_G che servono a svolgere funzioni diverse, ma necessarie, allo sviluppo del sistema finale.

Il servizio di hosting scelto consente, tramite licenza “educational”, di impostare la visibilità degli ambienti come privata.

Una volta iscritti i membri dovranno comunicare all'*Amministratore di Progetto* il loro nome utente che provvederà ad autorizzarne l'accesso.

- **doc_BDSM_App**: gestione della documentazione;
- **src_BDSM_App_client**: gestione del codice relativo al client;
- **src_BDSM_App_server**: gestione del codice relativo al server;

4.2.1.2 Gestione dei Git Hooks L'*Amministratore di Progetto* ha il compito di mantenere gli script che permettono di tenere il repository_G consistente con le norme descritte in questo documento.

Questo strumento per sua natura ha dei limiti, in quanto deve essere installato localmente in ogni macchina da parte dei singoli membri.

L'*Amministratore di Progetto* provvederà a notificare ai membri del gruppo quando applicare la procedura descritta nella sezione 4.2.2.1 tramite i sistemi visti nella sezione 4.1.1.1.2.

La notifica potrà avvenire principalmente in due casi:

- ogni volta che un membro clona il repository_G in locale nella sua macchina;
- ogni volta che l'*Amministratore di Progetto* effettua un aggiornamento degli script.

4.2.1.3 Gestione del template del messaggio di commit L'*Amministratore di Progetto* ha il compito di mantenere il meno ambiguo possibile l'ambiente di lavoro. È per questo che si è deciso di utilizzare un template_G standard per andare a scrivere il messaggio della commit_G. Questo strumento per sua natura ha dei limiti, in quanto deve essere installato localmente in ogni macchina da parte dei singoli membri.

L'*Amministratore di Progetto* provvederà a notificare ai membri del gruppo quando applicare la procedura descritta nella sezione 4.2.2.2 tramite i sistemi visti nella sezione 4.1.1.1.2.

La notifica potrà avvenire principalmente in due casi:

- la prima volta che un membro entra a fare parte del team di lavoro;
- ogni volta che l'*Amministratore di Progetto* effettua un aggiornamento degli script.

4.2.2 Procedure

4.2.2.1 Installazione e manutenzione Git Hooks I membri del team dovranno eseguire questa procedura una volta ricevuta la notifica da parte dell'*Amministratore di Progetto*.

Il diagramma presente nella figura 12 illustra in maniera più schematizzata i compiti da eseguire.

1. entrare dalla root del repository_G tramite terminale nella cartella “script”;
2. digitare sempre da terminale il comando:

```
make hook
```

4.2.2.2 Installazione e manutenzione messaggio di commit I membri del team dovranno eseguire questa procedura una volta ricevuta la notifica da parte dell'*Amministratore di Progetto*.

Il diagramma presente nella figura 13 illustra in maniera più schematizzata i compiti da eseguire.

1. scaricare il nuovo template_G allegato alla notifica ricevuta che avrà nome “gitmessage.txt”;
2. se è la prima volta che si esegue la procedura, lanciare il seguente comando dal terminale, altrimenti passare al passo successivo:

```
gitG config --global core.editor "mate"
```

Sostituite “mate” con l’editor che preferite;

3. entrare nella directory principale dell’utente (esempio: /Users/nameofuser);
4. copiare il file allegato in questa cartella rinominandolo in modo da renderlo nascosto inserendo un punto davanti al nome (esempio: .gitmessage.txt).

4.2.3 Norme

4.2.3.1 Repository

4.2.3.1.1 Nomi dei file in doc_BDSM_App I file e le cartelle presenti nel repository_G devono essere conformi al seguente formalismo tratto dallo Standard ISO_G 9660:1999 (Level 2):

- i caratteri usati sono solo quelli minuscoli a-z, 0-9, l’underscore (_) e il punto (.) (esempio: nome_del_documento.tex);
- non sono ammessi caratteri accentati;
- i nomi non possono includere spazi o finire con un punto (.);
- i nomi non devono contenere più di un punto (.) ad eccezione di quelli che fanno riferimento ad una specifica versione (esempio: studio_di_fattibilita_v1.0.0.pdf);
- i nomi non devono essere più lunghi di 21 caratteri esclusi i 3 destinati all’estensione.

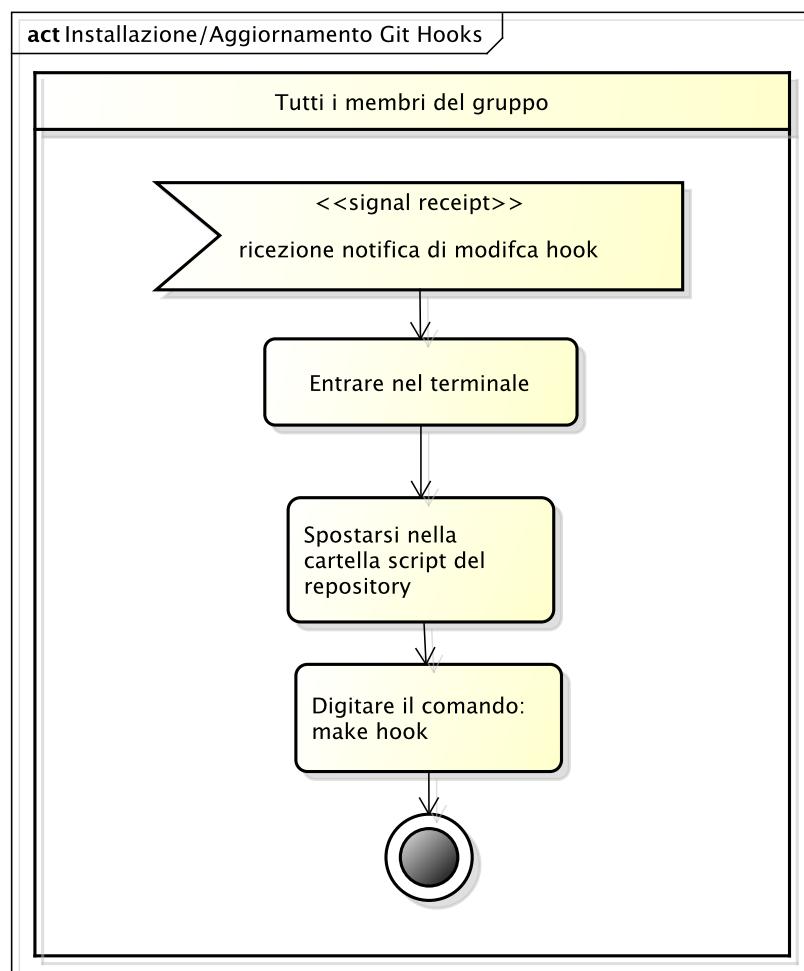


Figura 12: Diagramma di attività - installazione/aggiornamento dei Git Hooks

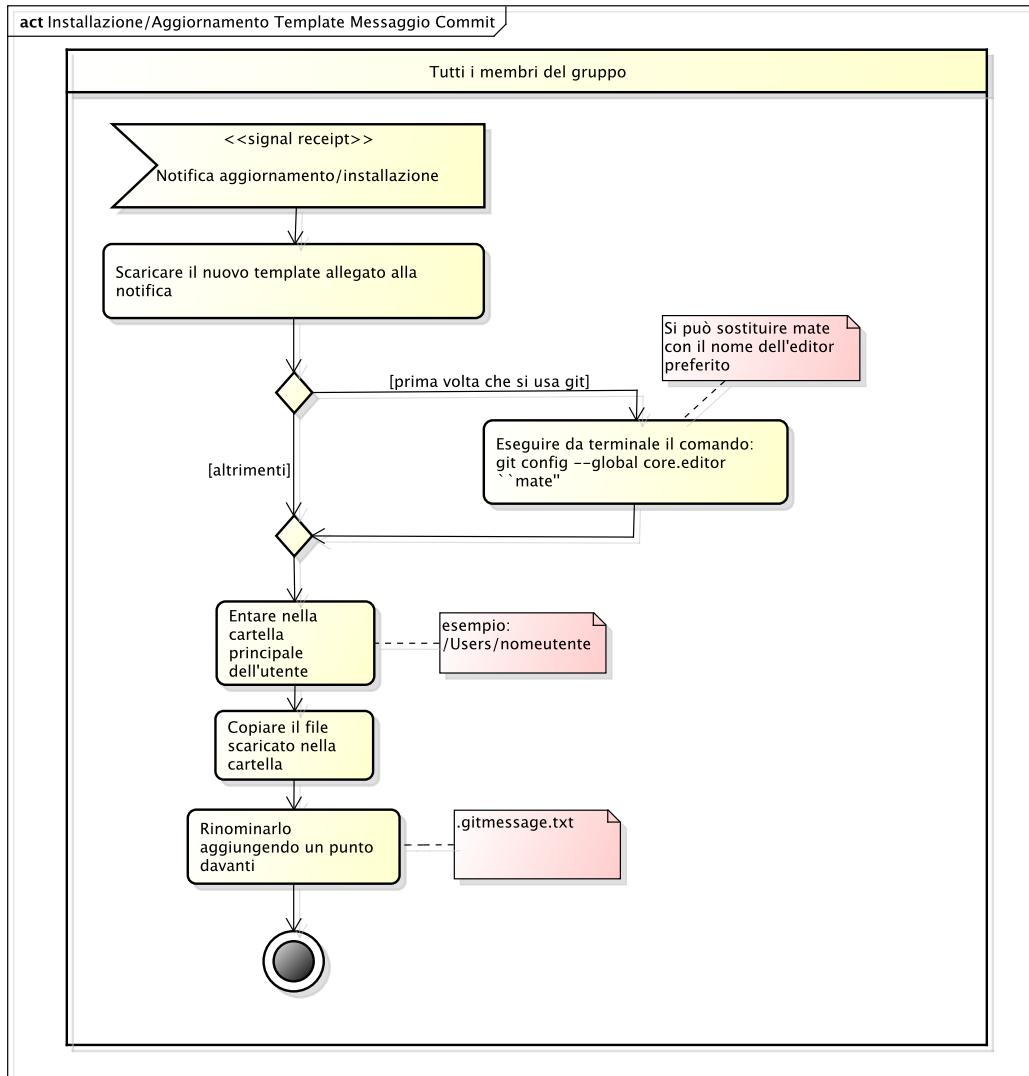


Figura 13: Diagramma di attività - installazione/aggiornamento del template del messaggio per la commit

4.2.3.1.2 Struttura di doc_BDSM_App Le cartelle nel repository_G verranno organizzate nel seguente modo a partire dalla root:

- **consegne:** contenente i documenti finali con anche la versione da consegnare alle diverse revisioni. Sono presenti le cartelle:
 - **revisione_dei_requisiti:** contenente i documenti necessari alla revisione dei requisiti;
 - **revisione_di_progettazione:** contenente i documenti necessari alla revisione di progettazione;
 - **revisione_di_qualifica:** contenente i documenti necessari alla revisione di qualifica;
 - **revisione_di_accettazione:** contenente i documenti necessari alla revisione di accettazione.

All'interno di ciascuna di esse ci sarà un'ulteriore suddivisione in due categorie:

- **interni:** contenente i documenti necessari al gruppo per la sua organizzazione;
- **esterni:** contenente i documenti necessari alla pianificazione e all'avanzamento dello sviluppo.

- **documenti:**

- **template_G:** contenente i file che servono per gestire in maniera univoca la redazione di un documento;
- **template_G_document:** contenente i file di esempio che dovranno essere utilizzati per ogni documento reale;
- una cartella per ogni documento che avrà come nome quello del documento in questione (esempio: norme_di_progetto).

- **script:** contenente tutti gli script necessari ad automatizzare il lavoro e il controllo della documentazione;
- **presentazioni:** contenente le presentazioni create per le diverse revisioni esporate dall'applicativo web utilizzato per esse. Viene fatto ciò per sicurezza qualora il sito non fosse raggiungibile a causa di diversi problemi. All'interno di essa c++_G è un ulteriore suddivisione rispetto alle singole revisioni nello stesso modo in cui è suddivisa la cartella "consegne".

4.2.3.1.3 Nomi dei file in src_BDSM_App_client I file e le cartelle presenti nel repository devono essere conformi al formalismo definito al link <https://github.com/johnpapa/angular-styleguide#naming>.

4.2.3.1.4 Struttura di src_BDSM_App_client Le cartelle nel repository_G verranno organizzate nel seguente modo a partire dalla root:

- **app:** serve per gestire il codice sorgente in produzione del client;
- **test:** serve per gestire i test di integrazione e di sistema relativi al client;
- **script_repo:** serve per gestire gli script che automatizzano parte del lavoro e dei controlli sul repository_G relativi al codice sorgente del client.

4.2.3.1.5 Nomi dei file in src_BDSM_App_server I file e le cartelle presenti nel repository devono essere conformi al formalismo presente al link <https://google-styleguide.googlecode.com/svn/trunk/pyguide.html#Naming>.

4.2.3.1.6 Struttura di src_BDSM_App_server Le cartelle nel repository_G verranno organizzate nel seguente modo a partire dalla root:

- **lib**: serve per contenere le librerie esterne che andranno utilizzate dall'applicazione;
- **src**: serve per gestire il codice sorgente in produzione del server;
- **test**: serve per gestire i test di unità, integrazione e di sistema relativi al server;
- **script_repo**: serve per gestire gli script che automatizzano parte del lavoro e dei controlli sul repository_G relativi al codice sorgente del server.

4.2.3.1.7 Modello di sviluppo Per lo sviluppo della documentazione e del codice necessari al progetto si è scelto di adottare il modello proposto dal proponente_G, spiegato nel dettaglio al seguente link:

<http://nvie.com/posts/a-successful-git-branching-model/>

Ogni membro del gruppo dovrà leggere l'articolo e applicarlo secondo le norme di denominazione dei branch presenti in esso.

4.2.3.1.8 Template messaggio di commit Il messaggio di commit_G dovrà essere conforme alla seguente notazione:

```
Title: [type]
Desc:
Task\gloss{}: Reported in #id_task
Option: [option] [option]
END
```

- **Title**: inserire una breve descrizione come titolo di quello che è stato fatto (massimo 43 caratteri). All'interno delle parentesi quadre inoltre andrà riportato il corrispondente valore presente nel titolo del ticket_G:

- [doc]: se la commit_G riguarda la stesura di un documento;
- [tpl]: se la commit_G riguarda la modifica/avanzamento sulla struttura del template_G usato per la documentazione o sulla struttura del documento stesso;
- [repo]: se la commit_G riguarda la modifica/inserimento/cancellazione di qualche elemento di utilità al repository_G;
- [script]: se la commit_G riguarda la modifica/inserimento/cancellazione di qualche script;
- [img]: se la commit_G riguarda la modifica/inserimento/cancellazione di immagini;
- [dia]: se la commit_G riguarda la modifica/inserimento/cancellazione di diagrammi/grafici;

- [bug_G]: se la commit_G riguarda la risoluzione di un bug riscontrato da un *Verificatore*.

In seguito verranno inseriti i tag relativi alle commit_G riguardanti il codice sorgente;

- **Desc:** inserire una descrizione più esaustiva dell'attività svolta qualora non fosse sufficiente quella data nel titolo;
- **Task:** al posto della dicitura “id_task” inserire l'id del task effettivo reperibile su Asana secondo le modalità esposte nella sezione 4.1.2.3;
- **Option:** Le opzioni applicabili sono:
 - **complete:** per chiudere il task_G qualora fosse l'ultimo necessario allo svolgimento del compito indicato;
 - **delay:** per aggiungere il tag DELAY al task_G se si è in ritardo con lo svolgimento del compito assegnato.

4.2.4 Strumenti

4.2.4.1 Git Git_G è il sistema di controllo di versione utilizzato per entrambi i repository_G del team.

4.2.4.2 GitHub GitHub_G è il servizio web di hosting adottato per tenere una copia del repository_G del progetto.

4.2.4.3 Git Hooks I Git_G Hooks sono degli script personalizzabili che vengono eseguiti in corrispondenza di un determinato evento avvenuto nel repository_G. Vengono utilizzati per controllare il rispetto delle norme in maniera automatizzata non permettendo a tutto ciò che non è conforme di entrare nel sistema. Sono anche impiegati per automatizzare la gestione dei task da parte dei componenti del gruppo secondo le norme presenti nella sezione 4.2.3.1.8.
Risiedono nella seguente cartella a partire dalla root principale del repository_G:

.git_G/hooks

4.2.4.4 Google Drive Google Drive è lo strumento che si è scelto di utilizzare per gestire file che non necessitano di essere sottoposti a controllo di versione. In particolare verrà impiegato per condividere manuali di utilità alla formazione dei membri del gruppo o per la stesura di idee veloci che andranno poi riviste e documentate ufficialmente nell'apposito repository_G.

4.2.4.5 Sistema Operativo I membri del gruppo operano su due sistemi operativi diversi.
Una parte utilizza sistemi basati su Ubuntu e l'altra utilizza Mac OS X con versione 10.10 Yosemite.

4.3 Formazione dei membri del team

I membri del gruppo, per soddisfare le richieste assegnate dal *Responsabile di Progetto* al quale non sanno fare fronte con le conoscenze attuali in loro possesso, dovranno documentarsi adeguatamente durante ore esterne a quelle di lavoro, non imputabili perciò come costi al proponente_G.

I membri possono trovare materiale utile a questo scopo nel luogo dove risiedono i file che non necessitano di versionamento come specificato nella sezione 4.2.4.4.

4.3.1 Attività

4.3.1.1 Incontri con il proponente Sono stati predisposti degli incontri in accordo con il proponente_G per la formazione dei membri del team su alcune delle tecnologie che si dovranno adottare per lo sviluppo del progetto. In particolare sarà illustrato l'utilizzo della Google Cloud Platform.

Le giornate fissate per lo svolgimento sono per il momento due, in data:

- 2015-02-18
- 2015-03-04

Entrambi avranno luogo nella sede di Zing S.r.l. presso:

Tenuta Ca' Tron, Via Sile, 41 - 31056 Roncade, Treviso

A Lista di controllo

Di seguito viene presentata la lista di controllo con gli errori più comuni da controllare durante una inspection:

- **Errori ortografici:**

- negli elenchi non numerati non è rispettato l'uso dei due punti nel primo termine in grassetto;
- negli elenchi non numerati la punteggiatura alla fine non è corretta;
- nell'uso dei comandi rapidi non viene inserita la parentesi graffa alla fine per generare la spaziatura.

- **Italiano:**

- i caratteri accentati all'inizio della frase vengono erroneamente sostituiti con l'apostrofo;
- proponente_G e committente_G: viene confuso il loro significato.

- **LATEX:**

- non vengono usati i comandi rapidi definiti nel file *commands.tex*;
- non vengono usate inserite le caption nelle figure e nelle tabelle.

- **UML_G:**

- non viene utilizzato il carattere corsivo per identificare le classi astratte;
- non viene utilizzata la notazione di UML_G 2.0 per la rappresentazione delle interfacce;
- non si utilizza la notazione UML_G 2.0 per la rappresentazione delle associazioni tra classi;
- i nomi delle variabili e dei metodi non rispettano le norme definite.

- **JavaScript_G:**

- la maggior parte della sintassi riguardante il codice viene controllata tramite l'utilizzo dello strumento JSHint che semplifica di molto le attività di verifica statica;

- **Python:**

- il nome delle variabili non viene scritto correttamente come specificato in C.2.

B Configurazione File AngularJS

Di seguito vengono presentati i template che ogni *Programmatore* dovrà utilizzare quando andrà ad effettuare la codifica del lato client. Andranno inseriti seguendo la procedura descritta in 2.2.2.3. Il linguaggio utilizzato per il sistema di templating è quello offerto dagli applicativi JetBrains e cioè Apache Velocity.

B.1 View

Le View_G sono template_G HTML che verranno richiamati opportunamente dal sistema di routing e di state offerto dal modulo **ui.router** disponibile in AngularJS_G. Per rendere validabili queste pagine, secondo le regole imposte dal W3C, ogni direttiva dovrà essere preceduta dal prefisso **data-**.

Esempi:

- data-ng-app
- data-ng-controller
- data-ng-repeat

B.2 Page Controller

```
(function(){
  'use strict';

  /**
   * Name: nome.js
   * Author: Cognome Nome
   * Mail. cognome.nome@mashup-unipd.it
   *
   * Modify
   * Version Date      Author          Desc
   * ========
   * 0.0.1   aaaa-mm-dd Cognome Nome  description
   * -----
   *
  */

  /**
   * @ngdoc function
   * @name nameModule.controller:NameCtrl
   * @description
   * # NameCtrl
   * Controller of the clientApp
  */

  function $NameCtrl(){
    var vm = this;
  }
})()
```

```

};

$NameCtrl.$inject = [];

angular
.module('$nameModule')
.controller('$NameCtrl', $NameCtrl);

})();

```

B.3 Route Controller

```

(function(){
  'use strict';

  /**
   * Name: $nameConfigRoute.js
   * Author: Cognome Nome
   * Mail. cognome.nome@mashup-unipd.it
   *
   * Modify
   * Version Date          Author          Desc
   * ======
   * 0.0.1    aaaa-mm-dd  Cognome Nome  description
   * -----
   *
  */

function $NameConfigRoutes(${DS}stateProvider) {

  ${DS}stateProvider
    .state('state',{
      url: '/url',
      views: {
        '':{
          templateUrl: 'url',
          controller: '$NameCtrl'
        }
      }
    })
  };

  angular
  .module('$nameRouteApp', ['ui.router'])
    .config(['${DS}stateProvider', '${DS}urlRouterProvider', $NameConfigRoutes]);

})();

```

B.4 Services Model

```
(function() {
```

```

'use strict';

/**
 * Name: $nameService.js
 * Author: Cognome Nome
 * Mail. cognome.nome@mashup-unipd.it
 *
 * Modify
 * Version Date Author Desc
 * =====
 * 0.0.1 aaaa-mm-dd Cognome Nome description
 *
 */

```

```

function $NameService(){

var factory = {
publicVar: '',
publicFunction: publicFunction
};

return factory;

///////////

```

```

/**
 * TODO
 */
function publicFunction(){
// TODO
}

///////////

```

```

/**
 * TODO
 */
function privateFunction(){
// TODO
}

}

```

```

$NameService.$inject = [];
angular

```

```
.module('$nameModule')
.factory('$NameService', $NameService);

})();
```

C Configurazione File Python

C.1 Configurazione costrutto switch

Python nella versione 2.7.9 non presenta il costrutto switch tipico di numerosi linguaggi di programmazione. Deve essere quindi implementato in maniera differente. Di seguito viene dato un esempio di una soluzione semplice di come il costrutto dovrà essere codificato qualora necessario. Tuttavia il suddetto esempio, anche se funziona, non è una delle soluzioni più eleganti. Spetterà quindi al *Amministratore di Progetto* cercare una migliore implementazione qualora lo ritenesse opportuno.

```
if n == 0:  
    print "You typed zero.\n"  
elif n== 1 or n == 9 or n == 4:  
    print "n is a perfect square\n"  
elif n == 2:  
    print "n is an even number\n"  
elif  n== 3 or n == 5 or n == 7:  
    print "n is a prime number\n"
```

C.2 Nome variabili

Essendo python un linguaggio senza variabili tipizzate, nonostante a runtime effettui un forte controllo dei tipi, può causare qualche disorientamento a comprendere il codice da parte di un *Programmatore* che non abbia lui stesso scritto quel modulo. Per ovviare a questo problema viene imposto che alla fine delle variabili sia indicato il tipo nel seguente modo:

- nome_var_int (intero)
- nome_var_str (stringa)
- nome_var_list (lista)
- nome_var_array (array)
- nome_var_obj (tipo non implementato nativamente in python, ma dagli sviluppatori dell'applicazione)