



# Norme di Progetto

## Informazioni sul documento

---

<b>Nome file:</b>	norme_di_progetto_v4.0.pdf
<b>Versione:</b>	4.0
<b>Data creazione:</b>	2014-12-17
<b>Data ultima modifica:</b>	2015-06-08
<b>Lista di distribuzione:</b>	Membri del Team
<b>Redattori:</b>	Mario Garavello
<b>Approvato da:</b>	Luca Tiozzo Brasiola
<b>Verificatori:</b>	Andrea Boscolo Nata
<b>Stato:</b>	Formale
<b>Uso:</b>	Interno

---

## Storia delle modifiche

Versione	Descrizione intervento	Autore	Ruolo	Data
4.0	Approvato	Luca Tiozzo Brasiola	Responsabile	2015-06-08
3.2	Verifica	Andrea Boscolo Nata	Verificatore	2015-06-08
3.1	Aggiornata <b>Lista di controllo</b>	Mario Garavello	Amministratore	2015-06-08
3.0	Approvato	Andrea Boscolo Nata	Responsabile	2015-05-08
2.4	Verifica	Mario Garavello	Verificatore	2015-05-08
2.3	Riorganizzazione secondo ISO 1220 e piccole modifiche alle norme stilistiche	Andrea Costa	Amministratore	2015-05-07
2.2	Aggiunte norme per i colori all'interno dello stile di testo.	Andrea Costa	Amministratore	2015-04-22
2.1	Aggiornata <b>Lista di controllo</b>	Andrea Costa	Amministratore	2015-04-21
2.0	Approvato	Mario Garavello	Responsabile	2015-03-04
1.5	Correzioni varie e verifica	Jacopo Cavallarin	Verificatore	2015-02-20
1.4	Aggiornato Indice di Gulpease. Inserite nuove funzionalità <b>Zeitrack</b> .	Davide Canal	Amministratore	2015-02-19
1.3	Aggiunta appendice <b>Lista di controllo</b>	Davide Canal	Amministratore	2015-02-18
1.2	Aggiornata sezione <b>Processo di verifica</b>	Davide Canal	Amministratore	2015-02-17
1.1	Riorganizzazione sezioni documento. Aggiunta sezione <b>Ruoli di Progetto</b>	Davide Canal	Amministratore	2015-02-17
1.0	Approvazione	Tiziano Longo	Responsabile	2015-01-22
0.18	Correzioni varie e verifica	Luca Tiozzo Brasiola	Verificatore	2015-01-20
0.17	Aggiornata sezione <b>Processo di verifica</b>	Andrea Boscolo Nata	Amministratore	2015-01-18
0.16	Correzioni errori ortografici, aggiornate parole collegate al glossario, aggiornata sezione <b>Tipi di documento</b>	Andrea Boscolo Nata	Amministratore	2015-01-17
0.15	Stesura <b>Classificazione dei casi d'uso</b>	Andrea Boscolo Nata	Amministratore	2015-01-11
0.14	Stesura <b>Diagrammi UML, Doodle Incontri</b> , corretti alcuni tempi verbali	Andrea Boscolo Nata	Amministratore	2015-01-10
0.13	Stesura <b>Milestone e Task</b>	Andrea Boscolo Nata	Amministratore	2015-01-09
0.12	Stesura <b>Tipi di documento, Processo di verifica, Classificazione dei requisiti, Ciclo di vita di un documento</b>	Andrea Boscolo Nata	Amministratore	2015-01-06

0.11	Completate sezioni <b>Diagrammi di Gantt e Zeittrack</b> . Aggiornata <b>Nomenclatura documenti</b>	Andrea Boscolo	Nata	Amministratore	2015-01-06
0.10	Stesura <b>Comunicazioni</b>	Andrea Boscolo	Nata	Amministratore	2015-01-04
0.03	Prima bozza <b>Comunicazioni</b>	Andrea Boscolo	Nata	Amministratore	2014-12-17
0.02	Stesura bozza <b>Processo di documentazione</b> , definizione sezioni da completare	Andrea Boscolo	Nata	Amministratore	2014-12-15
0.01	Stesura <b>Strumenti</b>	Andrea Boscolo	Nata	Amministratore	2014-12-13
0.01	Scheletro documento	Andrea Boscolo	Nata	Amministratore	2014-12-13

## Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
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
<b>2</b>	<b>Processi primari</b>	<b>2</b>
2.1	Processo di sviluppo . . . . .	2
2.1.1	Attività . . . . .	2
2.1.1.1	Analisi . . . . .	2
2.1.1.1.1	Studio di Fattibilità . . . . .	2
2.1.1.1.2	Analisi dei Requisiti . . . . .	2
2.1.1.2	Progettazione . . . . .	2
2.1.1.2.1	Specifica Tecnica . . . . .	2
2.1.1.2.2	Definizione di Prodotto . . . . .	3
2.1.2	Norme . . . . .	3
2.1.2.1	Classificazione dei requisiti . . . . .	3
2.1.2.2	Classificazione dei casi d'uso . . . . .	4
2.1.2.3	Nomi e norme stilistiche . . . . .	4
2.1.3	Strumenti . . . . .	5
2.1.3.1	Android Studio . . . . .	5
2.1.3.2	Spring Tool Suite . . . . .	5
2.1.3.3	HeidiSQL . . . . .	5
2.1.3.4	Zeitrack . . . . .	5
2.1.3.4.1	Requisiti . . . . .	5
2.1.3.4.2	Casi d'uso . . . . .	6
2.1.3.4.3	Componenti . . . . .	6
2.1.3.4.4	Classi . . . . .	6
2.1.3.4.5	Test . . . . .	6
2.1.3.4.6	Tracciamento . . . . .	6
2.1.3.4.7	Glossario . . . . .	6
<b>3</b>	<b>Processi di supporto</b>	<b>7</b>
3.1	Processo di documentazione . . . . .	7
3.1.1	Procedure . . . . .	7
3.1.1.1	Ciclo di vita di un documento . . . . .	7
3.1.2	Norme . . . . .	7
3.1.2.1	Template . . . . .	7
3.1.2.2	Nomenclatura documenti . . . . .	7
3.1.2.3	Formattazione dei documenti . . . . .	8
3.1.2.4	Norme tipografiche . . . . .	9
3.1.2.4.1	Punteggiatura . . . . .	9
3.1.2.4.2	Elenchi . . . . .	9
3.1.2.4.3	Stili di testo . . . . .	9
3.1.2.4.4	Formati ricorrenti . . . . .	9
3.1.2.5	Componenti . . . . .	10
3.1.2.5.1	Immagini . . . . .	10
3.1.2.5.2	Inserimento tabelle . . . . .	11
3.1.2.5.3	Inserimento link . . . . .	11
3.1.2.6	Tipi di documento . . . . .	11

3.1.2.6.1	Struttura verbali . . . . .	11
3.1.2.7	Versionamento dei documenti . . . . .	12
3.1.3	Strumenti . . . . .	12
3.2	Processo di verifica . . . . .	13
3.2.1	Attività . . . . .	13
3.2.1.1	Analisi . . . . .	13
3.2.1.1.1	Analisi statica . . . . .	13
3.2.1.1.2	Analisi dinamica . . . . .	13
3.2.1.2	Test . . . . .	13
3.2.1.2.1	Test di unità . . . . .	13
3.2.1.2.2	Test di integrazione . . . . .	14
3.2.1.2.3	Test di sistema . . . . .	14
3.2.1.2.4	Test di regressione . . . . .	14
3.2.1.2.5	Test di validazione . . . . .	14
3.2.1.3	Tracciamento . . . . .	14
3.2.2	Procedure . . . . .	14
3.2.2.1	Gestione delle anomalie . . . . .	14
3.2.3	Strumenti . . . . .	15
3.2.3.1	Correzione ortografica degli errori . . . . .	15
3.2.3.2	Calcolo Gulpease . . . . .	15
3.2.3.3	Strumenti per la validazione W3C . . . . .	15
<b>4</b>	<b>Processi organizzativi</b>	<b>16</b>
4.1	Processo di gestione . . . . .	16
4.1.1	Attività . . . . .	16
4.1.1.1	Incontri Interni . . . . .	16
4.1.1.2	Incontri Esterni . . . . .	16
4.1.1.3	Comunicazione Interna . . . . .	16
4.1.1.4	Comunicazione Esterna . . . . .	16
4.1.1.5	Diffusione documenti . . . . .	16
4.1.1.6	Milestone . . . . .	17
4.1.1.7	Task . . . . .	17
4.1.1.8	Repository . . . . .	17
4.1.2	Procedure . . . . .	17
4.1.2.1	Creazione nuova Milestone . . . . .	17
4.1.2.2	Creazione nuova lista Task . . . . .	18
4.1.2.3	Creazione nuovo Task . . . . .	19
4.1.3	Norme . . . . .	20
4.1.3.1	Ruoli di Progetto . . . . .	20
4.1.3.1.1	Responsabile . . . . .	21
4.1.3.1.2	Amministratore . . . . .	21
4.1.3.1.3	Analista . . . . .	21
4.1.3.1.4	Progettista . . . . .	21
4.1.3.1.5	Programmatore . . . . .	22
4.1.3.1.6	Verificatore . . . . .	22
4.1.3.2	Repository . . . . .	22
4.1.3.2.1	Registrazione . . . . .	22
4.1.3.2.2	Struttura . . . . .	22
4.1.3.2.3	File ignorati (.gitignore) . . . . .	23
4.1.4	Strumenti . . . . .	23
4.1.4.1	Google Calendar . . . . .	23
4.1.4.2	Diagrammi di Gantt . . . . .	23

4.1.4.3	Diagrammi UML . . . . .	23
4.1.4.4	Doodle Incontri . . . . .	23
4.1.4.5	TeamWork . . . . .	24
4.1.4.6	Google Drive . . . . .	24
4.1.4.7	Sistema operativo . . . . .	24
4.1.4.8	Prezi . . . . .	24
<b>A</b>	<b>Lista di controllo</b>	<b>25</b>
A.1	Norme Stilistiche . . . . .	25
A.2	L <sup>A</sup> T <sub>E</sub> X . . . . .	25
A.3	Italiano . . . . .	25
A.4	UML . . . . .	25
A.5	Codice Java . . . . .	25
A.5.1	Spring Tool Suite . . . . .	25
A.5.2	Android Studio . . . . .	25

## Elenco delle figure

1	Inserimento di un allegato tramite Google Drive . . . . .	17
2	Aggiunta di di una nuova Milestone . . . . .	18
3	Creazione di una nuova lista Task . . . . .	19
4	Creazione di un nuovo Task . . . . .	20
5	Inserimento di un nuovo sub-task . . . . .	20
6	Esempio per la gestione degli incontri . . . . .	23

## 1 Introduzione

### 1.1 Scopo del documento

Questo documento è stato realizzato per esplicitare le norme adottate da tutti i componenti del gruppo Zeitnot durante l'intero periodo di svolgimento del capitolato sHike, commissionato dall'azienda Si14 S.p.a. In particolare si vanno a definire le seguenti regole:

- Comunicazioni tra membri del gruppo;
- Relazioni interpersonali;
- Strumenti di lavoro;
- Creazione dei documenti.

### 1.2 Scopo del prodotto

Il prodotto denominato “sHike” si propone di fornire agli escursionisti uno strumento per tracciare la propria attività, al fine di stimolarli a migliorarsi e condividere con gli altri utenti i propri risultati e percorsi. Inoltre, il prodotto è dotato di una controparte web, in cui l'utente può salvare e visualizzare i propri percorsi con i relativi risultati e successivamente può condividerli e confrontarli con quelli degli altri utenti.

### 1.3 Glossario

Per evitare ambiguità dovute all'uso di termini tecnici nei documenti, in allegato viene fornito il documento *glossario\_v3.0.pdf*. All'interno di tale documento è possibile trovare tutti i termini marcati da una sottolineatura. I termini sono definiti e descritti in modo chiaro così da evitare incomprensioni.

### 1.4 Riferimenti

#### 1.4.1 Informativi

- **Analisi dei Requisiti:** *analisi\_dei\_requisiti\_v3.0.pdf*;
- **GitHub - Student Developer Pack:** <https://education.github.com/pack>;
- **[ISO 8601]:** [http://en.wikipedia.org/wiki/ISO\\_8601](http://en.wikipedia.org/wiki/ISO_8601);
- **Norme stilistiche Google del codice:** <https://google-styleguide.googlecode.com/svn/trunk/javaguide.html>.



## 2 Processi primari

### 2.1 Processo di sviluppo

#### 2.1.1 Attività

##### 2.1.1.1 Analisi

##### 2.1.1.1.1 Studio di Fattibilità

Il primo documento per l'analisi dei requisiti è lo Studio di Fattibilità, la cui realizzazione spetta agli analisti. È compito del responsabile organizzare riunioni per confrontare le diverse opinioni dei membri del gruppo e discutere su di esse. Gli analisti poi, basandosi su quanto deciso nelle riunioni, scriveranno un'analisi riguardante:

- **Breve descrizione:** si descrive brevemente quanto proposto dal capitolato;
- **Tecnologie richieste:** si valuta quanto sono conosciute le tecnologie richieste valutando le attuali conoscenze del gruppo;
- **Individuazione dei rischi:** si comprendono quali sono i punti critici della realizzazione;
- **Considerazioni sulla fattibilità:** si conclude con il riepilogo delle considerazioni sulla fattibilità del capitolato.

##### 2.1.1.1.2 Analisi dei Requisiti

Successivamente alla stesura dello studio di fattibilità, sarà sempre compito degli analisti scrivere l'analisi dei requisiti. Lo scopo è quello di produrre dei requisiti semplici e chiari, partendo dalle informazioni disponibili al gruppo. Per velocizzare e automatizzare il lavoro degli analisti è stato creato Zeitrack (sezione 2.1.3.4), nel quale verranno inseriti tutti i requisiti.

I requisiti vengono ottenuti dalle seguenti fonti:

- Capitolato d'appalto  
(<http://www.math.unipd.it/~tullio/IS-1/2014/Progetto/C5.pdf>);
- Incontri con il proponente;
- Incontri con il committente;
- Incontri interni al gruppo.

Inoltre, devono essere realizzati i diagrammi UML dei casi d'uso e i seguenti tracciamenti:

- requisiti/casi d'uso;
- requisiti/test di validazione.

##### 2.1.1.2 Progettazione

##### 2.1.1.2.1 Specifica Tecnica

È compito dei progettisti descrivere l'architettura dell'applicazione e dei singoli componenti nella specifica tecnica. Inoltre devono realizzare:

- **Diagrammi UML:** devono essere forniti i seguenti diagrammi:
  - Diagrammi delle classi;
  - Diagrammi di sequenza;
  - Diagrammi di attività;

- Diagrammi dei package.
- **Tracciamento:** ogni requisito deve essere tracciato al componente che lo soddisfa;
- **Design pattern:** devono essere descritti i *design pattern* utilizzati per realizzare l'architettura. Ogni diagramma deve avere una breve descrizione e un diagramma che ne illustri il funzionamento;
- **Test di integrazione:** per verificare che tutti i componenti funzionino nella maniera prevista vengono definite delle classi di verifica.

#### 2.1.1.2.2 Definizione di Prodotto

La definizione di prodotto viene redatta dai progettisti, descrivendo in essa la progettazione di dettaglio del sistema, ampliando quanto scritto nella specifica tecnica. Nel documento saranno presenti:

- **Diagrammi UML:** devono essere forniti i seguenti diagrammi:
  - Diagrammi delle classi;
  - Diagrammi di attività;
  - Diagrammi di sequenza.
- **Definizione di classe:** ogni classe prodotta viene qui descritta, esplicitando il suo scopo e la funzione da essa modellata;
- **Tracciamento:** ogni requisito deve essere tracciato alle classi che lo soddisfano;
- **Test di unità:** i progettisti devono definire i *test* di unità necessari affinché si possa verificare che il sistema funzioni nel modo previsto.

### 2.1.2 Norme

#### 2.1.2.1 Classificazione dei requisiti

La tabella dei requisiti mette in evidenza tutti i requisiti emersi dal capitolato d'appalto o dagli incontri con il proponente.

I requisiti vengono catalogati in base alla loro importanza (obbligatori, desiderabili, facoltativi). Ogni requisito ha un suo codice identificativo, che viene costruito con una sigla seguita da un numero. La sigla serve per determinare il tipo di requisito ed è formata nel seguente modo:

$$R\langle X\rangle\langle Y\rangle\langle Z\rangle\langle Gerarchia\rangle$$

dove:

- **X** indica a quale parte del prodotto si riferisce il requisito, che può differenziarsi in
  - A: Applicazione mobile;
  - W: Piattaforma Web;
  - T: Tutte le parti del prodotto.
- **Y** indica la categoria del requisito, che può differenziarsi in
  - F: Funzionale;
  - P: Prestazionale;
  - Q: di Qualità;
  - V: di Vincolo.

- **Z** indica l'importanza rilevata per quel requisito, sia esso
  - O: Obbligatorio;
  - D: Desiderabile;
  - F: Facoltativo.
- **Gerarchia** indica il grado di parentela dei requisiti e andrà specificato nella sintassi:

$$a.b.c$$

dove il numero più a sinistra rappresenta il padre e quelli più a destra i figli.

### 2.1.2.2 Classificazione dei casi d'uso

Come per i requisiti anche per i casi d'uso il nome deve essere univoco e rispettare la forma:

$$UC\langle X \rangle \{n1.n2...n3\}$$

dove:

- **X** indica la parte del prodotto interessata ad un certo caso d'uso:
  - A: Applicazione mobile;
  - W: Piattaforma Web.
- Con "n1.n2...n3" si intende la gerarchia del caso d'uso. Il numero più a sinistra rappresenta il padre e quelli più a destra i suoi figli.

*Esempio: UCA2.2.4 indica un caso d'uso riguardante l'applicazione con padre UCA2.2, che a sua volta è figlio di UCA2.*

Ogni caso d'uso, al fine di essere descritto nel modo più esaustivo possibile, conterrà i seguenti dettagli:

- **Nome:** composto dal codice identificativo e una breve descrizione;
- **Rappresentazione grafica:** utilizzando il formalismo di UML;
- **Descrizione:** vengono fornite le seguenti informazioni:
  - Attori principali;
  - Scopo e descrizione;
  - Precondizione;
  - Postcondizione;
  - Illustrazione scenario principale;
  - Illustrazione scenario alternativo.

### 2.1.2.3 Nomi e norme stilistiche

- I commenti devono essere scritti in lingua italiana;
- La parentesi { deve essere scritta *in-line*;
- Le parentesi vanno usate anche quando opzionali;
- Dopo ogni parentesi grafa deve esserci un ritorno a capo, fatta eccezione per gli *statement else/else-if* e *catch/finally*;

- Dopo ogni punto e virgola deve esserci un ritorno a capo;
- Ogni riga può contenere al massimo 100 caratteri;
- Dopo le *keyword* *if*, *else*, *for*, *while*, *try*, *catch* va inserito uno spazio;
- I nomi delle classi devono avere la prima lettera maiuscola;
- I nomi delle componenti devono essere al singolare;
- I nomi di variabili, metodi, funzioni e componenti devono avere la prima lettera minuscola e le successive iniziali delle parole che ne compongono il nome, in maiuscolo;
- I nomi di tutte le classi, variabili, metodi e funzioni devono essere in inglese e privi di simboli;
- Le costanti vanno scritte tutte maiuscole separate da un *underscore*;
- La tabulazione va usata solo a inizio riga e solo per indentare codice.

### 2.1.3 Strumenti

#### 2.1.3.1 Android Studio

Il *software* Android Studio è scelto ed utilizzato dal gruppo per scrivere il codice dell'applicazione Android di *sHike*.

#### 2.1.3.2 Spring Tool Suite

Il *software* Spring Tool Suite è scelto ed utilizzato dal gruppo per scrivere il codice della piattaforma web, riguardante la parte Spring. Per quanto riguarda il server per l'esecuzione di Spring è scelto ed utilizzato Apache Tomcat v7<sup>1</sup>.

#### 2.1.3.3 HeidiSQL

Il *software* HeidiSQL è utilizzato per la creazione e la gestione del database remoto MySQL della prodotto *sHike*.

#### 2.1.3.4 Zeitrack

In modo da rendere automatici i tracciamenti, si è creato **Zeitrack**, uno strumento apposito disponibile a tutti i membri del gruppo tramite *login* al sito. Ogni utente dispone del proprio *account*. Il portale permette di gestire in maniera più automatizzata possibile tutti i dati e semplificare i tracciamenti. Zeitrack è accessibile tramite *browser*<sup>2</sup> e gestisce i dati direttamente da un *database* MySQL. Ogni dato può essere inserito, modificato ed eliminato direttamente dal portale.

Dalla barra del menu è possibile decidere in che sezione entrare tra Requisiti, Casi d'uso, Componenti, Classi, Test e Glossario. È stata resa automatica la generazione del codice  $\text{\LaTeX}$  per gli elenchi, così da poterli poi integrare direttamente nei documenti che li richiedono.

##### 2.1.3.4.1 Requisiti

Dalla pagina è possibile inserire, modificare ed eliminare un requisito. Quando si aggiunge un requisito è necessario completare tutti i campi.

---

<sup>1</sup><https://tomcat.apache.org/download-70.cgi>

<sup>2</sup><http://www.shike.it/zeitrack>

#### 2.1.3.4.2 Casi d'uso

La pagina permette l'inserimento di un caso d'uso, con relativi scenari e requisiti associati, collegando i requisiti che va a soddisfare. Ogni caso d'uso deve soddisfare almeno un requisito.

#### 2.1.3.4.3 Componenti

I componenti sono elencati in un'unica tabella, in ordine gerarchico col rispettivo identificativo. Nella pagina è possibile aggiungere una componente selezionando i requisiti associati e le interazioni con altre componenti, oltre al nome della stessa e la sua descrizione.

#### 2.1.3.4.4 Classi

La pagina permette l'inserimento di una classe, con relativa descrizione e nome. Inoltre, è possibile specificare le classi che implementa e la componente alla quale appartiene. È possibile specificare la relazione con altre classi ed aggiungere i metodi e gli attributi che appartengono alla classe.

#### 2.1.3.4.5 Test

Con Zeitrack è possibile tracciare tutti i test di integrazione, di sistema e di validazione. L'apposita pagina elenca tutti i test inseriti fino a quel momento, divisi in sotto-pagine per categoria, con la relativa descrizione.

#### 2.1.3.4.6 Tracciamento

Zeitrack permette di esportare molte parti dei documenti del progetto direttamente in  $\text{\LaTeX}$ , automatizzando il processo di stesura di questi. Il tracciamento delle parti è presente per le seguenti coppie di elementi:

- Requisiti - Casi d'uso
- Casi d'uso - Requisiti
- Requisiti - Test di Sistema
- Test di Sistema - Requisiti
- Test di Validazione - Requisiti
- Test di Integrazione - Componenti
- Requisiti - Componenti
- Componenti - Requisiti
- Componenti - Classi
- Classi - Componenti

#### 2.1.3.4.7 Glossario

Questo strumento permette l'inserimento, la modifica e rimozione di termini dal glossario, che vengono caricati in un database *MySQL*. Dalla pagina di gestione è possibile esportare i termini in  $\text{\LaTeX}$ . Il responsabile del documento deve poi provvedere ad inserire il codice all'interno del *template* e caricarlo nella *repository*.

## 3 Processi di supporto

### 3.1 Processo di documentazione

Durante le varie fasi del progetto sono stati utilizzati vari *standard* per la creazione dei documenti. In questa sezione verranno illustrati. I documenti sono diffusi all'interno del gruppo Zeitnot come spiegato nella sezione 4.1.1.5.

#### 3.1.1 Procedure

##### 3.1.1.1 Ciclo di vita di un documento

1. La prima impostazione del documento avviene in locale da parte del redattore. Una volta strutturato lo scheletro può caricare il file .tex nel *repository* come indicato nella sezione 4.1.1.8.
2. Dopo ogni modifica sostanziale viene incrementato il contatore Y (vedi sezione 3.1.2.2) con relativo *commit* nel *repository*.
3. Una volta ultimato il documento, questo viene inviato al verificatore come illustrato nella sezione 4.1.1.5. Il verificatore controllerà che siano state seguite le norme riguardanti la stesura dei documenti (vedi sezione 3.2). Il verificatore può dare due esiti, positivo o negativo.
4. In caso di esito **negativo**, il documento viene riproposto al redattore evidenziandone gli errori riscontrati. Se invece l'esito è **positivo**, il documento è stato redatto secondo le norme e non presenta errori, passa quindi al responsabile di progetto che provvede all'approvazione formale.
5. Il responsabile di progetto può dare a sua volta due giudizi: idoneo o non idoneo. Se **idoneo** il documento è da considerarsi formale e può essere inserito nella cartella corrispondente alla *milestone* presente nella cartella **Revisioni**. Se il documento risulta **non idoneo**, questo viene rimandato al redattore, notificando il motivo di tale decisione, ripartendo dal punto 3.

#### 3.1.2 Norme

##### 3.1.2.1 Template

Ogni documento viene generato a partire dal *template* L<sup>A</sup>T<sub>E</sub>X presente nella sotto cartella **shared** della cartella **Documents**. Questo *template* è stato creato prima della stesura dei documenti, qualsiasi modifica a tale file dovrà essere richiesta all'amministratore che valuterà la necessità di tale modifica.

Se una modifica al *template* dovesse essere approvata, tutti i membri del gruppo dovranno provvedere a rigenerare i propri documenti con il nuovo *template*.

##### 3.1.2.2 Nomenclatura documenti

Ogni documento deve essere individuabile in modo univoco, a tale scopo il nome del file è composto nel seguente modo:

nome\_file\_vX.Y

- **nome\_file**: il nome del file è scritto solo con lettere minuscole e non contiene caratteri speciali. Nel caso di un nome composto da più parole, queste saranno separate dal carattere *underscore*;
- **X.Y**: rappresenta la versione del file, che deve rispettare le norme descritte nella sezione 3.1.2.7.

### 3.1.2.3 Formattazione dei documenti

Nella prima pagina del documento deve essere presente il logo di Zeitnot e il nome del documento. Subito a seguire al nome del documento è presente una tabella contenente delle informazioni sul documento come:

- Nome del file;
- Versione;
- Data creazione;
- Data ultima modifica;
- Redattori;
- Approvato da;
- Verificatori;
- Stato;
- Uso.

Per redattori e verificatori, si intendono rispettivamente tutte le persone che hanno partecipato alla stesura del documento e che lo hanno verificato nella fase temporale conclusasi al momento della consegna di un documento. In altre parole, il periodo di tempo compreso tra due *milestone* progettuali.

A cominciare dalla seconda pagina è presente la tabella delle modifiche effettuate al documento. Le modifiche più recenti sono inserite nella parte superiore della tabella. Ogni voce dovrà indicare:

- Versione del documento;
- Descrizione dell'intervento;
- Autore;
- Ruolo dell'autore;
- Data di modifica.

A seguire, cominciando da una nuova pagina, l'indice dell'intero documento e, in caso della presenza di tabelle e immagini, è presente anche l'indice di quest'ultimi. Per decidere l'inserimento o meno di questi due indici basta impostare a *true* l'apposito *flag* all'interno del *template*.

Il documento deve essere suddiviso in sezioni e sottosezioni e se necessario in parti e paragrafi. Ogni sezione deve cominciare in una nuova pagina con il comando `clearpage`.

Ogni documento inoltre deve rispettare le seguenti regole:

- Nell'**intestazione** della pagina compare il logo Zeitnot nella parte destra, mentre a sinistra il nome della sezione della pagina;
- Nel **piè di pagina** c'è il numero della pagina a destra e il nome del documento con la sua versione a sinistra.

### 3.1.2.4 Norme tipografiche

#### 3.1.2.4.1 Punteggiatura

Prima di ogni segno di punteggiatura non deve essere presente uno spazio. Dopo un punto fermo deve esserci uno spazio e la parola successiva deve avere la prima lettera maiuscola. Subito dopo un'apertura di parentesi non ci deve essere uno spazio, prima di una chiusura non va nessun segno di punteggiatura e spaziatura.

#### 3.1.2.4.2 Elenchi

La prima parola di ogni elemento di un elenco deve essere con la lettera maiuscola. Se l'elemento in questione non contiene frasi complesse, deve terminare con il simbolo di punteggiatura ';' altrimenti si usa '.'. Si usa '.' anche nel caso in cui l'elemento sia l'ultimo dell'elenco.

#### 3.1.2.4.3 Stili di testo

- **Maiuscolo:** usato solo per acronimi e sigle;
- **Grassetto:** evidenzia elementi importanti all'interno del testo. Si usa inoltre anche per evidenziare l'argomento trattato dagli elenchi puntati;
- **Corsivo:** da enfasi a termini presenti nel testo, usato anche per parole in lingua straniera;
- **Sottolineato:** la sottolineatura viene utilizzata per indicare i termini presenti nel glossario. Tutte le occorrenze devono essere sottolineate, ad esclusione dei titoli.
- **Colori:** l'utilizzo di differenti colori per il testo è da limitarsi solamente nel documento "Definizione di Prodotto". Tali verranno utilizzati per marcare maggiormente i nomi e i tipi degli attributi, e i nomi, i parametri e i tipi di ritorno dei metodi. In particolare, verrà seguito questo stile:
  - *LimeGreen*: questo colore verrà utilizzato per i parametri dei metodi;
  - *ForestGreen*: questo colore verrà utilizzato per il nome degli attributi e per il nome dei metodi;
  - *Dandelion*: questo colore verrà utilizzato per il tipo di ritorno dei metodi e per il tipo degli attributi.

#### 3.1.2.4.4 Formati ricorrenti

- **Nomi:** tutti i nomi propri di persona devono essere nel formato Nome e Cognome;
- **Date:** ogni data deve utilizzare lo standard [ISO 8601], con il formato AAAA-MM-GG dove:
  - AAAA: sta ad indicare l'anno;
  - MM: sta ad indicare il mese;
  - GG: sta ad indicare il giorno.
- **Orari:** ogni orario deve utilizzare lo standard [ISO 8601], con il formato HH:MM dove:
  - HH: indica il numero delle ore. Per la rappresentazione vengono usate esattamente due cifre.
  - MM: indica il numero dei minuti. Per la rappresentazione vengono usate esattamente due cifre.
- **Sigle:** sono presenti delle sigle ricorrenti all'interno dei documenti o delle immagini, qui di seguito elencate per evitare ambiguità:



- RR: Revisione dei Requisiti;
- RP: Revisione di Progettazione;
- RQ: Revisione di Qualifica;
- RA: Revisione di Accettazione;
- AdR: Analisi dei Requisiti;
- PdP: Piano di Progetto;
- PdQ: Piano di Qualifica;
- NdP: Norme di Progetto;
- ST: Specifica Tecnica;
- DdP: Definizione di Prodotto.

### 3.1.2.5 Componenti

#### 3.1.2.5.1 Immagini

Tutte le immagini presenti nei documenti devono avere estensione `.png` o `.jpg`. Le immagini devono essere posizionate nella cartella **Documents/images**. Se le immagini hanno una risoluzione troppo elevata vanno ridimensionate.<sup>3</sup>

Tutte le immagini devono avere un nome significativo che permetta di identificarle in modo univoco. Sono ammesse solo lettere e cifre, oltre al carattere *underscore* che sostituisce lo spazio. In particolare, le immagini che contengono i diagrammi UML dovranno essere contenute nella sotto cartella `uml` e rispettare la seguente regola:

TIPO-X1-X2-(...)-Xn

Il tipo deve corrispondere ad una di queste sigle:

- **UC<X>**: diagrammi casi d'uso (riferimento sez. 2.1.2.2);
- **CD**: diagrammi delle classi;
- **OD**: diagrammi degli oggetti.

X1-X2-(...)-Xn invece rappresenta la gerarchia dei diagrammi, la variabile più a sinistra è il padre dei figli a destra.

L'inserimento di immagini all'interno del documento viene fatto tramite l'utilizzo del comando `\begin{figure}` e `\includegraphics[parametri]{riferimento}` dove:

- **Parametri**: qui si possono settare vari parametri da utilizzare per formattare l'immagine. Ad esempio:
  - `[scale=0.7]` indica un ridimensionamento del 30%;
  - `[width=4cm, heigh=5cm]` indica un'immagine larga 4 cm e alta 5 cm;
  - `[width=\textwidth]` la figura deve corrispondere esattamente alla larghezza del testo.
- **Riferimento**: si intende il nome del file, senza estensione, presente nella cartella **images** (vedi sezione 4.1.3.2.2).

Prima della chiusura dell'immagine `\end{figure}` vanno inseriti anche i comandi `\caption{didascalia}` e `\label{riferimento}` per permettere l'indicizzazione e l'identificazione delle immagini.

<sup>3</sup>Software gratuito: <http://luci.criosweb.ro/riot/>.

### 3.1.2.5.2 Inserimento tabelle

Anche per le tabelle come per le immagini, vanno inseriti, prima della chiusura della tabella, i comandi `\caption{didascalia}` e `\label{riferimento}` per permetterne l'indicizzazione e l'identificazione.

Come *tag* opzionale, è possibile inserire subito prima dell'apertura della tabella: `\rowcolors{2}{grey}{white}`. Il *tag* serve per alternare i colori di sfondo delle righe, rendendo la tabella più leggibile.

### 3.1.2.5.3 Inserimento link

L'inserimento di link a pagine web avviene tramite il comando

`\url{link ipertestuale}`

dove con *link ipertestuale* indica l'indirizzo nella forma estesa

`http://www.dominio.com/estensione/...`

### 3.1.2.6 Tipi di documento

- **Informali:** tutti i documenti che non sono ancora stati approvati dal responsabile di progetto o non ancora completati, saranno marcati come informali. Questa tipologia di documenti è disponibile solo ad uso interno ed è vietata la distribuzione all'esterno del gruppo Zeitnot. I file sorgenti \*.tex sono reperibili all'interno della cartella **Documents** della *repository* come descritto nella sezione 4.1.3.2.2.
- **Formali:** una volta approvati dal responsabile i documenti diventano formali e quindi pronti alla distribuzione verso l'esterno. Sono reperibili nella cartella **Revisioni** nella *repository*, organizzati come descritto nella sezione 4.1.3.2.2.
- **Verbali:** tutti i verbali sono disponibili alla consultazione nella cartella **verbali** all'interno di **Documents** nella *repository*, mentre la struttura dei verbali deve seguire le linee guida definite nella sezione 3.1.2.6.1.
- **Glossario:** nel glossario è possibile trovare tutte le parole difficili da comprendere o che potrebbero essere fraintese. In questo documento non è presente nessun indice e le voci sono ordinate in ordine alfabetico. Per facilitare la creazione del glossario è stata definita una *utility* all'interno del sito *Zeitrack* (vedi sezione 2.1.3.4.7).

#### 3.1.2.6.1 Struttura verbali

Ogni incontro deve avere il suo verbale impostato con il template `_verbale` nella cartella **verbali**. Il nome del verbale è formattato come segue:

*incontro\_AAAA-MM-GG*

Ogni verbale contiene:

- **Data:** nel formato AAAA/MM/GG;
- **Ora:** nel formato HH:MM;
- **Luogo:** indirizzo incontro/telematico;
- **Partecipanti:** tutti i nomi dei presenti sotto forma di elenco puntato;
- **Oggetto incontro:** elenco delle questioni discusse;
- **Risultati:** constatazioni finali sulle tematiche discusse.

### 3.1.2.7 Versionamento dei documenti

Ogni volta che si effettua una modifica e questa viene segnata nella storia delle modifiche, va assegnato un nuovo numero di versione rispettando le seguenti regole:

- La versione iniziale di un documento è la 0.1;
- L'approvazione del documento incrementa l'unità del primo indice ed imposta il secondo a 0;
- Una modifica qualunque incrementa il secondo indice lasciando invariato il primo.

### 3.1.3 Strumenti

I documenti devono essere scritti in italiano e in  $\text{\LaTeX}$ . Il *software* di editor utilizzato è TeXstudio, programma *open source* e multi-piattaforma. *TeXstudio* è un *software* molto flessibile e permette di adattarsi al meglio alle esigenze del gruppo, fornendo strumenti come la correzione ortografica automatica, l'anteprima del PDF generato e l'auto completamento dei comandi. La versione è scaricabile al seguente *link*:

`http://texstudio.sourceforge.net/#download`.

Oltre all'*editor* è necessario scaricare anche il compilatore  $\text{\LaTeX}$ , disponibili al link:

`http://miktex.org/download`.

## 3.2 Processo di verifica

La verifica è un'attività fondamentale svolta dal verificatore, con lo scopo di accertare che i prodotti siano conformi alle norme riportate su questo documento. Inoltre il verificatore dovrà confermare che tali prodotti sono conformi ai requisiti descritti nel Piano di Qualifica.

### 3.2.1 Attività

#### 3.2.1.1 Analisi

##### 3.2.1.1.1 Analisi statica

L'analisi statica prevede controlli basati sulla non esecuzione del codice, ovvero su qualsiasi tipo di prodotto anche non eseguibile (ad esempio documentazione). Per questo tipo di analisi si utilizzeranno due forme di analisi statica: il controllo manuale e il controllo tramite strumenti automatici.

Per quello che riguarda il primo caso, chiamato anche *desk check* ci sono due varianti:

- **Walkthrough:** esame ad ampio spettro eseguito dal verificatore. Questa tecnica è molto onerosa sia a livello di tempo che di sforzo. Inoltre può essere essa stessa soggetta ad errori. Tuttavia, nella fase iniziale del progetto, è l'unica soluzione, vista la poca esperienza da parte dei membri del gruppo nella gestione di progetti di grandi dimensioni e complessità. Per risolvere questo problema è previsto che, durante le operazioni di verifica, sia stilata una lista di controllo degli errori più frequenti e delle aree più critiche in modo tale da creare una base solida destinata a potenziare le future attività di ispezione.
- **Inspection:** consiste in una verifica mirata a codice o a documentazione. Questa è possibile solo dopo che si è raggiunto un elevato grado di conoscenza e si è consapevoli degli errori più comuni e delle aree più critiche identificate tramite la fase di *walkthrough*. Questa tecnica è meno dispendiosa in termini di risorse visto che non è necessaria un'analisi completa ma mirata solo ad alcune parti.

##### 3.2.1.1.2 Analisi dinamica

L'analisi dinamica è effettuata tramite controlli dinamici, detti *test*, e consiste nel testare le parti *software* in un ambiente controllato, con dati di *input* specificatamente pensati per testare le funzionalità e il rispetto dei requisiti, mettendo in luce eventuali difetti. Una caratteristica fondamentale dei *test* è la loro ripetibilità. Infatti, dato lo stesso *input* e nella stessa situazione di esecuzione, l'*output* deve essere deterministico e univocamente determinato. Grazie a questi *test* è possibile identificare errori e passare alla loro risoluzione. Nel caso in cui il *test* non segnali errori da correggere non significa necessariamente che non ce ne siano. Per ogni *test* deve essere definito:

- **Ambiente:** l'*hardware* e il *software* sui quali è pianificata l'esecuzione del *test*;
- **Specifica:** il dettaglio degli *input* e *output* attesi;
- **Procedure:** la specifica di come va eseguito il *test* e di come vanno interpretati i risultati.

### 3.2.1.2 Test

#### 3.2.1.2.1 Test di unità

Il *test* di unità ha come oggetto le singole unità, per verificarne modulo e dati di esempio, ma può anche coinvolgere altre parti del sistema attive o passive, che siano in grado di simulare le parti del sistema non ancora disponibili nel momento in cui il *test* viene eseguito.

I *test* di unità possono essere identificati grazie alla seguente sintassi:

TU-[Parte riferimento]-[Classe appartenenza]-[Codice Requisito]

La classe di appartenenza corrisponde alla classe in cui è possibile trovare il test, mentre la parte prodotto indica se il test si riferisce all'applicazione mobile (A), alla piattaforma web (W). Il codice requisito è un numero progressivo per identificare in modo univoco un test all'interno della classe di appartenenza.

#### 3.2.1.2.2 Test di integrazione

Il *test* di integrazione valuta che due o più componenti, una volta assemblati, funzionino come previsto. Tale *test* va a creare una *build*, ovvero un sottosistema funzionante, che può essere eseguito in modo indipendente.

TI[Parte prodotto]-[Identificativo del componente]

Tale identificativo corrisponde alla componente i cui elementi sono integrati, mentre la parte prodotto indica se il test si riferisce all'applicazione mobile (A), alla piattaforma web (W) o ad entrambe le parti (T).

#### 3.2.1.2.3 Test di sistema

Il *test* di sistema verifica che l'intero sistema rispetti tutti i requisiti identificati nella fase di analisi dei requisiti.

TS-[Tipo Requisito] [Codice Requisito]

Il tipo e il codice si riferiscono al requisito di cui verrà testato il soddisfacimento.

#### 3.2.1.2.4 Test di regressione

Il *test* di regressione viene effettuato subito dopo che una componente viene modificata. Consiste nel rieseguire tutti i *test* per verificare che dopo le modifiche il resto dei moduli continui a funzionare.

#### 3.2.1.2.5 Test di validazione

Il *test* di validazione consiste nel verificare, in presenza del committente, l'aderenza del prodotto ai requisiti richiesti.

TV-[Tipo Requisito] [Codice Requisito]

Il tipo e il codice si riferiscono al requisito padre di cui verrà testato il soddisfacimento.

### 3.2.1.3 Tracciamento

La verifica della corrispondenza di ogni requisito con una o più fonti è compito del verificatore. In caso di errori o incongruenze bisogna seguire quanto descritto dalla sottosezione 3.2.2.1.

## 3.2.2 Procedure

### 3.2.2.1 Gestione delle anomalie

Il documento è pronto alla verifica non appena il redattore ritiene che quest'ultimo sia completato. Nel caso in cui il verificatore dovesse riscontrare un'anomalia, il procedimento di gestione si suddivide nelle seguenti fasi:

1. Se gli errori riscontrati sono di tipo logico, il verificatore provvede ad inviare un *issue* per ogni errore, creato tramite l'apposito strumento di GitHub<sup>4</sup>. È compito del redattore poi effettuare la correzione per una successiva verifica. Durante il *commit* si deve segnalare quale errore, o errori se più di uno, si vanno a correggere. Per fare ciò bisogna inserire nella descrizione o nel titolo la seguente stringa:

<sup>4</sup><https://github.com/tixgit/sHike/issues>

termine #ID

dove:

- **Termine** indica una delle parole riconosciute da GitHub per la chiusura delle *issue* tramite *commit*, come ad esempio *close*, *fix* e *resolve*<sup>5</sup>;
  - **ID** indica l'id della *issue*.
2. Dopo la correzione il verificatore provvede a controllare, tramite tecnica *inspection*, che il documento sia valido.

### 3.2.3 Strumenti

#### 3.2.3.1 Correzione ortografica degli errori

Si effettua una completa analisi ortografica, valutando il corretto uso della punteggiatura. Per fare ciò il verificatore utilizza il correttore automatico presente in TeXstudio, raggiungibile dal menu “Strumenti/Controllo ortografico..”. Dopo quest’analisi effettua un controllo sintattico, verificando che i periodi siano ben strutturati e con un senso logico compiuto.

#### 3.2.3.2 Calcolo Gulpease

Per calcolare la leggibilità di un documento si utilizza l’indice Gulpease. Uno degli script creati dal gruppo provvede al calcolo di tale indicatore. Lo script è stato scritto in python<sup>6</sup>. La linea di comando da eseguire è:

```
python gulpse [nomedocumento]
```

l’output del comando è un numero, l’indice di Gulpease relativo al file `nomedocumento.pdf`.

#### 3.2.3.3 Strumenti per la validazione W3C

Per il portale web vengono utilizzati gli strumenti di validazione online <http://validator.w3.org/> per le pagine web e <http://jigsaw.w3.org/css-validator/> per i fogli di stile.

---

<sup>5</sup>Qui la lista completa: <https://help.github.com/articles/closing-issues-via-commit-messages/>

<sup>6</sup>Download al link: <http://www.python.it/download/>

## 4 Processi organizzativi

### 4.1 Processo di gestione

#### 4.1.1 Attività

##### 4.1.1.1 Incontri Interni

Gli incontri vengono fissati dal responsabile e possono essere di due tipi: incontri di persona ed incontri telematici. Gli incontri telematici vengono effettuati tramite servizio di video chiamata online. In entrambi i casi ogni persona deve rispondere al responsabile indicando la sua presenza o meno all'incontro (vedi sezione 4.1.4.4 **Doodle Incontri**). Nel caso in cui un membro non sia disponibile ad un incontro è sua responsabilità informarsi ed aggiornarsi su quanto avvenuto. Un modo è sfruttare il verbale degli incontri, presente all'interno della cartella GitHub Documents/verbali.

##### 4.1.1.2 Incontri Esterni

Fissare un incontro con il committente o proponente è a carico del responsabile. Chiunque può richiedere al responsabile un incontro esterno specificandone il motivo. L'utilità o meno dell'incontro è valutata dal responsabile, presentando la necessità al resto del gruppo. Con tre voti favorevoli all'incontro questo è fissato, in caso contrario la proposta d'incontro viene respinta.

##### 4.1.1.3 Comunicazione Interna

I contatti di ogni membro del gruppo sono disponibili all'interno della cartella Drive **Varie**. Le comunicazioni importanti avvengono tramite *email*, mentre per le questioni più immediate vengono utilizzati servizi di messaggistica istantanea.

##### 4.1.1.4 Comunicazione Esterna

Il responsabile del gruppo si occupa della comunicazione con enti esteri al gruppo. A tale scopo è stata creata un' *email* di gruppo: zeitnotswe@gmail.com. Tutti i membri del gruppo ricevono le risposte inviate a quell' *email* grazie ad un inoltro automatico.

##### 4.1.1.5 Diffusione documenti

Per la diffusione dei documenti all'interno del gruppo si è deciso di utilizzare la posta elettronica allegando il documento su *Google Drive*. Per inoltrare ai membri del gruppo un documento è sufficiente caricarlo su *Google Drive* nell'apposita cartella. Quando si invia un' *email* da *Google Gmail*, una volta cliccato sull'icona di Drive, basta selezionare il file caricato in precedenza o caricarlo direttamente dalla finestra di dialogo apertasi. Fatto ciò, cliccando su *Inserisci* viene fornito in automatico il link all'allegato presente su Drive.

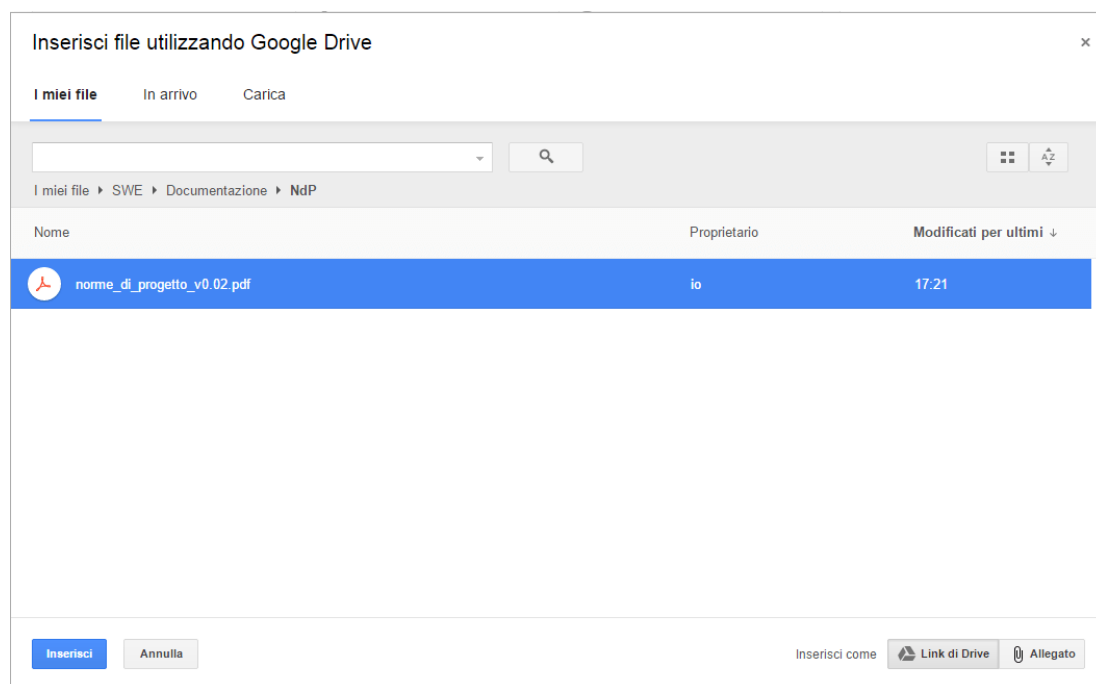


Figura 1: Inserimento di un allegato tramite Google Drive

Questa tecnica permette di mantenere tutti gli allegati ordinati su *Google Drive* e non sparsi sulle varie *email* dell'*account* di posta.

Tutto questo è stato possibile dopo aver constatato che tutti gli utenti del gruppo usavano Gmail e disponevano di un *account* Google.

#### 4.1.1.6 Milestone

Come prima cosa le *milestone* previste dal progetto vengono inserite dal responsabile di progetto su TeamWork<sup>7</sup>. Le voci presenti corrispondono a tutte le *milestone* previste dal progetto, ossia le quattro revisioni.

#### 4.1.1.7 Task

Con la parola *task* si va ad indicare un compito da svolgere. Ogni *task* deve avere una lista di appartenenza ed ogni lista deve far riferimento ad una *milestone*.

#### 4.1.1.8 Repository

Per svolgere al meglio il progetto è necessario trovare uno strumento con il quale sia possibile il caricamento e la consultazione di tutti i documenti e i file di codifica. Per questo si è scelto di usare una *repository*, in particolare quella offerta da GitHub. GitHub offre un servizio di versionamento integrato, il gruppo non deve quindi utilizzare un'ulteriore servizio separato dalla *repository*.

### 4.1.2 Procedure

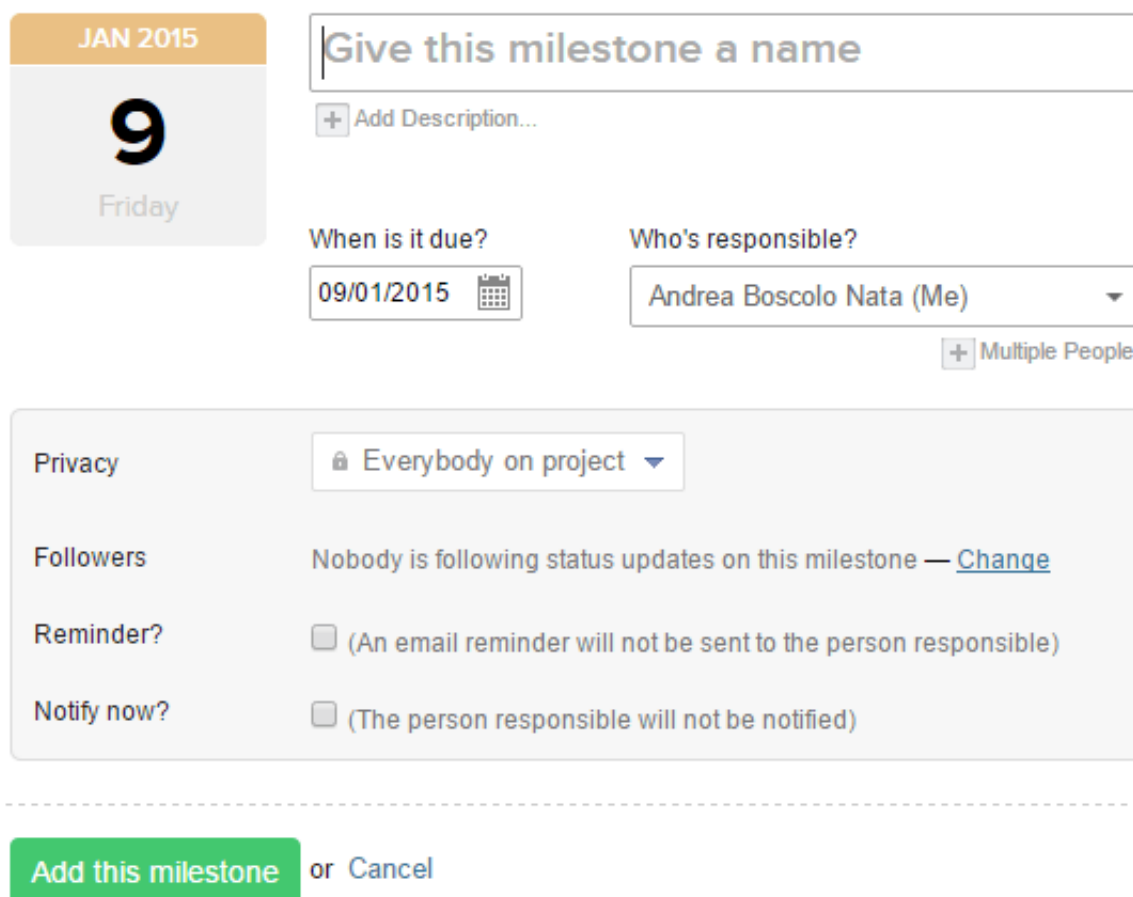
#### 4.1.2.1 Creazione nuova Milestone

Per inserire una nuova *milestone* prima si deve selezionare la pagina *milestone* e successivamente cliccare su "Add a milestone" dalla pagina di TeamWork.

<sup>7</sup> Dettagli presenti nella sotto-sezione 4.1.4.5



## Add a new milestone



JAN 2015

9  
Friday

Give this milestone a name

+ Add Description...

When is it due?  
09/01/2015

Who's responsible?  
Andrea Boscolo Nata (Me)

+ Multiple People

Privacy  
Everybody on project

Followers  
Nobody is following status updates on this milestone — [Change](#)

Reminder?  
☐ (An email reminder will not be sent to the person responsible)

Notify now?  
☐ (The person responsible will not be notified)

Add this milestone or [Cancel](#)

Figura 2: Aggiunta di di una nuova Milestone

Il form va completato nel seguente modo:

- Nella prima riga va inserito il **nome** della *milestone*;
- **When is it due?**: la data entro il quale la *milestone* deve essere completata;
- **Who's responsible?**: il responsabile per quella *milestone*.

Gli altri campi sono opzionali e a discrezione del responsabile del progetto. È consigliabile inserire nella lista dei *followers* tutti i membri del gruppo.

### 4.1.2.2 Creazione nuova lista Task

Per inserire un nuovo *task* è prima necessario creare la lista ad esso annessa su TeamWork, se questa non presente. Dopo aver selezionato la pagina *task* si clicca su “Add task list”.

- Il **nome della lista** deve essere qualcosa di significativo, se si sta creando una lista di *task* riguardanti un documento, il nome della lista è il nome del documento;

## New task list ✕

Give the list a name (eg. "Changes to layout" or "Bug fixes / typos")

Notes Privacy Milestone Advanced

Does this list relate to a milestone?

RR (Due in 14 days - 23/01/2015 ) ▼

+ Create a new milestone...

Add this task list

 or [Cancel](#)

Figura 3: Creazione di una nuova lista Task

- All'interno della *label* "Milestone" è necessario selezionare la milestone collegata alla lista dei task che si va a creare;
- Tutte le altre informazioni possono essere completate se si ritiene opportuno dare maggiori dettagli.

#### 4.1.2.3 Creazione nuovo Task

Una volta creata la lista task è possibile inserire al suo interno un nuovo compito. Dopo aver selezionato la pagina *Tasks* cliccare sul bottone "Add task" corrispondente alla lista nella quale si desidera inserire il nuovo compito.

Fatto ciò, bisogna compilare i dati richiesti:

- **What needs to be done?** inserire il nome del compito da svolgere;
- **Who should do this?** selezionare chi deve svolgere tale compito, la selezione multipla è ammessa;
- *Opzionale*: selezionare la data di inizio e/o completamento task.

### ▼ Analisi dei Requisiti

Linked to completed milestone RR

Tiziano L. + 2... Tabella dei requisiti (Start: 6 days ago, Sat Jan 3rd) (Due: 6 days ago, Sat Jan 3rd)

Tiziano L. + 2... Casl d'uso (Due: in 14 days, Fri Jan 23rd)

+ | What needs to be done?

Who should do this?  

Anyone

☐ Notify by email? 

+ Multiple People

Start Date (optional)

Today | +1 Day | +1 Week | Mon | No Date

Save my changes or Cancel

Figura 4: Creazione di un nuovo Task

Cliccando su *Save my changes* il *task* viene memorizzato all'interno della lista. È possibile anche creare un *sub-task*:

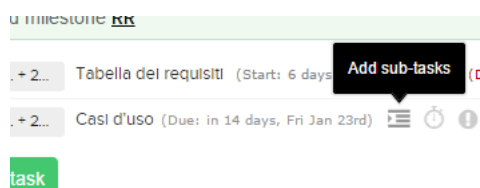


Figura 5: Inserimento di un nuovo sub-task

### 4.1.3 Norme

#### 4.1.3.1 Ruoli di Progetto

In un progetto si prevede la collaborazione di persone con ruoli differenti. Tali ruoli rappresentano le varie figure aziendali, indispensabili per il buon esito del progetto. Durante l'intera durata del progetto si garantisce che ogni componente del gruppo ricoprirà tutti i ruoli almeno una volta. Dovranno essere evitati conflitti d'interesse (esempio: dopo aver prodotto un certo documento, essere il verificatore dello stesso). Le attività saranno pianificate con attenzione. Sarà poi compito del verificatore controllare che la pianificazione rispetti le considerazioni di cui

sopra. Se si manifesteranno incongruenze, queste verranno segnalate al responsabile che avrà la responsabilità di risolvere la questione.

#### **4.1.3.1.1 Responsabile**

Il responsabile rappresenta il progetto nelle sue parti e detiene il potere decisionale, quindi la responsabilità riguardo:

- Gestione e controllo delle risorse;
- Analisi e gestione dei rischi;
- Pianificazione e controllo delle attività;
- Approvazione dei documenti;
- Approvazione dell'offerta economica.

I compiti del responsabile sono:

- Garantire che non vi siano conflitti di interesse tra redattori e verificatori.
- Assicurarsi che le attività di verifica vengano svolte seguendo le “Norme di Progetto”.
- Garantire che vengano rispettati i ruoli e le competenze assegnate dal “Piano di Progetto”.

#### **4.1.3.1.2 Amministratore**

L'amministratore è responsabile di tutto ciò che riguarda l'ambiente di lavoro. Le sue mansioni riguardano:

- Gestire l'archiviazione e il versionamento della documentazione di progetto;
- Ricercare strumenti che possano automatizzare il maggior numero di compiti e operazioni;
- Redigere il documento “Norme di Progetto” nelle quali è spiegato e regolato l'uso degli strumenti.

#### **4.1.3.1.3 Analista**

L'analista è il responsabile delle attività di analisi. Le mansioni previste da questo ruolo sono le seguenti:

- Comprendere la natura e la complessità del problema;
- Produrre una specifica di progetto che sia completa e chiara in ogni suo punto, sia dal proponente, sia dal committente, sia dai progettisti;
- Redigere il documento “Analisi dei Requisiti” e “Studio di Fattibilità”.

#### **4.1.3.1.4 Progettista**

Il progettista è il responsabile delle attività di progettazione. Le mansioni di questo ruolo sono le seguenti:

- Effettuare scelte progettuali in modo da rendere il prodotto facilmente manutenibile in futuro;
- Produrre una soluzione comprensibile, attuabile e motivata;
- Redigere la “Specificazione Tecnica” e la “Definizione del prodotto”.

#### 4.1.3.1.5 Programmatore

Il programmatore è il responsabile delle attività di codifica. Le mansioni previste da questo ruolo sono le seguenti:

- Scrivere codice che sia documentato, versionato e manutenibile;
- Rispettare le metriche e le norme riguardanti la scrittura del codice;
- Implementare in maniera rigorosa le soluzioni del progettista;
- Redigere il “Manuale Utente”.

#### 4.1.3.1.6 Verificatore

Le attività di verifica sono compito del verificatore. Le sue mansioni riguardano:

- Controllare la conformità di ogni stadio del ciclo di vita del prodotto;
- Garantire che le attività siano conformi alle norme stabilite.

#### 4.1.3.2 Repository

Per gestire la *repository* in modo più semplice, viene utilizzato il *software* di GitHub fornito gratuitamente al seguente *link*:

<https://windows.github.com/>

##### 4.1.3.2.1 Registrazione

Per poter utilizzare GitHub è stato necessario effettuare una registrazione:

<https://github.com>

È stata sfruttata la possibilità offerta da GitHub, tramite uno *Student Pack*, di creare un *micro account* con 5 *repository* private. L'*account* rimarrà gratuito fino a quando l'iscritto risulti essere uno studente. Non essendo possibile creare un'organizzazione e ottenere l'*account* gratuito, la *repository* è stata creata nell'*account* di un membro del gruppo che poi ha invitato gli altri membri del gruppo come collaboratori.

##### 4.1.3.2.2 Struttura

- **Progetto:** <https://github.com/tixgit/sHike>;
- **Revisioni:** <https://github.com/tixgit/sHike/Revisioni>  
Cartella contenente quattro sotto cartelle (RR, RP, RQ, RA) che contengono a loro volta i documenti formali consegnati alle revisioni del progetto;
- **Documentazione:** <https://github.com/tixgit/sHike/Documents>  
Cartella contenente tutti i documenti del progetto. Ogni documento in formato L<sup>A</sup>T<sub>E</sub>X deve essere contenuto in una sotto cartella, avente come nome il nome del documento stesso. Tutte le immagini vanno inserite all'interno della cartella *images*, presente in *Documents*;
- **Codice:** <https://github.com/tixgit/sHike/Code>  
Qui sono disponibili tutti i file sorgente;
- **Others:** ogni altro file che non rientra in nessuna delle precedenti categorie è caricato nella cartella *Others*.

#### 4.1.3.2.3 File ignorati (.gitignore)

All'interno della *repository* di GitHub è stato anche caricato un file `.gitignore` che permette di escludere il caricamento di determinati file effettuando i *commit*. Questo strumento si rivela molto utile per escludere i file temporanei creati durante la compilazione di documenti L<sup>A</sup>T<sub>E</sub>X.

#### 4.1.4 Strumenti

##### 4.1.4.1 Google Calendar

Ogni avvenimento importante, come scadenze relative al progetto e incontri fissati dal responsabile, sono appuntati su un calendario condiviso presente nell'*account* Google<sup>8</sup> di ogni membro del gruppo.

##### 4.1.4.2 Diagrammi di Gantt

Sfruttando la possibilità offerta dall'università con MSDN-AA di ottenere *software* gratuitamente, si è deciso di utilizzare il Microsoft Project 2013 che permette una semplice ed efficace gestione dei diagrammi di Gantt.

##### 4.1.4.3 Diagrammi UML

Come strumento di modellazione per i diagrammi UML è stato scelto **Astah**<sup>9</sup> nella versione *Community*. Il *software* presenta un'interfaccia semplice ed intuitiva permettendo di creare tutti i tipi di diagrammi necessari al gruppo. Astah consente anche l'esportazione nel formato immagine png.

##### 4.1.4.4 Doodle Incontri

Per facilitare la programmazione degli incontri è stato utilizzato un servizio che ne permette la pianificazione<sup>10</sup>. **Doodle** permette di creare dei sondaggi per valutare il momento più adatto per effettuare un incontro in base alla disponibilità data dai membri del gruppo.



Figura 6: Esempio per la gestione degli incontri

<sup>8</sup><https://www.google.com/calendar>

<sup>9</sup><http://astah.net/editions/community>

<sup>10</sup><http://doodle.com/>

#### 4.1.4.5 TeamWork

Per favorire una corretta gestione delle operazioni di assegnazione dei lavori è stato scelto un servizio esterno, *Team Work*<sup>11</sup>. Questo *tool* permette la creazione di *milestone* e *task*, strumenti essenziali per tener traccia del lavoro svolto dal gruppo, permettendo di delineare una roadmap degli obiettivi, sia a lungo che a breve termine.

#### 4.1.4.6 Google Drive

Google Drive viene utilizzato solamente come supporto per il gruppo Zeitnot, nessuna porzione di codice o file che necessita di versionamento verrà mai salvata qui. La principale funzione di questo *cloud* è lo scambio di file tra i membri del gruppo, in sostituzione agli allegati *email*, come descritto nella sezione 4.1.1.5. Si è scelto Drive perché, oltre ad offrire gratuitamente lo spazio, permette anche un rapido collegamento con gli allegati delle *email* di Gmail.

#### 4.1.4.7 Sistema operativo

Lo strumento principale utilizzato per la codifica è Android Studio, multi piattaforma e disponibile per Windows, Mac OS e Linux. Il progetto viene implementato dal gruppo attraverso sistemi Windows, in particolare Windows 7 e Windows 8.1.

#### 4.1.4.8 Prezi

Per le presentazioni formali del gruppo si è deciso di utilizzare il software Prezi<sup>12</sup>. Prezi è uno strumento che permette di creare presentazioni direttamente online, apportando modifiche in modo concorrente.

---

<sup>11</sup><https://zeitnot.teamwork.com/>

<sup>12</sup><http://prezi.com/>

## A Lista di controllo

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

### A.1 Norme Stilistiche

- Elenchi: non terminano con il punto se è l'ultimo elemento;
- Elenchi: non iniziano con la lettera maiuscola;
- Nome documento: non viene indicata la versione del documento dove necessaria;
- Glossario: non viene evidenziata una parola che appartiene al glossario;
- Corsivo: non viene posto in corsivo un termine straniero.

### A.2 L<sup>A</sup>T<sub>E</sub>X

- Spazi: non viene inserito uno spazio dopo le macro che lo richiedono;
- Riferimenti: dopo l'eliminazione o la modifica di una sezione rimangono dei riferimenti non corretti, visualizzando nel documento '??'.

### A.3 Italiano

- Carattere 'È': non viene scritto correttamente utilizzando il comando apposito;
- Periodi: frasi troppo lunghe rendono i concetti di difficile comprensione.

### A.4 UML

- Attori: il sistema non è un attore;
- Non viene rimosso il simbolo di non navigabilità dai diagrammi delle classi generati automaticamente;
- Generalizzazione scambiata per estensione o inclusione e viceversa.

### A.5 Codice Java

- Non rispetto delle norme stilistiche adottate;
- Dimenticarsi il private su metodi e attributi che nella definizione di prodotto erano segnati come privati;
- Omissione del carattere ';' per terminare uno *statement*.

#### A.5.1 Spring Tool Suite

- Dimenticare di creare il "bean" corrispondente ad un DAO.

#### A.5.2 Android Studio

- Controllare che i pulsanti ai quali si cerca di accedere siano stati istanziati.