

# SWEENEYTHREADS

## ACTORBASE

A NoSQL DB BASED ON THE ACTOR MODEL

---

## Norme di progetto

---

*Redattori:*

Nicoletti Luca

Tommasin Davide

*Approvazione:*

Maino Elia

*Verifica:*

Bonato Paolo



Versione 2.0.4

13 maggio 2016

# Indice

<b>1</b>	<b>Introduzione</b>	<b>4</b>
1.1	Scopo del documento . . . . .	4
1.2	Scopo del prodotto . . . . .	4
1.3	Glossario . . . . .	4
1.4	Riferimenti . . . . .	5
1.4.1	Informativi . . . . .	5
1.4.2	Normativi . . . . .	6
<b>2</b>	<b>Processi primari</b>	<b>7</b>
2.1	Fornitura . . . . .	7
2.1.1	Studio di fattibilità . . . . .	7
2.2	Sviluppo . . . . .	7
2.2.1	Casi d'uso . . . . .	7
2.2.2	Tecniche di analisi e classificazione requisiti . . . . .	9
2.2.3	Gestione dei requisiti . . . . .	10
2.2.4	Tracciamento dei requisiti . . . . .	11
2.2.5	Gestione cambiamento requisiti . . . . .	11
2.2.6	Progettazione architetturale . . . . .	11
2.2.7	Definizione testuale delle componenti architettureali . . . . .	11
2.2.8	Diagrammi delle componenti architettureali . . . . .	12
2.2.9	Design Pattern . . . . .	13
2.2.10	Codifica in <i>LaTeX</i> . . . . .	13
2.2.11	Codifica in <i>Scala</i> . . . . .	14
2.2.12	Integrazione del software . . . . .	14
<b>3</b>	<b>Processi di supporto</b>	<b>16</b>
3.1	Documentazione . . . . .	16
3.1.1	Template . . . . .	16
3.1.2	Struttura documenti . . . . .	16
3.1.3	Prima pagina . . . . .	16
3.1.4	Indice . . . . .	17
3.1.5	Diario delle modifiche . . . . .	17
3.1.6	Formattazione generale delle pagine . . . . .	17
3.1.7	Norme tipografiche . . . . .	17
3.1.8	Stile del testo . . . . .	18
3.1.9	Formati . . . . .	18
3.1.10	Sigle . . . . .	18
3.1.11	Componenti grafiche . . . . .	19
3.1.12	Classificazione documenti . . . . .	19
3.1.13	Versionamento documenti . . . . .	19
3.1.14	Ciclo di vita dei documenti . . . . .	20
3.2	Accertamento della qualità . . . . .	20
3.2.1	Analisi dei processi . . . . .	20
3.2.2	Analisi dei documenti . . . . .	20
3.3	Gestione delle versioni . . . . .	21
3.4	Qualifica . . . . .	22
3.4.1	Procedure di controllo qualità per i processi . . . . .	22
3.4.2	Procedure di controllo qualità per i documenti . . . . .	22
3.4.3	Procedure di controllo qualità per il prodotto . . . . .	22
3.5	Strumenti e metodi per l'applicazione delle metriche . . . . .	22
3.6	Comunicazione e risoluzione di anomalie . . . . .	23

<b>4</b>	<b>Processi organizzativi</b>	<b>25</b>
4.1	Processi di gestione dell'infrastruttura . . . . .	25
4.1.1	Documentazione di pianificazione . . . . .	25
4.1.2	Ticketing . . . . .	25
4.1.3	Versioning . . . . .	26
4.1.4	Repository . . . . .	27
4.2	Processi di management . . . . .	28
4.2.1	Ruoli . . . . .	28
4.2.2	Comunicazioni . . . . .	29
4.2.3	Riunioni . . . . .	30
4.3	Meccanismi di controllo e rendicontazione . . . . .	31
4.3.1	Meccanismi di controllo . . . . .	31
4.3.2	Andamento delle attività . . . . .	31
4.3.3	Controllo metriche di progetto . . . . .	31
	<b>Elenco delle figure</b>	<b>33</b>
	<b>Elenco delle tabelle</b>	<b>34</b>

## Diario delle modifiche

Versione	Data	Autore	Descrizione
2.0.4	2016-05-13	<i>Progettista</i> Bonato Paolo	Modificata la sezione Qualifica: Aggiunte le procedure di controllo di qualità per i documenti, incrementati i concetti di verifica e validazione e le procedure di controllo sui processi
2.0.3	2016-05-06	<i>Progettista</i> Bonato Paolo	Inserite in processi di supporto le sezioni: Accertamento della qualità, Gestione delle versioni, Qualifica, Strumenti e metodi per l'applicazione delle metriche e Comunicazione e risoluzione delle anomalie, precedentemente contenute nel Piano di Qualifica v2.0.0. spostata la sezione meccanismi di controllo e rendicontazione sotto la sezione di processi organizzativi
2.0.2	2016-05-02	<i>Progettista</i> Nicoletti Luca	Inserita la sezione riguardante i meccanismi di controllo e rendicontazione precedentemente nel piano di progetto sotto la sezione di processi di supporto
2.0.1	2016-04-30	<i>Progettista</i> Maino Elia	Incrementate le sezioni relative all'automazione tramite piattaforma <i>SPitFire</i> (glossario, requisiti e tracciamento). Incremento norme di progettazione. Modifiche minori.
2.0.0	2016-04-11	<i>Responsabile</i> Maino Elia	Documento approvato
1.5.0	2016-04-11	<i>Verificatore</i> Paolo Bonato	Verificato il documento.
1.4.6	2016-04-10	<i>Progettista</i> Nicoletti Luca	Separata tabella del diario della modifiche in file esterno.
1.4.5	2016-04-10	<i>Progettista</i> Nicoletti Luca	Rivista la parte riguardante le repository. Cambiate le repository riguardanti il progetto e la sua documentazione. Cambiata la sezione riguardante il tracciamento dei requisiti. Inserito il link al tool web sviluppato dal gruppo.
1.4.4	2016-04-07	<i>Progettista</i> Nicoletti Luca	Cambiato la norma riguardante il software utilizzato per il disegno di casi d'uso, diagrammi di package, sequenza e attività.
1.4.3	2016-03-21	<i>Analista</i> Tommasin Davide	Controllo e sostituzione di termini all'interno della sezione di introduzione del documento
1.4.2	2016-03-11	<i>Progettista</i> Nicoletti Luca	Inserita in Sezione 3.1.3 la norma riguardante i nomi in prima pagina di ogni documento. Inserito in Sezione 3.1.14 una norma riguardante il controllo ortografico prima di sottomettere i documenti all'attività di verifica. Spostata vecchia sezione 2.1.1 in Sezione 3.1 Tecniche di analisi e classificazione dei requisiti.
1.4.1	2016-03-01	<i>Responsabile</i> Nicoletti Luca	Modifiche apportate a tutti gli errori riscontrati dall'analisi effettuata in data odierna. Inserita norma per nuovi comandi per scrivere riferimenti a dei documenti o a dei termini di glossario in sezione 2.2.2.

Versione	Data	Autore	Descrizione
1.4.0	2016-02-23	<i>Responsabile</i> Nicoletti Luca	Prime modifiche post revisione RR: riviste le norme riguardanti il versionamento dei documenti in sezione 3.1.13; modificata l'intera tabella del diario delle modifiche; inserita norma riguardante i termini presenti nel glossario per ogni documento.
1.3.0	2016-01-18	<i>Responsabile</i> Padovan Tommasin	Documento approvato, pronto alla consegna
1.2.0	2016-01-18	<i>Verificatore</i> Tommasin Davide	Verifica effettuata, segnalati errori e dimenticanze
1.1.2	2016-01-17	<i>Amministratore</i> Nicoletti Luca	Modifica margini, inserita norma sui margini
1.1.1	2016-01-11	<i>Amministratori</i> Maino Elia, Nicoletti Luca	Stesura norme riguardanti commit e politiche di ticketing, correzione errori
1.1.0	2016-01-10	<i>Verificatori</i> Biggieri Mattia, Tommasin Davide	Verifica del documento
1.0.5	2016-01-09	<i>Amministratore</i> Nicoletti Luca	Stesura norme riguardanti documentazione di pianificazione, correzione errori
1.0.4	2016-01-09	<i>Amministratore</i> Maino Elia	Stesura norme riguardanti risoluzione problemi, versioning, comunicazioni e casi d'uso
1.0.3	2016-01-09	<i>Amministratore</i> Nicoletti Luca	Stesura norme riguardanti Ticketing, Repository, Ruoli, Riunioni
1.0.2	2016-01-08	<i>Amministratore</i> Maino Elia	Stesura norme Documentazione
1.0.1	2016-01-08	<i>Amministratore</i> Nicoletti Luca	Stesura norme riguardanti Studio di fattibilità e requisiti, codifica $\text{\LaTeX}$ e <i>Scala</i>
1.0.0	2016-01-08	<i>Amministratori</i> Maino Elia, Nicoletti Luca	Creazione scheletro documento e struttura organizzativa: inserite le sezioni: Introduzione, Processi primari, Processi di supporto, Processi organizzativi Stesura Introduzione

Tabella 1: Diario delle modifiche

# 1 Introduzione

## 1.1 Scopo del documento

Nel seguente documento sono definite le norme che l'intero gruppo SWEneyThreads si impegna a rispettare durante lo svolgimento del progetto Actorbase.

Ogni membro è tenuto a leggere il documento e a rispettare le norme al fine di dare maggiore uniformità allo svolgimento dei processi, migliorandone l'efficacia, riducendo il numero di errori e i tempi di sviluppo.

Poiché il gruppo ha deciso di basarsi sulla struttura a processi *ISO/IEC 12207* la struttura di questo documento ne rispecchia l'organizzazione. In particolare la suddivisione in processi primari, di supporto e organizzativi.

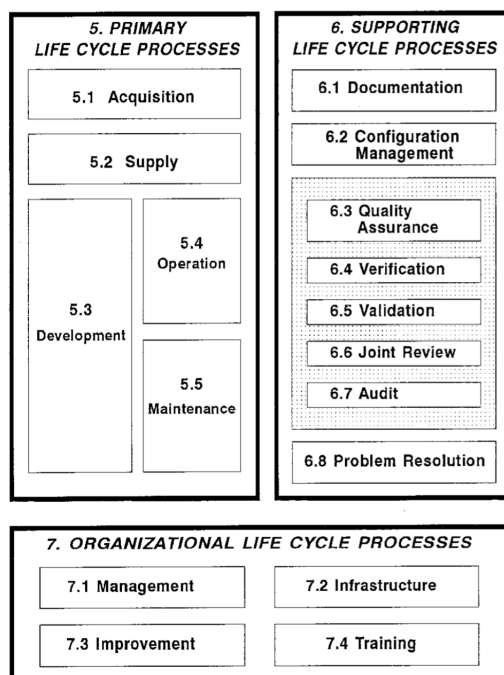


Figura 1: Processi ISO/IEC 12207

## 1.2 Scopo del prodotto

Lo scopo del progetto è la realizzazione di un DataBase NoSQL key-value basato sul modello ad Attori con l'obiettivo di fornire una tecnologia adatta allo sviluppo di moderne applicazioni che richiedono brevissimi tempi di risposta e che elaborano enormi quantità di dati. Lo sviluppo porterà al rilascio del software sotto licenza MIT.

## 1.3 Glossario

Al fine di evitare ambiguità di linguaggio e di massimizzare la comprensione dei documenti, il gruppo ha steso un documento interno che è il *Glossario v2.0.0*. In esso saranno definiti, in modo chiaro e conciso i termini che possono causare ambiguità o incomprensione del testo. In ogni documento, ogni occorrenza di un termine presente nel glossario, verrà contrassegnata con una *G* a pedice come la seguente:

*G.*

L'inserimento della *G* a pedice è automatizzato grazie ad una funzionalità della piattaforma *SPitFire*, insieme di tool al supporto della documentazione e del tracciamento sviluppato dal gruppo. Essa permette di effettuare l'upload di un file .tex a cui si vogliono aggiungere le *G* e di ottenere un file con le aggiunte corrette in corrispondenza dei termini del glossario. *SPitFire* permette di salvare la lista dei termini di

glossario semplicemente effettuando l'upload del file .tex del Glossario, ricava in automatico la lista delle parole e le utilizza per aggiungere le  $G$  correttamente.

## 1.4 Riferimenti

### 1.4.1 Informativi

- Specifiche UTF-8:  
[http://unicode.org/faq/utf\\_bom.html](http://unicode.org/faq/utf_bom.html)
- Licenza MIT:  
<https://opensource.org/licenses/MIT>
- Scala Programming Language:  
<http://www.scala-lang.org/>
- ISO/IEC 12207:  
[http://www.iso.org/iso/catalogue\\_detail?csnumber=43447](http://www.iso.org/iso/catalogue_detail?csnumber=43447)
- ISO 8601:2004:  
<http://www.iso.org/iso/home/standards/iso8601.htm>
- L<sup>A</sup>T<sub>E</sub>X:  
<https://www.latex-project.org>
- UML:  
<http://www.uml.org>
- Astah Professional:  
<http://astah.net/>
- Telegram:  
<https://telegram.org>
- Google Drive:  
[https://www.google.com/intl/it\\_it/drive/](https://www.google.com/intl/it_it/drive/)
- Google Hangouts:  
<https://hangouts.google.com>
- TeamWork:  
<https://www.teamwork.com/>
- IntelliJ:  
<https://www.jetbrains.com/idea/>
- ProjectLibre:  
<http://www.projectlibre.org/>
- GitHub:  
<https://github.com/>
- Texmaker:  
<http://www.xm1math.net/texmaker/>
- altervista:  
<http://it.altervista.org/>
- SPitFire:  
<http://sweeneytreadaas.altervista.org/>
- SweeneyThreads:  
<http://sweeneythreads.github.io/>
- Piano di progetto:  
*Piano di Progetto v2.0.0*
- Piano di qualifica:  
*Piano di Qualifica v2.0.0*

#### 1.4.2 Normativi

- Capitolato d'appalto Actorbase (C1):  
<http://www.math.unipd.it/~tullio/IS-1/2015/Progetto/C1p.pdf>



## 2 Processi primari

Sono state definite delle norme relative ai processi primari che maggiormente riguardano le attività svolte dal gruppo: fornitura e sviluppo.

### 2.1 Fornitura

#### 2.1.1 Studio di fattibilità

Lo *Studio di fattibilità* del progetto deve essere steso dai membri che ricoprono il ruolo di *Analisti* sulla base delle prime riunioni effettuate, decise dal *Responsabile di progetto*, e delle preferenze espresse da ogni singolo membro del gruppo. In seguito il documento verrà analizzato e valutato da altri membri del gruppo. Lo *Studio di fattibilità* deve contenere:

- **Dominio:** conoscenza delle tecnologie richieste e del dominio applicativo;
- **Rapporto costo/benefici:** eventuali prodotti simili già presenti sul mercato, competitori, costo della realizzazione del prodotto e quantità di requisiti obbligatori;
- **Individuazione dei rischi:** evidenziare lacune tecniche e di conoscenza del dominio dei membri del gruppo, comprensione dei punti critici, difficoltà nel determinare i requisiti obbligatori e opzionali e nella loro classificazione.

Tale valutazione deve essere svolta per tutti i capitoli presenti, eventualmente con un maggior livello di dettaglio per quanto riguarda il *capitolo C1*.

### 2.2 Sviluppo

#### 2.2.1 Casi d'uso

Come detto in precedenza alcuni requisiti possono essere ricavati dai Casi d'uso<sub>G</sub>, ad essi si può fare riferimento anche con la dicitura *use case* o con l'acronimo *UC* nel caso fosse necessario utilizzarli in tabelle o diagrammi.

I Casi d'uso<sub>G</sub> vanno identificati dagli *Analisti*, attraverso una procedura che va dal generale al particolare.

Un *caso d'uso* richiede la definizione dei seguenti campi:

- Codice gerarchico
- Nome sintetico
- Attori
  - Principali
  - Secondari
- Pre-condizione
- Post-condizione
- Flusso degli eventi relativi allo scenario principale
- Eventuali scenari alternativi
- Lista di requisiti dedotti

A tale insieme di informazioni va associato un diagramma di caso d'uso in UML. Per la scrittura dei Casi d'uso<sub>G</sub> in UML il gruppo ha deciso di utilizzare il software *Astah Professional* alla versione 7.0. La scelta è caduta su tale programma poiché è disponibile per tutti i principali sistemi operativi desktop e permette di ottenere una licenza gratuita per studenti.

È degli *Analisti* il compito di inserire tutte le informazioni relative ai casi d'uso e i loro diagrammi nel documento di *Analisi dei requisiti*.

Il codice gerarchico del *caso d'uso* ha la forma seguente:

UC[codice univoco del padre].[codice progressivo]

Dove il codice progressivo può definire diversi livelli di gerarchia separati da un punto.

I Casi d'uso<sub>G</sub> saranno espressi in forma tabellare nel documento di *Analisi dei requisiti* nel seguente modo, in più, ogni caso d'uso sarà accompagnato da un diagramma UML che ne descrive il flusso di eventi.

Di seguito si riporta un esempio di caso d'uso comprensivo di diagramma UML e di descrizione tabellare.

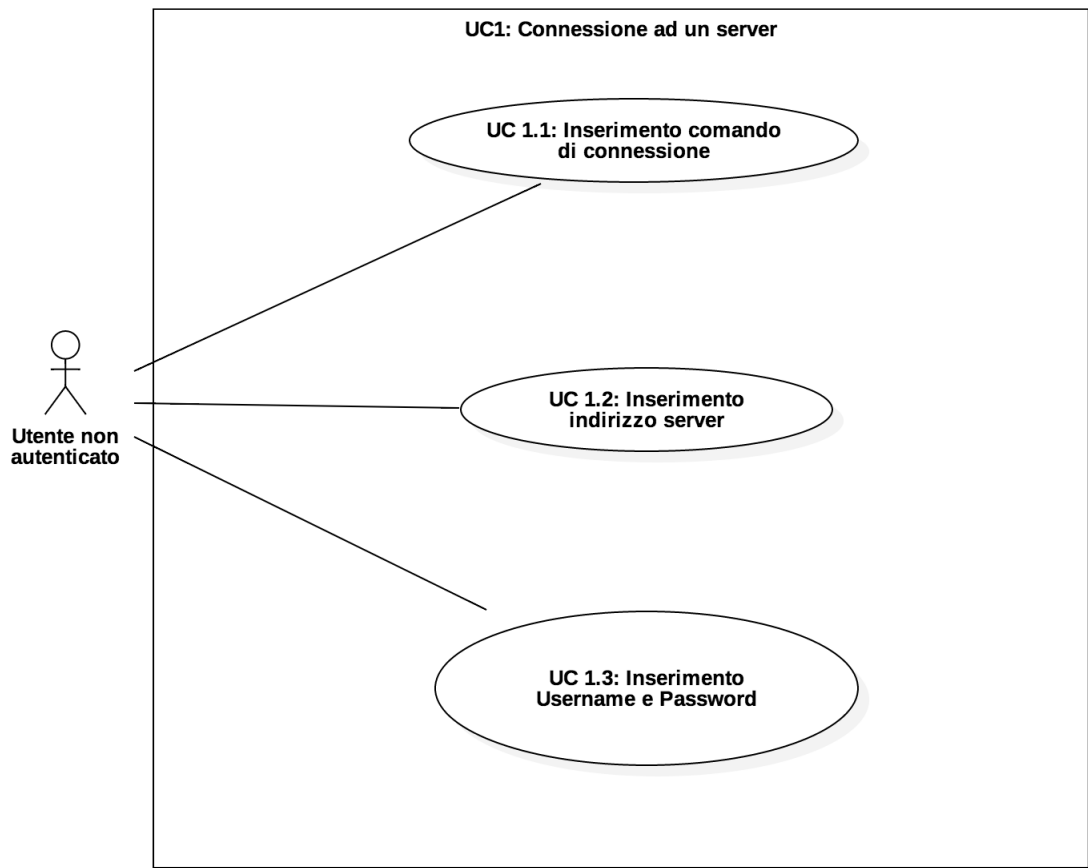


Figura 2: Diagramma di UC1: Connessione al server

### Descrizione

L'utente ha appena avviato l'interfaccia CLI dell'applicativo e intende connettersi ad un server contenente i database utilizzando il comando di connessione (UC 1.1). L'utente deve inserire l'indirizzo del server (UC 1.2) e successivamente l'username e la password necessari per l'accesso (UC 1.3), nel caso in cui il tentativo di connessione dovesse fallire, l'utente riceve un messaggio di errore informativo.

<b>Codice gerarchico</b>	UC1
<b>Nome sintetico</b>	Connessione al server
<b>Attore principale</b>	Utente non autenticato
<b>Attori secondari</b>	Nessuno
<b>Pre-condizione</b>	L'utente ha avviato l'interfaccia CLI, non è autenticato e intende connettersi ad un server
<b>Post-condizione</b>	L'utente risulta connesso al server
<b>Flusso eventi</b>	<ol style="list-style-type: none"> <li>1. L'utente inserisce il comando di connessione (UC 1.1)</li> <li>2. L'utente inserisce l'indirizzo del server (UC 1.2)</li> <li>3. L'utente inserisce username e password (UC 1.3) e preme invio</li> </ol>
<b>Scenari alternativi</b>	Nessuno
<b>Lista requisiti</b>	...

Tabella 2: Caso d'uso UC 1 - Connessione al server

### 2.2.2 Tecniche di analisi e classificazione requisiti

Sempre compito degli *Analisti*, sarà quello di stilare l'*Analisi dei requisiti*. Essi potranno ricavarli da eventuali Casi d'uso<sub>G</sub> emersi da Brainstorming o riunioni con il committente.

I requisiti saranno elencati secondo un ordine. Ogni requisito seguirà la seguente codifica:

R[Codice][Importanza][Tipo]

#### Codice

Un codice univoco ed espresso in modo gerarchico;

#### Importanza

Può assumere i seguenti valori:

- **N:** Necessario;
- **D:** Desiderabile;
- **O:** Opzionale.

#### Tipo

Può assumere i seguenti valori:

- **F:** funzionale;
- **Q:** di qualità;
- **P:** prestazionale;
- **V:** vincolo.

I requisiti saranno inseriti in una tabella, che includerà anche un nome per ogni attributo, una fonte e una breve descrizione. La forma tabellare di un requisito risulta quindi essere:

Codice	Nome	Fonte	Descrizione
R[1.4.5][N][P]	Requisito di qualità	Capitolato	Garantire ...

### 2.2.3 Gestione dei requisiti

Per automatizzare il più possibile la gestione dei requisiti ed il relativo tracciamento il gruppo ha sviluppato una piattaforma accessibile via web denominata *SPitFire*.

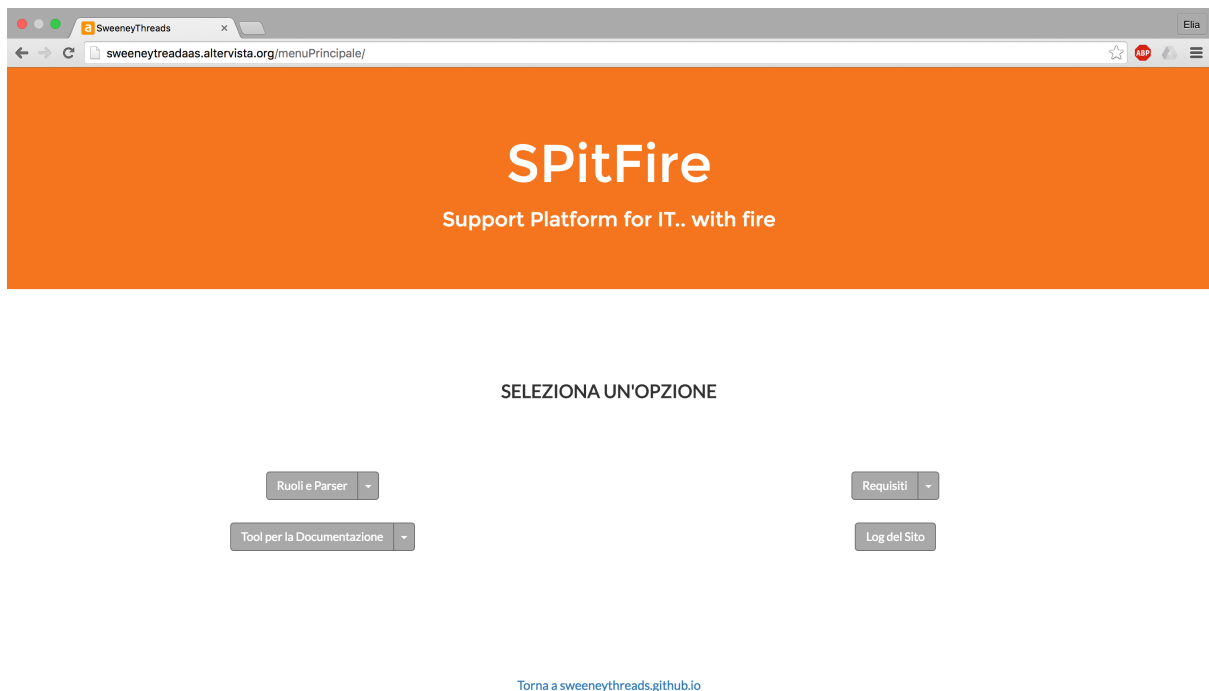


Figura 3: Pagina principale di SPitFire

*SPitFire* si basa sul salvataggio dei dati in un database mySQL e permette di svolgere le seguenti operazioni sui requisiti:

- Inserire un requisito tramite interfaccia web
- Visualizzare i requisiti presenti, navigando dai requisiti di livello superiore a quelli di livello inferiore
- Segnare il completamento di un requisito
- Importare i requisiti da una tabella .tex
- Esportare i requisiti in una tabella .tex



Figura 4: Opzioni di gestione dei requisiti offerte da *SPitFire*

In particolare l'esportazione da database a file .tex e l'importazione da file .tex a database permettono di ridurre notevolmente il carico di lavoro degli analisti per quanto riguarda la semplice stesura della documentazione. Le tabelle dei requisiti presenti nel documento di *Analisi dei requisiti* sono infatti generate in automatico.

## 2.2.4 Tracciamento dei requisiti

Il tracciamento dei requisiti è anch'esso automatizzato grazie alla piattaforma *SPitFire*.

Il tool individua automaticamente le dipendenze di tipo padre-figlio e genera in tal modo un albero dei requisiti navigabile semplicemente. Sono stati definiti dei trigger SQL per cui non risulta possibile segnare come completato un requisito che presenta requisiti figli incompleti. Inoltre il completamento di tutti i requisiti figli rende automatico il completamento del padre.

Il tracciamento tra fonti e requisiti è automatico: *SPitFire* permette di generare le tabelle di tracciamento in formato .tex basandosi sui dati presenti nel database.

### FONTI - REQUISITI

```
Capitolato & R[1] [N] [F] \newline
R[1.1] [N] [F] \newline
R[1.1.3] [N] [F] \newline
R[1.2] [N] [F] \newline
R[1.4.4.2.1] [N] [F] \newline
R[1.4.4.2.2] [N] [F] \newline
R[1.4.4.2.3] [N] [F] \newline
R[1.4.4.2.4] [N] [F] \newline
R[1.4.4.2.5] [O] [F] \newline
R[1.4.4.2.6] [O] [F] \newline
R[1.4.5.2.1] [N] [F] \newline
R[1.4.5.2.2] [N] [F] \newline
R[1.4.5.2.3] [N] [F] \newline
R[1.4.5.2.4] [N] [F] \newline
R[1.4.5.2.5] [O] [F] \newline
R[1.4.5.2.6] [O] [F] \newline
R[1.4.6.3.1] [D] [F] \newline
R[1.4.6.3.2] [D] [F] \newline
R[1.4.6.3.3] [D] [F] \newline
R[1.4.6.3.4] [D] [F] \newline
R[1.4.6.3.5] [O] [F] \newline
```

Figura 5: Tabelle di tracciamento generate da *SPitFire*

## 2.2.5 Gestione cambiamento requisiti

Per quanto riguarda il cambiamento dei requisiti, nel Database verrà tenuta una tabella di backup dei "vecchi" requisiti, nella loro forma di dichiarazione, con un puntatore al "nuovo" requisito, che invece avrà le specifiche aggiornate. È compito del *Responsabile di progetto* mantenere aggiornata la tabella dei requisiti, copiando prima il requisito che necessita di cambiamento nella tabella dei vecchi requisiti, e poi aggiornando il requisito stesso.

Inoltre *SPitFire* individua automaticamente eventuali incongruenze tra i file L<sup>A</sup>T<sub>E</sub>X presenti e le informazioni contenute nel database, in tal modo è semplice tenere traccia dei requisiti non aggiornati.

## 2.2.6 Progettazione architetturale

La progettazione architetturale prevede la definizione delle componenti architetture. La descrizione di ogni componente package deve avvenire sia in forma testuale, che in forma di diagrammi UML. La descrizione di classi e interfacce può invece avvenire semplicemente a livello testuale.

Nello specifico i diagrammi UML che il gruppo dovrà utilizzare in fase di progettazione sono i seguenti:

- Diagrammi dei package
- Diagrammi delle classi
- Diagrammi di attività
- Diagrammi di sequenza

## 2.2.7 Definizione testuale delle componenti architetture

Per garantire uniformità nel documento di *Specifica Tecnica*, i membri del gruppo dovranno rispettare una struttura definita per quanto riguarda la descrizione testuale di componenti architetture, siano esse Package, classi o interfacce.

**Componente package** La descrizione deve essere composta da al più i seguenti elementi:

- Descrizione
- Package figli
- Classi
- Interfacce
- Relazioni con altre componenti

**Componente classe/interfaccia** La descrizione deve essere composta da al più i seguenti elementi:

- Descrizione
- Utilizzo
- Classi estese
- Interfacce implementate
- Classi figlie
- Relazioni con altre classi

### 2.2.8 Diagrammi delle componenti architetturali

In linea con quanto definito per la stesura dei diagrammi dei casi d'uso, i diagrammi delle componenti dovranno essere realizzati utilizzando il software *Astah Professional* alla versione 7.0 .

**Diagrammi dei package** La distinzione tra package si basa principalmente sul colore, per tale motivo ogni qual volta si utilizzi una combinazione di colori differente, essa deve essere accompagnata da una legenda descrittiva.

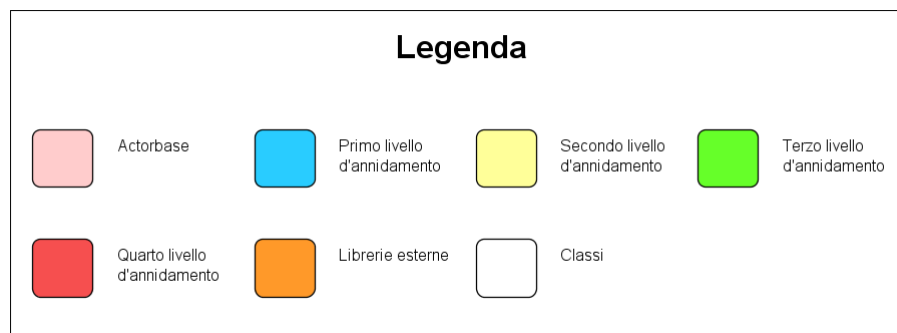


Figura 6: Esempio di legenda per un diagramma dei package

In particolare è consigliato utilizzare colori diversi per distinguere:

- Componenti di livello diverso
- Componenti interni da librerie esterne e framework

**Diagrammi delle classi** Poiché i diagrammi delle classi raggiungono facilmente un elevato livello di complessità, i membri del gruppo devono assicurarsi di inserire nei documenti diagrammi che siano leggibili. Nel caso in cui un diagramma risultasse troppo dettagliato la sua immagine andrà inserita in repository ma non nel documento, in esso sarà necessario dividere il diagramma in più parti, ed inserire immagini distinte e leggibili per ognuna di esse.

**Diagrammi di attività e di sequenza** Valgono le stesse norme di leggibilità definite per i diagrammi delle classi, inoltre ogni diagramma di attività o sequenza dovrà essere accompagnato da una descrizione testuale.

### 2.2.9 Design Pattern

I membri del gruppo sono invitati ad utilizzare design pattern noti dove possibile in fase di progettazione. L'utilizzo di essi permetterà di avere un'architettura più comprensibile ed estendibile sia dal punto di vista del codice che dal punto di vista della documentazione che lo accompagna.

Ogni volta che una parte dell'architettura del sistema si basa su un design pattern sarà necessario fornire una descrizione dello stesso in un'apposita sezione del documento di *Specifica Tecnica*.

### 2.2.10 Codifica in L<sup>A</sup>T<sub>E</sub>X

Regole riguardanti la stesura di codice L<sup>A</sup>T<sub>E</sub>X:

- Ogni file deve iniziare con 3 righe di commento come quelle riportate in seguito:

```
%Document-Author: Cognome Nome + Cognome Nome
%Document-Date: aaaa/mm/gg
%Document-Description: descrizione
```

Figura 7: L<sup>A</sup>T<sub>E</sub>X - Commenti ad inizio file

- Ogni file deve contenere nella prima parte tutti gli `\usepackage{}` necessari
- I commenti andranno inseriti in una riga vuota, eventualmente prima della riga di codice a cui fanno riferimento
- I commenti su più righe useranno il comando `\begin{comment}` - `\end{comment}`
- Tra ogni `\begin{PART}` e `\end{PART}` tutto il testo e il codice andrà indentato:

```
\begin{PART1}
> testo
> \begin{PART2}
> > testo
> > testo
> \end{PART2}
\end{PART1}
```

Figura 8: L<sup>A</sup>T<sub>E</sub>X - Indentazione 1

- Per quanto riguarda il comando personalizzato `\newpage` `\section{}` o altre sezione `\subsection{}`, `\subsubsection{}` verranno comunque indentate le parti innestate al loro interno come segue:

```

\mychapter{n}{Titolo}
> Testo
> \section{sezione}
> > testo
> > \subsection{sottosezione}
> \section{sezione2}
> > testo
> > \subsection{sottosezione2}
> > > \subsubsection{non numerata}
> > > > testo
> > > \subsubsection{non numerata2}
> > > > testo
> > > > testo
> \section{sezione3}

```

Figura 9: L<sup>A</sup>T<sub>E</sub>X - Indentazione 2

- Verrà utilizzato T1 come encoding del font:  
`\usepackage[T1]{fontenc}`
- Verrà utilizzato utf8 come encoding dell'input:  
`\usepackage[utf8]{inputenc}`
- Verrà utilizzato english, italian come parametro per babel:  
`\usepackage[english, italian]{babel}` in modo da usare inglese e italiano nello stesso documento tenendo italiano come lingua principale
- Prima di ogni immagine, verrà inserito un commento su una riga, come definito sopra, per facilitarne l'individuazione:

```

%immagine
\begin{figure}[H]
> \centering
> \includegraphics[scale=x.x]{nome_file.estensione}
> \caption{Titolo immagine}
\end{figure}

```

Figura 10: L<sup>A</sup>T<sub>E</sub>X - Commento prima di ogni immagine

- A fine documento, come commento su più righe, andrà inserita la documentazione e la descrizione (anche breve) del file

### 2.2.11 Codifica in *Scala*

Tutte le regole di indentazione, assegnazione dei nomi, scrittura delle parentesi, nomina file, e documentazione sono quelle definite dalla documentazione ufficiale di *Scala*: <http://docs.scala-lang.org/style/>



Figura 11: Scala Style Guide

### 2.2.12 Integrazione del software

Per l'integrazione continua delle componenti del sistema ed il relativo testing il gruppo ha deciso di utilizzare *Travis CI*. Sarà compito dei *progettisti* definire il sistema di build.





Figura 12: Logo di Travis CI

## 3 Processi di supporto

### 3.1 Documentazione

In questo capitolo si descrivono le convenzioni definite e adottate dal gruppo riguardanti le modalità di redazione, verifica e approvazione dei documenti.

Tutti i documenti formali prodotti da SWEeneyThreads sono scritti utilizzando il linguaggio  $\text{\LaTeX}$ , compilati e forniti in formato PDF (per quanto riguarda le versioni digitali). Per la stesura dei documenti il gruppo utilizzerà il software *Termaker*.

#### 3.1.1 Template

Al fine di rendere più rapida e meno incline a differenziazioni la stesura dei diversi documenti è stato prodotto un template  $\text{\LaTeX}$ , reperibile nel repository in `Actorbase/LaTeX/Templates`.

#### 3.1.2 Struttura documenti

La struttura dei documenti presenta una suddivisione in sezioni, sottosezioni e ulteriori sotto-sottosezioni. Tutti le sezioni, sottosezioni e sotto-sottosezioni sono state create usando i comandi standard  $\text{\LaTeX}$  `\section{}`, `\subsection{}` e `\subsubsection{}`.

La numerazione delle sezioni è utilizzata fino al terzo livello di profondità (x.y.z), dal quarto livello in poi le sottosezioni non presentano numerazione. Tale scelta è stata presa al fine di rendere più leggibile l'indice.

Di seguito viene fornita una descrizione più dettagliata di alcuni elementi di un documento:

#### 3.1.3 Prima pagina

La prima pagina di un documento presenta gli elementi seguenti:

- Nome del gruppo
- Nome del progetto
- Sottotitolo del progetto
- Titolo del documento
- Cognome e nome dei redattori del documento
- Cognome e nome di chi approva il progetto in qualità di responsabile
- Cognome e nome dei verificatori del documento
- Logo del gruppo
- Numero di versione del documento
- Data di rilascio del documento

La prima pagina è parte del template disponibile nel repository.

I cognomi e i nomi dei redattori, del responsabile e dei verificatori, verranno cambiati di versione in versione. Questo è possibile grazie al versionamento dei documenti, infatti, i nomi di chi lavora al documento, fanno riferimento solo alla versione in cui il documento è in un determinato momento. Questo comporta un cambiamento dei nomi dei responsabili di un documento di versione in versione. Il numero di nomi è quindi variabile, in una fase ci possono essere 2 persone per la scrittura del documento, in quella successiva 3, o solamente una.

### 3.1.4 Indice

In ogni documento sono presenti in ordine

- Un indice delle sezioni;
- Un indice delle tabelle;
- Un indice delle figure.

Tali indici sono generati automaticamente tramite appositi comandi  $\text{\LaTeX}$ , l'assenza di figure e/o tabelle nel documento comporta l'omissione del corrispondente indice.

Data la natura secondaria degli indici relativi alle tabelle e alle figure, si è deciso di posizionarli alla fine del documento. L'indice dei contenuti si trova invece subito dopo la pagina iniziale.

### 3.1.5 Diario delle modifiche

Ogni documento deve contenere una sezione denominata "Diario delle modifiche" in cui annotare tutte le attività svolte sul documento.

Lo schema della tabella è il seguente:

- **Versione:** numero di versione del documento dopo le modifiche;
- **Data:** data in cui sono state apportate le modifiche;
- **Autore:** ruolo, cognome e nome dell'autore che apportato le modifiche (gli autori possono essere più di uno);
- **Descrizione:** descrizione delle modifiche apportate al documento.

La compilazione della tabella è un'attività obbligatoria nel caso di modifiche rilevanti al documento. Un documento non può cambiare di versione senza che tale cambiamento venga annotato nella tabella, nel caso di modifiche minori che non cambiano di molto il contenuto (es. correggere un accento) la tabella può rimanere invariata.

Versione	Data	Autore	Descrizione
----------	------	--------	-------------

Tabella 3: Schema del diario delle modifiche

Il Diario delle modifiche non è incluso nella numerazione delle sezioni, si trova dopo l'indice e prima di qualsiasi capitolo numerato

### 3.1.6 Formattazione generale delle pagine

La formattazione generale di una pagina prevede la diminuzione dei margini destri e sinistri, ma non prevede altre modifiche importanti e si basa, per tutte le altre regole, sulla formattazione standard di  $\text{\LaTeX}$  usata per i documenti di classe "Report".

Per effettuare la modifica viene usato i seguenti comandi:

```
\usepackage{geometry}  
\geometry{margin=1in}.
```

### 3.1.7 Norme tipografiche

Questa sezione contiene norme tipografiche e ortografiche adottate dal gruppo al fine di garantire uno stile uniforme e una semantica coerente per tutti i documenti.

### 3.1.8 Stile del testo

Il font utilizzato in tutti i documenti formali scritti dal gruppo sarà il *Computer Modern*, ovvero quello standard utilizzato da L<sup>A</sup>T<sub>E</sub>X.

- **Corsivo:** il corsivo va utilizzato nei casi seguenti:

- Citazioni;
- Nomi particolari;
- Documenti;
- Riferimenti;

A seconda della semantica del testo si utilizzano i comandi L<sup>A</sup>T<sub>E</sub>X `\emph{}` e `\textit{}`.

- **Grassetto:** il grassetto va utilizzato nei casi seguenti:
  - Elenchi puntati: evidenzia il concetto sviluppato nella continuazione del punto.
- **Maiuscolo:** una parola completamente in maiuscolo deve indicare un acronimo o una sigla.
- **L<sup>A</sup>T<sub>E</sub>X:** ogni riferimento al linguaggio L<sup>A</sup>T<sub>E</sub>X va scritto utilizzando il comando `\LaTeX`.

### 3.1.9 Formati

- **Percorsi:**
  - Indirizzi email : comando L<sup>A</sup>T<sub>E</sub>X `\href{mailto:nome@dominio}{nome@dominio}`;
  - Indirizzi web completi: comando L<sup>A</sup>T<sub>E</sub>X `\url`;
  - Indirizzi relativi: comando L<sup>A</sup>T<sub>E</sub>X `\verb`.
- **Date:** le date presenti nei documenti seguono lo standard ISO 8601:2004:

AAAA - MM - GG

Dove:

- AAAA rappresenta l'anno;
  - MM rappresenta il mese;
  - GG rappresenta il giorno.
- **Ruoli di progetto:** quando si fa riferimento ad un ruolo di progetto questo va scritto in corsivo e con la prima lettera maiuscola (es. *Responsabile*).
  - **Documenti:** i riferimenti vanno scritti in corsivo (es. *Analisi dei requisiti*).
  - **Nomi dei file:** i nomi dei file vanno scritti utilizzando il comando L<sup>A</sup>T<sub>E</sub>X `\verb` (es. `immagine.png`).
  - **Nomi propri:** I nomi propri seguono la forma "Cognome Nome".
  - **Nome del gruppo:** il nome del gruppo è SWEeneyThreads, la distinzione tra lettere maiuscole e minuscole va rispettata ogni volta che vi si fa riferimento.

### 3.1.10 Sigle

L'utilizzo di sigle e abbreviazioni per riferirsi a documenti va limitato il più possibile, tuttavia nel caso il loro uso fosse funzionale alla lettura (come nel caso di tabelle o diagrammi) il loro uso è consentito:

- **SdF:** Studio di Fattibilità;
- **AdR:** Analisi dei Requisiti;
- **GL:** Glossario;
- **NdP:** Norme di Progetto;

- **PdQ:** Piano di Qualifica;
- **PdP:** Piano di Progetto;
- **ST:** Specifica Tecnica;
- **RR:** Revisione dei Requisiti;
- **RP:** Revisione di Progettazione;
- **RQ:** Revisione di Qualifica;
- **RA:** Revisione di Accettazione.

### 3.1.11 Componenti grafiche

Le componenti grafiche previste all'interno dei documenti sono immagini e tabelle. Ogni occorrenza di un elemento grafico è accompagnata da una didascalia indicizzata, in modo da poterla associare alla sezione relativa del documento. **Tabelle** Le tabelle sono definite utilizzando un template in  $\text{\LaTeX}$  realizzato dal gruppo e disponibile nel repository all'indirizzo `Actorbase/LaTeX/Templates` **Immagini** Il formato scelto per le immagini è Portable Network Graphics (PNG).

Le immagini vanno sempre inserite utilizzando la seguente sequenza di comandi  $\text{\LaTeX}$ :

```
\begin{figure}[H]
\centering
\includegraphics[scale=0-1]{Immagini/nome.png}
\caption{Titolo - didascalia}
\end{figure}
```

### 3.1.12 Classificazione documenti

I documenti prodotti dal gruppo si dividono in formali e informali. **Documenti formali** Quando un documento riceve l'approvazione del *Responsabile* viene definito formale e risulta idoneo al rilascio all'esterno del gruppo.

Per risultare approvato un documento deve aver completato con successo il percorso di verifica e validazione descritto nel *Piano di Qualifica*. **Documenti informali** Un documento rimane informale finché non viene approvato dal *Responsabile*, durante tale fase il suo uso è da considerarsi esclusivamente interno al gruppo.

Alcuni documenti prodotti dal gruppo possono rimanere informali per l'intera durata del loro ciclo di vita.

### 3.1.13 Versionamento documenti

I documenti prodotti dal gruppo devono essere sempre identificati da un numero di versione del tipo:

X.Y.Z

Dove:

- X: è il numero principale di versione, viene incrementato ad ogni uscita formale del documento;
- Y: viene incrementato quando il documento entra in una fase successiva del suo ciclo di vita;
- Z: viene incrementato quando si apportano modifiche minori al documento.

All'interno di un documento quando si intende fare riferimento ad una specifica versione di un altro documento la notazione da utilizzare è:

*Nome Documento vX.Y.Z.*

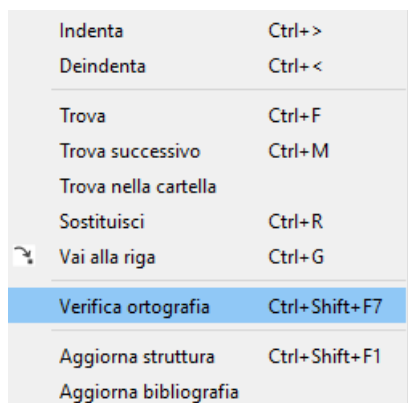
Mentre per fare riferimento ad un file vero e proprio:

NomeDocumento\_vX.Y.Z.estensione

### 3.1.14 Ciclo di vita dei documenti

Ogni documento prodotto dal gruppo rispetta il seguente ciclo di vita:

- **Lavorazione/Modifica:** il documento entra in questa fase al momento della sua creazione e vi rimane per tutto il tempo in cui il suo contenuto viene modificato. Prima di terminare la sua fase di modifica ed essere messo a disposizione dei verificatori, su ogni documento deve essere effettuato il controllo ortografico messo a disposizione dal software *TexMaKer*:



Indenta	Ctrl+>
Deindenta	Ctrl+<
Trova	Ctrl+F
Trova successivo	Ctrl+M
Trova nella cartella	
Sostituisci	Ctrl+R
Vai alla riga	Ctrl+G
Verifica ortografia	Ctrl+Shift+F7
Aggiorna struttura	Ctrl+Shift+F1
Aggiorna bibliografia	

Figura 13: Controllo ortografico - strumento di TexMaKer

- **Verifica:** quando termina la fase di modifica, il documento passa nelle mani dei *Verificatori* che lo analizzano al fine di individuare eventuali errori o incongruenze sintattiche e semantiche;
- **Approvazione:** dopo essere stato verificato il documento deve essere approvato dal *Responsabile*. Se il documento ottiene l'approvazione diventa ufficiale e raggiunge lo stato finale del suo ciclo di vita per quanto riguarda la corrente versione.

Ogni documento prodotto può attraversare più volte ogni fase del suo ciclo di vita, allo stesso modo può non attraversarle tutte. Quando si inizia una revisione formale su un documento già approvato questo ricomincia il ciclo da capo con un numero di versione incrementato.

## 3.2 Accertamento della qualità

### 3.2.1 Analisi dei processi

I processi compiuti durante una fase verranno analizzati secondo il seguente protocollo:

- **Controllo delle metriche:** Alla conclusione di ogni fase del progetto si calcolano gli indici definiti nella sezione 2.2.1 confrontando le macro-attività preventivate nel *Piano di progetto* con i dati effettivi riscontrati dal sistema di ticketing.
- **Analisi PDCA:** Secondo il ciclo PDCA una fase di progetto ha inizio con la pianificazione di come possono essere migliorati i processi. Durante la fase vengono attuati i cambiamenti prefissati e alla fine viene effettuato un controllo. Se un processo risulta essere migliore viene adottato in modo definitivo, se invece risulta inalterato o addirittura peggiore i cambiamenti vengono scartati.

### 3.2.2 Analisi dei documenti

Ogni documento redatto è verificato mediante il seguente protocollo:

- **Controllo sintattico:** Il testo deve venire sottoposto a controllo dell'ortografia con il tool fornito dall'ambiente di sviluppo  $\text{\LaTeX}$  utilizzato. Alcuni errori non possono essere comunque rilevati da meccanismi automatici, quindi al fine di ottenere correttezza sintattica e semantica i Verificatori effettueranno un walkthrough al fine di ricercare errori sfuggiti al correttore ortografico.
- **Controllo semantico:** I Verificatori sono tenuti a leggere il documento, controllando che le frasi lette abbiano senso compiuto e che siano pertinenti all'argomento trattato nella sezione. Errori di questo tipo possono essere dovuti a sviste o errori di copia nella stesura.

- **Rispetto delle Norme di progetto:** I Verificatori sono tenuti a verificare che il documento rispetti tutte le norme tipografiche e di struttura del documento riportate nelle *Norme di progetto*. Porzioni di questa verifica sono automatizzabili, i Verificatori dovranno quindi usare tool ove possibile.
- **Inspection secondo checklist:** I Verificatori dovranno scorrere la lista di controllo e verificare che non sia presenti gli errori comuni lì riportati.
- **Verifica Glossario:** I Verificatori dovranno controllare che tutti i termini contenuti nel Glossario siano indicati all'interno dei documenti con la G a pedice.
- **Calcolo delle metriche:** I Verificatori dovranno calcolare le seguenti metriche per ogni documento:
  - **Gulpease:** I verificatori calcoleranno l'indice Gulpease del documento tramite strumenti automatici forniti su [http://sweeneytreedaas.altervista.org/menuPrincipale/documentation\\_tools/](http://sweeneytreedaas.altervista.org/menuPrincipale/documentation_tools/), in caso i risultati non siano soddisfacenti presenterà al redattore le sezioni peggiori;
  - **Numero figure e tabelle su numero pagine:** Tramite gli elenchi delle figure e delle tabelle sarà calcolato il rapporto tra la somma delle stesse e il numero di pagine, e se risulterà scadente il verificatore lo segnalerà al redattore del documento.
  - **Media parole per section:** La metrica sarà calcolata tramite strumenti automatici e in caso di valori non accettabili le sezioni peggiori saranno comunicate al redattore.
- **Miglioramento:** Se nello svolgimento di uno qualunque dei punti precedenti un Verificatore notasse nuove possibilità di automatizzazione dovrà segnalarle e si dovrà cercare o costruire uno strumento per rendere effettiva tale automatizzazione. Inoltre durante le esecuzioni dei walkthrough, i verificatori sono tenuti ad annotare gli errori più frequenti o potenzialmente dannosi rilevati. Tali errori andranno poi ad aggiornare la lista di controllo che verrà utilizzata per le successive inspection.

### 3.3 Gestione delle versioni

Il processo di verifica inizia fin dal primo rilascio ufficiale nella repository di un documento vX.0.0, il risultato del processo di verifica cambia la versione del documento, incrementandone la versione al secondo livello di profondità, il documento quindi passa a vX.1.0. Ogni modifica apportata al documento ne aumenta la versione all'ultimo livello di profondità. Una volta completate le modifiche e le verifiche continue sui documenti, esso passa nella fase di validazione, a carico del responsabile, che ne fa cambiare la versione al secondo livello di priorità.

Grazie al diario delle modifiche è più facile individuare dove concentrare l'attenzione nelle verifiche successive alla prima, in quanto sono segnate le sezioni che sono state aggiunte o che hanno subito dei cambiamenti dall'ultima verifica, evitando così di controllare sempre tutto il documento ad ogni rilascio di versione, per cui ottimizzando il tempo necessario al controllo.

Per ognuna delle 5 fasi del progetto descritte nel *Piano di Progetto v2.0.0*, sono necessarie diverse attività di verifica, descritte nella sezione Analisi del *Piano di Qualifica v3.0.0*, a causa dei diversi output ottenuti:

- **Scelta ed approccio al capitolato:** Si devono eseguire le attività di verifica sui processi e sui documenti.
- **Analisi di dettaglio:** Si devono eseguire le attività di verifica sui processi e sui documenti.
- **Progettazione e sviluppo:** Si devono eseguire le attività di verifica sui processi, sui documenti e sul codice prodotto.
- **Sviluppo ulteriore ed incremento:** Si devono eseguire le attività di verifica sui processi, sui documenti e sul codice prodotto.
- **Progettazione e sviluppo:** Si devono eseguire le attività di verifica sui processi, sui manuali e sul prodotto finito.

Come detto in precedenza conclusa la verifica, ha inizio il processo di approvazione. Durante questo processo è compito del Responsabile di Progetto accertarsi che i prodotti ottenuti siano conformi a quanto pianificato e progettato.

## 3.4 Qualifica

### 3.4.1 Procedure di controllo qualità per i processi

La qualità del processo viene garantita da:

- **Pianificazione:** i processi devono essere pianificati nel dettaglio, in maniera da determinare i punti e le tempistiche in cui effettuare controlli.
- **Controllo:** i controlli pianificati devono essere eseguiti in maniera oggettiva e neutrale, quindi con strumenti automatici ovunque possibile. Alla fine di ogni fase di progetto è necessario calcolare le metriche per i processi indicate nel *Piano di Qualifica v4.0.0* e trarne delle conclusioni.
- **Miglioramento continuo:** l'adozione del principio di *PDCA* implica che alle conclusioni tratte dalle metriche seguano degli accorgimenti atti a contrastare le anomalie riscontrate. Tali accorgimenti possono riguardare le pianificazioni delle fasi successive, l'adozione di tecnologie diverse e l'organizzazione, ma vanno presi tenendo sempre conto dell'impatto che essi avranno sul budget a disposizione.

### 3.4.2 Procedure di controllo qualità per i documenti

La qualità della documentazione viene garantita da:

- **Verifica:** le attività di verifica sui documenti non devono mai essere trascurate. Nelle prime fasi del progetto i verificatori dovranno sottoporre i documenti ad un'analisi di tipo *Walkthrough*, per passare poi quando possibile all'*Inspection*. Inoltre al termine di ogni fase è necessario verificare che le metriche indicate nel *Piano di Qualifica v4.0.0* in sezione 2.2.2 siano entro il limite di accettabilità e provvedere ove opportuno.
- **Validazione:** prende in considerazione l'intero documento ed è atta a dimostrare oggettivamente che sia conforme alle aspettative. Prima della presentazione al Committente il Responsabile di progetto ha il compito di approvare la documentazione.
- **Quality Assurance:** il rispetto delle norme di progetto assicura il raggiungimento degli obiettivi di qualità prefissati, riducendo il ricorso a tecniche retrospettive, e con esse si riducono le iterazioni.

### 3.4.3 Procedure di controllo qualità per il prodotto

La qualità del prodotto viene garantita da:

- **Comprensione ed analisi del dominio** approfondita mediante la produzione della documentazione secondo norme.
- **Verifica:** determina che l'output di una fase sia consistente, completo e corretto. Deve essere eseguita costantemente per tutta la durata del progetto, ma cercando di essere minimamente invasiva. Consiste nel controllo delle metriche indicate nel *Piano di Qualifica v4.0.0* in sezione 2.2.3 e nella produzione dei test di sistema, di integrazione e di unità. In particolar modo le modifiche apportate al codice sono sempre esaminate tramite i test, per evitare l'introduzione di errori. Questa norma preventiva è molto importante in quanto l'individuazione degli errori è un'attività particolarmente difficoltosa e costosa.
- **Validazione:** prende in considerazione l'intero prodotto ed è atta a dimostrare oggettivamente che sia conforme alle aspettative. Prima della presentazione al Committente il prodotto dovrà essere sottoposto a dei test di validazione opportunamente prodotti.
- **Quality Assurance:** il rispetto delle norme di progetto assicura il raggiungimento degli obiettivi di qualità prefissati, riducendo il ricorso a tecniche retrospettive, e con esse si riducono le iterazioni.

## 3.5 Strumenti e metodi per l'applicazione delle metriche

Le risorse software che si utilizzeranno durante il processo di verifica sono:

- **TeXMaker:** un ambiente grafico Open-Source per  $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$  cross-platform, permette la compilazione rapida e la visualizzazione del PDF generato, sarà anche usato per il controllo ortografico;



- **Scalastyle:** analizzatore statico che rileva potenziali problemi nel codice, da utilizzare sia durante la produzione del codice, sia durante la sua verifica (<https://github.com/scalastyle/scalastyle>);
- **Scapegoat:** un altro analizzatore statico che si concentra maggiormente sugli standard di stile e di coding (<https://github.com/sksamuel/scapegoat>);
- **CLOC (Count Lines Of Code):** misura alcune metriche riguardanti il codice sorgente in vari linguaggi, tra cui *Scala*, servirà a calcolare le linee di codice per linee di commento ([cloc.sourceforge.net](http://cloc.sourceforge.net));
- **ScalaTest:** framework per i test su Scala (<http://www.scalatest.org/>).
- **Repo's Outpost:** tool su piattaforma web sviluppato dai componenti del gruppo, disponibile (previo login) all'indirizzo:  
[http://sweeneytreadaas.altervista.org/menuPrincipale/documentation\\_tools/documenti\\_latex.php](http://sweeneytreadaas.altervista.org/menuPrincipale/documentation_tools/documenti_latex.php)  
Fornisce una *dashboard* aggiornata in tempo reale che riporta le principali metriche dei documenti presenti nel repository. I valori sono evidenziati con diversi colori che ne caratterizzano ottimalità, accettabilità o non accettabilità.

Nome documento	Versione	Parole tot	Figure tot	Gulpease	Lunghezza media sezioni
Analisi dei requisiti ->	v 1.3.6	4151	235	48,31	38,44
Glossario ->	v 1.3.0	2060	4	44,30	93,64
Lettera di presentazione ->		139	1	98,35	
Norme di progetto ->	v 1.3.3	4180	34	44,03	87,08
Piano di progetto ->	v 1.4.4	4492	120	45,01	56,15
Piano di qualifica ->	v 1.1.4	5118	14	42,95	111,26
Studio di fattibilità ->	v 1.2.1	1471	4	39,84	49,03

Figura 14: Screenshot del tool Repo's Outpost

- **Camel Calculator:** tool su piattaforma web sviluppato dai componenti del gruppo, disponibile (previo login) all'indirizzo:  
[http://sweeneytreadaas.altervista.org/menuPrincipale/documentation\\_tools/metric\\_calc.php](http://sweeneytreadaas.altervista.org/menuPrincipale/documentation_tools/metric_calc.php)  
Permette di calcolare nel dettaglio le metriche di uno specifico documento. È possibile caricare il file mediante diverse modalità, compresi dei link diretti per i documenti nel repository del gruppo.
- **Gloss Buddy:** tool su piattaforma web sviluppato dai componenti del gruppo, disponibile (previo login) all'indirizzo:  
[http://sweeneytreadaas.altervista.org/menuPrincipale/documentation\\_tools/glossario.php](http://sweeneytreadaas.altervista.org/menuPrincipale/documentation_tools/glossario.php)  
Marca i termini del glossario con la simbologia corretta.

### 3.6 Comunicazione e risoluzione di anomalie

Una anomalia è una violazione da parte di un documento, o unità di codice, di una o più delle seguenti condizioni:

- Conformità alla norme tipografiche o di codifica;
- Appartenenza al range di accettabilità per tutte le metriche descritte nella sezione Visione generale della strategia di verifica del *Piano di Qualifica v3.0.0*;

- Congruenza del prodotto con funzionalità indicate nell'*analisi dei requisiti*;
- Congruenza del codice con il design del prodotto.

Se un *Verificatore* dovesse trovare una anomalia egli è tenuto ad aprire un sotto-ticket all'interno della task-list a lui assegnata. Nel caso la risoluzione del ticket avesse la necessità di essere strutturato in sotto-attività sarà compito del *Responsabile* aprire una nuova task-list ed assegnarla alle figure coinvolte.

## 4 Processi organizzativi

### 4.1 Processi di gestione dell'infrastruttura

#### 4.1.1 Documentazione di pianificazione

Per quanto riguarda la documentazione della pianificazione si è scelto di adoperare *ProjectLibre*, un software Open-source per il project management, che permette di automatizzare molte mansioni che altrimenti il *Progettista* dovrebbe svolgere a mano.

*ProjectLibre* è stato scelto per le sue ottime caratteristiche:

- Portabilità, in quanto basato su *Java*;
- Open-source;
- Genera automaticamente diagrammi di Gantt, WBS e PERT;
- Calcola automaticamente i costi, sia totali che per singola attività/risorsa, aiutando a tenere sotto controllo il budget.

#### 4.1.2 Ticketing

**Scelta della piattaforma di ticketing** Per quanto riguarda l'emissione e la gestione dei ticket si è scelto di affidarsi alla piattaforma *Teamwork* in quanto:

- Ha ottenuto buoni punteggi da reviews di utenti e di critica;
- Fornisce 100Mb di storage e la possibilità di avere due progetti attivi, contemporaneamente;
- Fornisce un analizzatore di rischi e benefici;
- Genera automaticamente diagrammi di Gantt interattivi;
- Include un ottimo Task management (priorità, task history, possibilità di aggiungere in automatico task ricorrenti);
- Notifiche sms e *Notification group*.

La principale alternativa presa in considerazione è stata *Zoho*, ma non è stata ritenuta all'altezza in quanto offre meno features. Segue una breve lista per mettere a confronto le principali funzionalità messe a disposizione dalle due piattaforme:

ZOHO	TEAMWORK
Calendar	Calendar
Gant	Gantt interattivi
Task management	To-do list
Time tracking	Track Project Hours
Bug tracking	Analizzatore rischi/benefici
Document management	Template di progetto
	Priorities
	Track Burn Rate
	Track Staff Hours
	SMS di notifica

Tabella 4: Zoho / Teamwork - Lista features

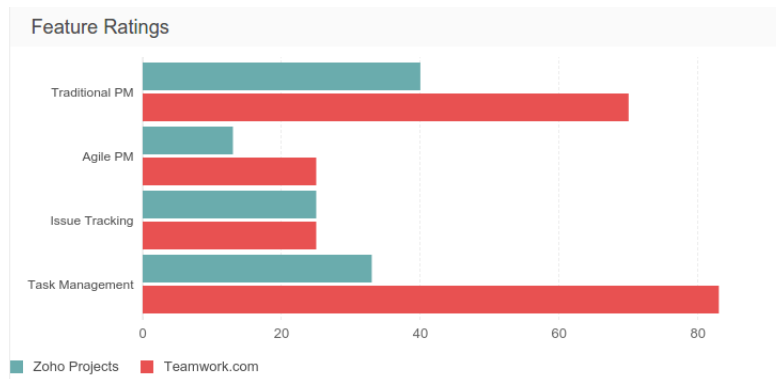


Figura 15: Zoho / Teamwork - Rating delle features a confronto

Secondo *SoftwareInsider* ([softwareinsider.com](http://softwareinsider.com)) sono molto simili nelle funzionalità principali; ma *Teamwork* offre alcuni strumenti in più per la gestione di processi software tradizionali.

I principali:

	ZOHO	TEAMWORK
Calendar	✓	✓
Gantt interattivi	✓	✓
Template di progetto	✓	✓
Risk/benefits analyzer	✗	✓
Scheduling	✗	✓

Tabella 5: Zoho / Teamwork - Differenza strumenti

Come task management *Zoho* offre solamente delle To-do List, mentre *Teamwork* ha anche le seguenti feature:

- Add Recurring Tasks;
- Group Tasks by Projects;
- Set Priorities;
- Task History.

*Zoho* offre alcune funzionalità in più in quanto a comunicazione real-time tra membri del gruppo, ma questo risulta irrilevante per il nostro gruppo, in quanto per la comunicazione real-time viene adottato un sistema diverso. **Politiche di ticketing** I Ticket devono essere assegnati dal *Responsabile*. Poiché un ticket assegna una o più attività ad una o più persone, chi riceve un ticket può generare altri ticket relativi a sotto attività da lui individuate per tenere traccia dello sviluppo in maniera più chiara. In ogni caso a tutti gli altri membri del gruppo non è assolutamente permessa la generazione di ticket esterni ad attività già assegnate.

#### 4.1.3 Versioning

Per gestire il versionamento il gruppo utilizza *GitHub*. Tale scelta è dovuta sia ad un apprezzamento comune da parte dei membri del gruppo per la piattaforma, che ad una richiesta esplicita di pubblicazione del progetto sulla stessa da parte del committente.

È stato creato un account ufficiale del gruppo, raggiungibile all'indirizzo <https://github.com/SweeneyThreads>

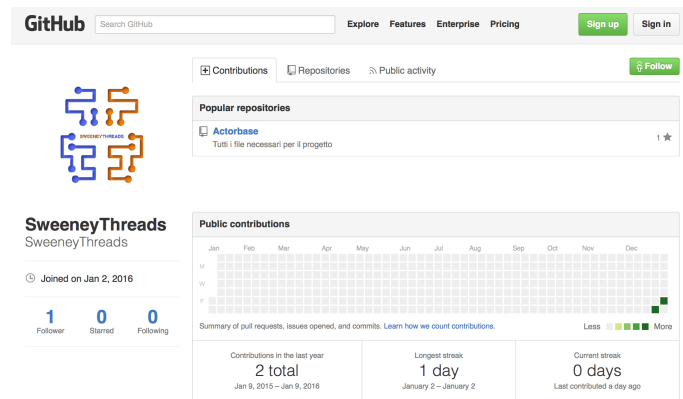


Figura 16: Account GitHub SWEeneyThreads

#### 4.1.4 Repository

Sono state previste diverse *Repository* necessarie allo sviluppo del progetto: Actorbase, ActorbaseDoc, RR, RP, RQ, RA e ConsegnaActorbase. Actorbase conterrà tutti i file del prodotto da sviluppare, mentre le *Repository* RR, RP, RQ e RA si riferiscono alle 4 consegne del progetto previste: revisione dei requisiti, revisione di progettazione, revisione di qualifica, revisione di accettazione. ActorbaseDoc conterrà tutta la documentazione necessaria per il progetto. Effettuata la consegna del materiale, la *Repository* Actorbase/ verrà copiata in quella corrispondente, che servirà quindi come backup della *Baseline* a cui fa riferimento. La *Repository* ConsegnaActorbase servirà per effettuare le consegne, il link alla repository verrà consegnato al committente nel giorno delle consegne.

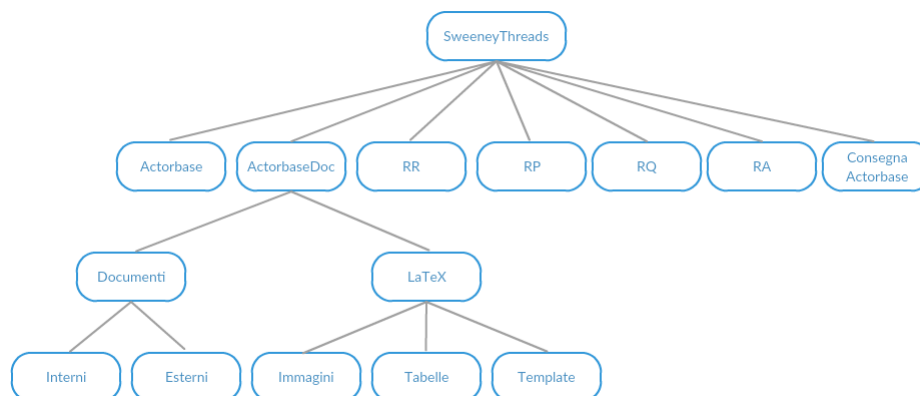


Figura 17: Struttura delle *Repository* GitHub

In ActorbaseDoc/ saranno presenti le seguenti sottocartelle:

- Documenti;
- LaTeX;

##### Documenti

Nella cartella ActorbaseDoc/Documenti/ verranno inseriti tutti i PDF generati dal comando `pdflatex nome-documento.tex`. Non saranno presenti altri file in questa cartella. I documenti saranno divisi in *Interni* ed *Esterni* per questo saranno create delle sottocartelle: ActorbaseDoc/Documenti/Interni e ActorbaseDoc/Documenti/Esterni.

##### LaTeX

Nella cartella ActorbaseDoc/LaTeX/ saranno presenti tutti i file `*.tex` pronti per la compilazione. In questa cartella verrà inserita anche una cartella

`ActorbaseDoc/LaTeX/Immagini/` contenente tutte le immagini necessarie alla compilazione dei file. Inoltre verrà aggiunta una cartella `Actorbase/LaTeX/Templates/` contenente i template per la stesura di documenti e per il disegno appropriato di tabelle. Per le tabelle su più pagine, si terranno dei file separati dal file principale in cui sono incluse. Per questo è stata creata la tabella `ActorbaseDoc/LaTeX/Tabelle`.

## Progetto

Per il progetto in sé, è stata creata una repository a parte `Actorbase` in modo da semplificare l'integrazione continua con *Travis*.

## Normative per i commit

Si è deciso di dare a tutti i membri del gruppo la possibilità di effettuare commit sul master-branch del repository senza dover attendere l'approvazione di un account centrale. Tale scelta impone però la definizione di alcune norme di commit da rispettare:

- I commit devono seguire l'emissione di un ticket. Tale norma serve ad evitare un eccessivo numero di commit sul repository contenenti poche modifiche;
- Nel caso il commit interessasse un documento o un file, il numero di versione dello stesso deve risultare aggiornato;
- Nella descrizione del commit è obbligatorio inserire una descrizione delle modifiche effettuate. È preferito l'uso di elenchi puntati, la forma discorsiva va utilizzata solo se non è possibile esprimere il contenuto come elenco.

## 4.2 Processi di management

### 4.2.1 Ruoli

Per quanto riguarda i ruoli, il gruppo utilizzerà quelli definiti nelle slide 7-11 disponibili all'indirizzo: <http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/L04.pdf>. È stato deciso, unanimemente, che le rotazioni dei ruoli principali come *Amministratore* e *Responsabile di progetto* avverranno ogni 2 settimane. Come stabilito, una persona può ricoprire contemporaneamente più ruoli, la rotazione di altri ruoli come *Analista*, *Progettista* e *Programmatore* potrà avvenire meno frequentemente, in quanto potrebbe risultare dannoso dover abbandonare un'attività di analisi o di programmazione prima della sua conclusione.

Il gruppo stabilisce la rotazione dei ruoli in base alle attività, ai task, e alla disponibilità fornita da ogni membro.

È possibile ottenere in qualsiasi momento una panoramica dei ruoli assegnati tramite l'apposita sezione del progetto creato su *Teamwork*, all'indirizzo

<https://actorbase.teamwork.com/projects/188894/projectroles>.

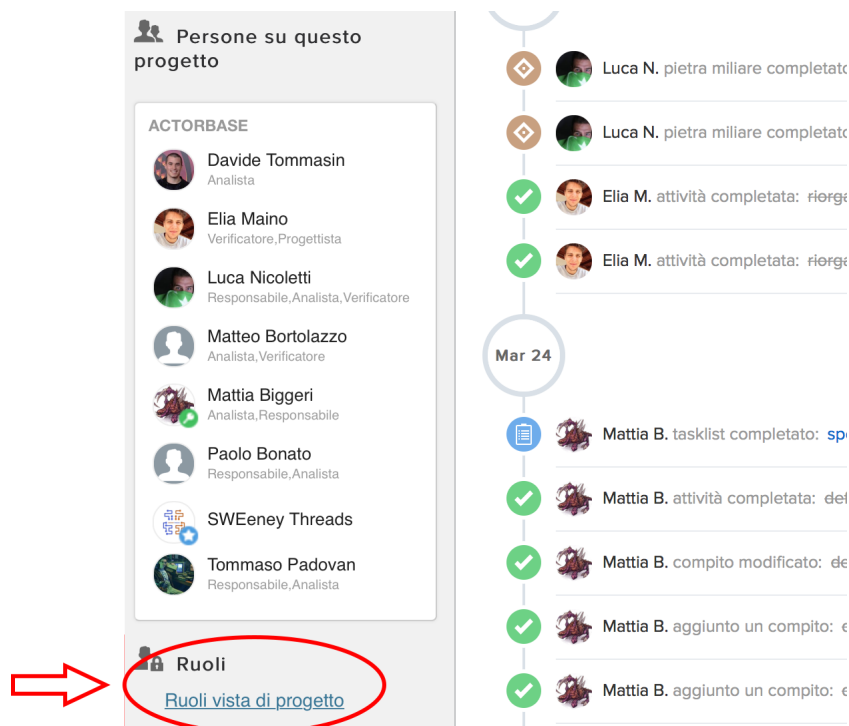


Figura 18: Accesso alla sezione ruoli da Teamwork

Grazie a *SPitFire* è inoltre possibile importare i ruoli definiti in *Teamwork* in un progetto *ProjectLibre* grazie ad un parser che automatizza tale operazione.

#### 4.2.2 Comunicazioni

Le comunicazioni possono avvenire tra membri del gruppo (interne), o tra il gruppo e terzi (esterne). Gli strumenti utilizzati differiscono a seconda della tipologia della comunicazione.

##### Interne

- **Chat:** Per le comunicazioni interne il gruppo ha deciso di adottare una chat di messaggistica istantanea: *Telegram*. All'interno di questo mezzo di comunicazione verranno concordate date e orari delle riunioni; comunicati eventuali ritardi ai meeting; proposte idee informali, che verranno poi riproposte in modo ufficiale alle riunioni (questo per evitare di dimenticarsene o per lasciare tempo agli altri membri del gruppo di ragionare più tempo su una proposta); inoltre *Telegram* verrà utilizzato per l'invio di files temporanei, di documentazione o informativi. Sarà compito del *Responsabile di progetto* prelevare file di documentazione e riportarli nella repository adatta, e nel Drive del gruppo.

La scelta di *Telegram* è dovuta alla possibilità di utilizzare il servizio sia da desktop che da mobile, e alla possibilità di inviare qualsiasi tipo di file;

- **Videoconferenze:** Per le videoconferenze di gruppo si utilizzerà *Google Hangouts*.  
È utilizzabile da tutti i dispositivi e richiede semplicemente un account *Google* di cui disponevano già tutti i membri del gruppo.

##### Esterne

Per tutte le comunicazioni esterne va utilizzata la mail ufficiale del gruppo: [sweeneythreads@gmail.com](mailto:sweeneythreads@gmail.com). La gestione di tale indirizzo email spetta al *Responsabile* che dunque risulta essere l'unico componente del gruppo a poter comunicare con il committente in maniera ufficiale. Il *Responsabile* ha il compito di informare gli altri membri del gruppo sulle discussioni avute con il committente, tale aggiornamento può avvenire a voce durante le riunioni e gli incontri oppure tramite l'inoltro delle email ricevute agli indirizzi personali dei componenti interessati.

Le email ufficiali devono rispettare le seguenti linee guida:

- **Destinatario:** poiché questo indirizzo email va usato esclusivamente per comunicazioni ufficiali il destinatario del messaggio va salvato tra i contatti (funzione di Gmail), nel caso non dovesse già farne parte;
- **Oggetto:** l'oggetto deve esprimere in maniera chiara ed esaustiva il contenuto dell'email, deve essere breve e non deve rendere l'email confondibile con le altre preesistenti. Nel caso il messaggio fosse una risposta l'oggetto deve essere preceduto dalla particella "Re:", nel caso di un inoltrato dalla particella "I:";
- **Corpo:** nel caso il messaggio fosse una risposta o un inoltrato, il contenuto aggiunto va sempre scritto in testa al fine di non costringere i lettori a scorrere tutta l'email. La cancellazione della restante parte del messaggio è sconsigliata, per facilitare una visione completa della conversazione;
- **Allegati:** L'aggiunta di allegati al messaggio è consentita con l'unico vincolo di inviare file che possiedono un nome esplicativo o di specificare il contenuto dell'allegato nel corpo se il nome del file potrebbe essere poco comprensibile.

Inoltre il gruppo ha creato una pagina web di presentazione all'indirizzo [sweeneythreads.github.io](https://sweeneythreads.github.io). Tale pagina, oltre a permettere un rapido accesso alla piattaforma di amministrazione *SPitFire*, contiene diverse informazioni relative al gruppo e allo svolgimento del progetto: consegne effettuate, news, ...

L'aggiornamento della pagina è compito del *Responsabile*.

### 4.2.3 Riunioni

#### Ufficiali

Le riunioni possono essere due tipi: con la presenza del committente o senza la presenza del committente. Il gruppo si impegna a tenere almeno una riunione ufficiale senza presenza del committente ogni due settimane. Le riunioni hanno una durata minima di due ore, che potrà essere prolungata a piacere, in questo caso, nel verbale di riunione dovrà comparire di quanto si è superato il tempo previsto durante la riunione, e il motivo del prolungamento. Queste modifiche sono a carico del *Responsabile di progetto*. I verbali prodotti andranno inseriti nella *Repository Actorbase/Documenti/Verballi*, che verrà suddivisa in due sottocartelle per i verbali interni e quelli esterni.

Le riunioni con presenza del committente, andranno concordate secondo le norme di comunicazioni esterne con quest'ultimo e comunicate tramite i mezzi di comunicazione interni a tutti i membri del gruppo, ognuno dei quali è fortemente tenuto ad essere presente. Potranno verificarsi casi in cui non tutti i membri del gruppo potranno presentarsi alle riunioni con presenza del committente, ma non potrà verificarsi l'assenza del *Responsabile di progetto* e dell'*Amministratore*. I quali sono tenuti a riferire quanto emerso dalle riunioni a tutti i restanti membri assenti. Nelle riunioni con il committente può verificarsi un cambiamento riguardante ad un requisito, in questo caso il cambiamento va inserito nel verbale, che deve essere messo a disposizione del committente nella cartella *Actorbase/Documenti/Verballi/Esterni*.

#### Non ufficiali

Le riunioni non ufficiali sono da considerarsi riunioni tra pochi membri del gruppo, ad esempio tra i due realizzatori di questo stesso documento, o incontri occasionali avvenuti senza comunicazioni nei canali ufficiali. Queste riunioni non necessitano di una stesura di un verbale; se da queste riunioni emergesse un grave errore, o una comunicazione importante, i membri presenti sono tenuti a richiedere una riunione ufficiale straordinaria, che dovrà essere approvata dall'*Amministratore*. In caso contrario, tutte le scelte non rilevanti non necessitano di approvazione. **Brainstorming**

I *Brainstorming* vengono tenuti sotto richiesta di qualsiasi membro del gruppo, e approvati, se per motivazioni valide, dal *Responsabile di progetto*. Un *Brainstorming* ha durata minima di un'ora e massima di due; durante il quale ogni membro ricopre un ruolo di egual importanza rispetto agli altri, le decisioni vengono prese all'unisono o con la maggioranza dei membri a favore, non è compito del *Responsabile di progetto* approvare le soluzioni emerse da un *Brainstorming*.



Durante un *Brainstorming* ci sarà un membro con il compito di scrivere le *Minute*, ovvero un *Notaio*. Ad ogni *Brainstorming* sarà anche scelto un *Moderatore* che ricoprirà un ruolo di servizio. Ovvero dovrà far rispettare le regole di base. Una volta finito il *Brainstorming*, il *Notaio* dovrà riorganizzare gli appunti presi in un verbale ordinato.

## 4.3 Meccanismi di controllo e rendicontazione

### 4.3.1 Meccanismi di controllo

All'interno dell'ambiente di lavoro sono stati predisposti meccanismi per:

- Controllare l'andamento delle attività ed eventuali ritardi;
- Permettere un aggiornamento facilitato della pianificazione;
- Rendicontare le ore di lavoro spese nelle varie attività.

### 4.3.2 Andamento delle attività

Per monitorare i ritardi sulle attività e acquisire maggiore esperienza per stime future si adotta la funzione timer di Teamwork. Ogni componente del gruppo è invitato a tenere attivo il timer durante tutto lo svolgimento delle attività a lui assegnate. In questo modo si può avere una misurazione del tempo effettivo impiegato da ogni membro per svolgere le attività, che può poi essere confrontata con la stima fatta a priori.

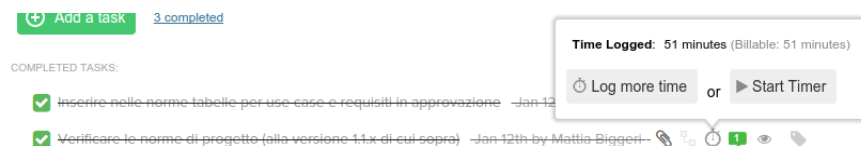


Figura 19: Timer da attivare durante il lavoro svolto.

Inoltre per ogni attività è predisposta anche una *due to date*, ovvero la data entro la quale la task deve essere soddisfatta. Teamwork segnala ogni attività nel riepilogo non completata entro la data di fine con una scritta rossa che riporta il ritardo. È facile per il responsabile individuare a colpo d'occhio le task in ritardo e provvedere a comunicare con il/i componenti del gruppo a cui essa è assegnata per capire le motivazioni del ritardo ed eventualmente rivedere le stime future.

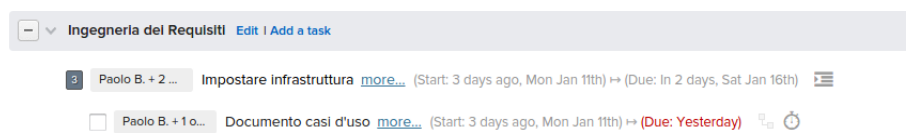


Figura 20: Visualizzazione data di scadenza di un task.

Se necessario è possibile impostare notifiche automatiche in prossimità o al superare di una scadenza. Il sistema di ticketing adottato fornisce un calendario in cui vengono indicate le date stimate di inizio e fine di ogni attività.

Ogni membro del gruppo può consultarlo liberamente per pianificare il proprio lavoro in base agli altri impegni privati.

### 4.3.3 Controllo metriche di progetto

L'introduzione delle metriche nel progetto fornisce una maniera il più possibile oggettiva e sistematica per misurare le performance del gruppo. Dal punto di vista del controllo del progetto le metriche impiegate sono:

- Budget Variance
- Schedule Variance

Questi indicatori permettono al team di:

- Identificare i problemi di costo/schedulazione prima che diventino criticità;
- Aiutare il team a focalizzarsi sul completamento delle proprie attività.

## Elenco delle figure

1	Processi ISO/IEC 12207 . . . . .	4
2	Diagramma di UC1: Connessione al server . . . . .	8
3	Pagina principale di SPitFire . . . . .	10
4	Opzioni di gestione dei requisiti offerte da <i>SPitFire</i> . . . . .	10
5	Tabelle di tracciamento generate da <i>SPitFire</i> . . . . .	11
6	Esempio di legenda per un diagramma dei package . . . . .	12
7	L <sup>A</sup> T <sub>E</sub> X- Commenti ad inizio file . . . . .	13
8	L <sup>A</sup> T <sub>E</sub> X- Indentazione 1 . . . . .	13
9	L <sup>A</sup> T <sub>E</sub> X- Indentazione 2 . . . . .	14
10	L <sup>A</sup> T <sub>E</sub> X- Commento prima di ogni immagine . . . . .	14
11	Scala Style Guide . . . . .	14
12	Logo di Travis CI . . . . .	15
13	Controllo ortografico - strumento di TexMaKer . . . . .	20
14	Screenshot del tool Repo's Outpost . . . . .	23
15	Zoho / Teamwork - Rating delle features a confronto . . . . .	26
16	Account GitHub SWEeneyThreads . . . . .	27
17	Struttura delle <i>Repository GitHub</i> . . . . .	27
18	Accesso alla sezione ruoli da Teamwork . . . . .	29
19	Timer da attivare durante il lavoro svolto. . . . .	31
20	Visualizzazione data di scadenza di un task. . . . .	31

## Elenco delle tabelle

1	Diario delle modifiche . . . . .	3
2	Caso d'uso UC 1 - Connessione al server . . . . .	9
3	Schema del diario delle modifiche . . . . .	17
4	Zoho / Teamwork - Lista features . . . . .	25
5	Zoho / Teamwork - Differenza strumenti . . . . .	26