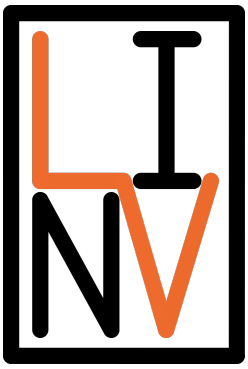

LINV Team

Way of working

Progetto di ingegneria del software
A.A 2022/2023

Informazioni	
Versione	0.10.0
Uso	Interno
Data	07/01/2023
Destinatari	LINV Team
Responsabile	Alessandro Baldissera
Amministratore	Mauro Carnuccio
Verificatori	Nicola Ravagnan
Redattori	Alessandro Baldissera
	Mauro Carnuccio
	Alberto Casado Moreno
	Matteo Cusin
	Nicola Ravagnan
	Riccardo Rossi
	Alessandro Santin

Indice

Elenco delle figure

Elenco delle tabelle

Registro delle modifiche

Ver.	Data	Autore	Ruolo	Descrizione
0.10.0	07/01/2023	Mauro Carnuccio	Amministrator	Aggiunto processo di sviluppo Analisi dei Requisiti
0.9.2	07/01/2023	Mauro Carnuccio	Amministrator	Aggiunto processo di fornitura
0.9.1	06/01/2023	Mauro Carnuccio	Amministrator	Aggiunta codificai ai processi di sviluppo
0.9.0	05/01/2023	Matteo Cusin	Analista	Aggiunta sezione relativa allo script cleanup.sh.
0.8.0	26/12/2022	Matteo Cusin	Amministrator	Aggiornata struttura file, aggiunto schema di flusso delle attività.
0.7.1	26/12/2022	Matteo Cusin	Amministrator	Aggiunti ruoli di progetto, normato incontro interno settimanale.
0.7.0	21/12/2022	Matteo Cusin	Amministrator	Aggiornata la sezione di organizzazione delle attività.
0.6.0	19/12/2022	Matteo Cusin	Amministrator	Aggiornati riferimenti al glossario.
0.5.0	16/12/2022	Matteo Cusin	Amministrator	Aggiornata la sezione delle regole tipografiche.
0.4.0	11/12/2022	Nicola Ravagnan	Amministrator	Sostituito Github Projects con Jira, aggiunta sezione tracciamento ore e altre regole di scrittura.
0.3.1	09/11/2022	Alessandro Baldissera	Amministrator	Aggiornate le norme sul registro delle modifiche e sul numero di versione.
0.3.0	05/11/2022	Alessandro Baldissera	Amministrator	Aggiunta della sezione di verifica, automazione, accettazione e processi organizzativi.
0.2.0	31/10/2022	Alessandro Baldissera	Amministrator	Aggiunta della sezione di stesura della documentazione.
0.1.0	28/10/2022	Alessandro Baldissera	Amministrator	Prima versione preliminare.

1 Introduzione

1.1 Scopo del documento

Lo scopo del documento è quello di definire regole e "best practices" che ogni membro del gruppo si impegna a rispettare per raggiungere in modo efficiente ed efficace la realizzazione del prodotto finale. Vengono inoltre specificate delle convenzioni sull'uso dei vari strumenti scelti per lo sviluppo del prodotto.

1.2 Glossario

Questo documento, come tutti gli altri stilati durante la realizzazione del progetto, è corredato da un ***Glossario*** che si può trovare allegato alla documentazione, nel quale si definiscono tutti i termini specifici al progetto o di significato ambiguo. Quando un termine è definito nel ***Glossario*** si trova una G a pedice del termine stesso.

1.2.1 Raccolta termini del glossario

I termini del ***Glossario*** verranno raccolti tramite un documento Google Docs$_G$ condiviso tra i membri del gruppo dove vengono trascritti termini di rilevante interesse nel dominio del progetto in una check-list. Questi termini in un momento successivo saranno approfonditi nella stesura del ***Glossario*** da parte dell'*Amministratore*.

2 Processi primari

Il seguente capitolo descrive i processi adottati dal gruppo relativamente alle operazioni di fornitura e sviluppo del prodotto.

2.1 Fornitura

2.1.1 Scopo

Il processo di fornitura ha lo scopo di determinare i compiti, le attività e le risorse necessarie allo svolgimento del progetto nel suo complesso soddisfando le richieste del proponente.

2.1.2 Aspettative

Le aspettative del processo di fornitura sono:

- Determinare i bisogni che il prodotto dovrà soddisfare;
- Chiarire i dubbi e stabilire i vincoli con il proponente;
- Stabilire i tempi di lavoro;
- Stimare i costi;
- Ottenere feedback da parte del proponente e dei committenti riguardo al lavoro svolto.

2.1.3 Descrizione

Questa sezione elenca gli obiettivi, i documenti e gli strumenti utilizzati durante il processo di fornitura.

2.1.4 Documenti

Piano di Qualifica
Questo documento descrive le norme necessarie per garantire la qualità dei prodotti.
È diviso nelle seguenti sezioni:

-

Piano di Progetto
Questo documento viene mantenuto durante tutta la durata del progetto.
È diviso nelle seguenti sezioni:

- Stima dei costi;
- Analisi dei rischi;
- Pianificazione del lavoro;
- Preventivo delle ore e dei costi;
- Consuntivo delle ore e dei costi.

2.1.5 Strumenti

Gli strumenti da utilizzare per produrre i documenti precedentemente elencati sono:

- **Google Sheets**: Utilizzato per annotare le ore, calcolare i costi dei vari periodi e per produrre i relativi grafici;
- **Online Gantt**: Utilizzato per creare i diagrammi di Gantt.

2.2 Sviluppo

2.3 *Analisi dei Requisiti*

2.3.1 Scopo

Lo scopo dell'*Analisi dei Requisiti* è di identificare i requisiti obbligatori e desiderabili dei prodotti richiesti dal proponente tramite uno studio approfondito del capitolato.

2.3.2 Aspettative

Il risultato di questa attività è la creazione di un documento che contenga tutti i requisiti richiesti dal proponente anche dal punto di vista dell'utente utilizzatore dei prodotti.

2.3.3 Casi d'Uso

Ciascun Caso d'Uso definisce uno scenario in cui ci sono interazioni con il sistema da parte di uno o più attori.
Il codice dei Casi d'Uso è definito così:

UC[Applicazione]-[Numero caso d'uso].[Sottocaso] [Titolo]

In cui:

- **UC**: Acronimo di "Use Case";
- **Applicazione**: Identifica a quale applicazione si riferisce il Caso d'Uso:
  - **V**: SmartLogViewer;
  - **S**: SmartLogStatistics.

Un Caso d'Uso è composto da:

- Descrizione;
- Scenario;
- Estensioni;
- Attore principale;
- Attore secondario;
- Precondizioni;
- Postcondizioni;
- Generalizzazioni.

2.3.4 Requisiti

I requisiti vengono ricavati nei seguenti modi:

- Analisi approfondita del capitolato;
- Discussioni fra i componenti del gruppo;
- Confronto con il proponente.

Il codice dei requisiti è definito così:

R[Tipololgia][Applicazione]-[Importanza].[Caso d'uso relativo].[Sottocaso]

In cui:

- **R**: Acronimo di "Requirement";
- **Tipologia**:Tipologia del requisito:
  - **F**: Funzionale;
  - **Q**: Qualità;
  - **V**: Vicolo.
- **Applicazione**: Identifica a quale applicazione si riferisce il requisito:
  - **V**: SmartLogViewer;
  - **S**: SmartLogStatistics.
- **Importanza**: Identifica l'importanza del requisito:
  - **1**: Obbligatorio;
  - **2**: Desiderabile;
  - **3**: Opzionale.

2.4 Codifica

2.4.1 Scopo

Questa attività ha lo scopo di mettere in pratica il risultato del processo di progettazione sviluppando il software concreto.

2.4.2 Aspettative

Il risultato di questa attività sarà un prodotto software che rispetti tutti i requisiti richiesti dal proponente. Sono stati decisi dei linguaggi e delle tecnologie da usare, oltre che degli strumenti per lo sviluppo. Inoltre il codice prodotto dovrà rispettare delle regole, elencate in seguito, che permetteranno una coerenza nelle varie parti del codice nonostante la sua scrittura possa essere fatta da persone diverse e una maggiore leggibilità.

2.4.3 Linguaggi

Per il **back-end** verranno utilizzate le seguenti tecnologie:

- **ASP.NET**: un framework per C# per la creazione di web app;
- **Moq**: una libreria per C# per creare mock delle classi per facilitare lo Unit Testing$_G$.

Il **front-end** invece verrà scritto in HTML, CSS e javascript utilizzando le seguenti tecnologie:

- **Angular**: un framework front-end per applicazioni web;
- **Bootstrap v5.3**: una libreria per lo stile delle pagine;
- **D3.js**: una libreria per la creazione dei grafici.

2.4.4 Strumenti

Come descritto nella sezione Utilizzo dei repository, per lo sviluppo delle applicazioni, il gruppo utilizzerà un repository$_G$ dedicato al solo codice.
Per lo sviluppo delle applicazioni potranno essere utilizzati i seguenti ambienti di sviluppo e la scelta è dettata dalla preferenza individuale di ciascun componente del gruppo data la loro intercompatibilità e la possibilità su entrambi di sviluppare sia back-end che front-end:

- **Rider**: un IDE cross-platform prodotto dalla *JetBrains* per lo sviluppo in C# e ASP.NET;
- **Visual Studio 2022**: un IDE Windows-only prodotto dalla *Micorsoft* per lo sviluppo in C# e ASP.NET.

2.4.5 Regole generali di codifica

Regole per C#:

- Variabili:
  - Nomi: camelCase.
- Metodi:
  - Nomi: PascalCase;
  - Allo scopo di rendere il codice più lineare e leggibile, in qualsiasi metodo non possono essere superati i 3 livelli di annidamento.
- Classi:
  - Nomi: PascalCase;
  - Quelle che non devono poter essere ereditate devono essere marchiate con la parola chiave `sealed`;
  - È vietato l'utilizzo della keyword `partial`.
- Interfacce:

  - Nomi: PascalCase;
  - Secondo la documentazione ufficiale di C# i nomi delle interfacce dovrebbero avere come prefisso `I` ad indicare il fatto che sono interfacce. Abbiamo deciso di non adottare questa convenzione, di conseguenza i nomi delle interfacce saranno trattati esattamente come i nomi delle classi.

Regole per il testing:

- Ogni requisito deve avere ben definiti tutti i test necessari per determinare se il requisito implementato rispetta le aspettative;
- Tutti i test devono essere correttamente implementati;
- Nel momento in cui la Pull Request$_G$ è pronta per la revisione, il codice da revisionare deve passare almeno tutti i test relativi alla parte di codice interessata dal ticket.

3 Processi di supporto

Nella seguente sezione sono descritti tutti i processi di supporto legati allo sviluppo del progetto, dalla documentazione alla verifica di tutte le componenti prodotte.

3.1 Documentazione

3.1.1 Scopo

Tutti i processi e le attività di sviluppo dovranno essere documentati al fine di tenere traccia e consultare in modo semplice e veloce ciò che è stato fatto. Questa sezione annota tutte le norme$_G$ per una corretta stesura e verifica di tutta la documentazione che verrà prodotta durante lo sviluppo del progetto.

3.1.2 Ciclo di vita dei documenti

- **Pianificazione**: il contenuto viene ideato e organizzato logicamente. Fase di fondamentale importanza per documenti ricchi di contenuti;
- **Impostazione**: viene creato lo scheletro del documento a partire da un template adatto. Il *Responsabile* quindi crea dei ticket$_G$ sulla board di Jira che rappresentano le varie parti del documento;
- **Scrittura**: i membri identificati alla stesura del documento si auto organizzano, assegnandosi i ticket$_G$ relativi ai contenuti che ognuno deve stilare;
- **Verifica**: ogni documento deve essere verificato da un componente (che non è il redattore del documento stesso) alla ricerca di errori sia grammaticali, logici e di stile;
- **Approvazione**: terminata la verifica il documento passa al *Responsabile* che lo rilegge e decide se approvarlo o meno; se il risultato è positivo si passa alla pubblicazione nel repository$_G$ pubblico, altrimenti si torna alle fasi precedenti per sistemare i problemi identificati.

3.1.3 Strumenti utilizzati

Per scrivere i documenti utilizzeremo il linguaggio LaTeX tramite i seguenti strumenti:

- **pdfTex** versione `3.141592653-2.6-1.40.24` dalla distribuzione **TexLive** scaricabile dal seguente link: `https://www.latex-project.org/get/#tex-distributions`;
- **Microsoft Visual Studio Code** come editor di testo (d'ora in poi VSC);
- **LaTeX Workshop** versione `v8.29.0` come plug-in di VSC come supporto alla scrittura del documento;
- **GitHub Actions**$_G$ per la compilazione automatica dei documenti direttamente nel repository$_G$ attraverso l'immagine **xu-cheng/latex-action@v2**.

3.1.4 Nomi dei file

Tutti i documenti devono essere contenuti nella propria cartella omonima, se il nome del documento è composto da più parole si utilizzano le abbreviazioni che si possono trovare nelle tabelle riguardanti i documenti e le revisioni. Il nome del file invece deve essere pari al nome del documento per esteso in **PascalCase**.

3.1.5 Utilizzo dei template

Tutti i documenti scritti con LaTeX devono inizialmente essere impostati tramite dei comandi. In particolare utilizzeremo le seguenti impostazioni per i nostri documenti:

- **Documenti generici**: definite nel template `../Template/Documento.tex`;
- **Verbali**: definite nel template `../Template/Verbale.tex`;
- **Lettere**: definite nel template `../Template/Lettera.tex`;
- **Preventivi**: definite nel template `../Template/Preventivo.tex`;
- **Pianificazioni**: definite nel template `../Template/Pianificazione.tex`;
- **Consuntivi**: definite nel template `../Template/Consuntivo.tex`.

Per scrivere un nuovo documento basterà copiare il template, rinominare il file e procedere con la scrittura.

3.1.6 Comandi di comodo

Nel file `../Common/Goodies.tex` è presente una raccolta di comandi personalizzati per la formattazione o per inserire particolari costrutti ripetuti nei documenti e mantenere separata la parte logica di supporto e formattazione dai contenuti. Ogni membro del gruppo è libero di inserire un nuovo comando in base alle sue necessità nel file indicato prima. Le regole per l'inserimento di un nuovo comando sono le seguenti:

- Tutti i nuovi comandi devono poter compilare basandosi solo sui pacchetti inclusi nel template, in caso contrario bisognerà attivarsi per correggere tale problema in modo da evitare possibili fallimenti durante la compilazione del codice;
- Ogni comando inserito deve essere preceduto da una descrizione dettagliata di cosa fa e come si utilizza, fornendo una lista esaustiva degli eventuali parametri che necessita;
- Si aggiunge il comando alla lista riassuntiva, all'inizio del file, con una breve descrizione del comportamento.

3.1.7 Prima pagina

Tutti i documenti devono iniziare con un'opportuna intestazione, posizionata in prima pagina, standardizzata nei comandi contenuti nel file `../Common/Intestazione.tex` da importare nel file principale del documento utilizzando il comando `\input`. Le diverse intestazioni sono documentate direttamente nel file `../Common/Intestazione.tex`. Le intestazioni disponibili sono:

- **Documenti generici**: un'intestazione completa ed esaustiva per i principali documenti (ad es. ***Way of Working***, ***Piano di Progetto***, etc.);
- **Verbali**: un'intestazione adattata alla stesura di un verbale;
- **Lettere**: un'intestazione semplificata per la stesura di lettere (ad es. ***Lettera di Candidatura***, lettere di presentazione alle revisioni).

3.1.8 Indice

Tutti i documenti generici (ad eccezione di verbali e lettere) devono essere provvisti di indice. Dove necessario deve essere presente un indice delle tabelle (vedi Tabelle) e delle immagini (vedi Immagini).

3.1.9 Regole tipografiche

- I **titoli** delle sezioni iniziano con la lettera **maiuscola**;
- Gli **elementi degli elenchi** iniziano tutti con la lettera **maiuscola** e terminano con un ';' tranne per l'ultimo elemento che termina con un '.';
- Gli **acronimi** si scrivono utilizzando solo lettere **maiuscole**, tranne nei casi nel quale il nome preveda esplicitamente delle lettere minuscole (ad es. VoIP);
- I **nomi dei ruoli** si scrivono con l'iniziale **maiuscola** e in **italico** (ad es. *Responsabile*);
- I **nomi dei documenti** e delle revisioni si scrivono per esteso, in italico e grassetto, si scrivono con le iniziali maiuscole tranne per le proposizioni (ad es. ***Piano di Progetto***);
- I **nomi dei documenti** e delle **revisioni** che si ripetono spesso possono essere abbreviati seguendo la tabella presentata nella sezione Nomi dei documenti abbreviati, mantenendo sempre la regola del carattere italico e grassetto;
- I **nomi propri** dei componenti del gruppo vanno scritti per esteso, con il **nome** che precede il **cognome** e in italico;
- I **nomi propri** dei componenti del gruppo nelle intestazioni e nel changelog$_G$ vengono scritti per esteso ma non in italico;
- I **nomi dei file** vanno scritti in carattere monospaziato con nome del file ed estensione;
- Le **date** vengono scritte nel formato GG/MM/AAAA utilizzando il comando `\data`;
- I **numeri razionali** si scrivono utilizzando la virgola come separatore tra parte intera e parte decimale.

3.1.10 Stili di testo

Il testo nei documenti può essere enfatizzato utilizzando i seguenti stili:

- **Grassetto**: per parole ritenute importanti;
- **Monospaziato**: per link esterni ai documenti e snippet di codice;
- **Sottolineato**: per riferimenti interni al documento.

3.1.11 Registro delle modifiche

Tutti i documenti generici (ad eccezione di verbali e lettere) devono essere provvisti di un **registro delle modifiche** che contiene un riassunto delle modifiche apportate al documento in formato tabellare. Per convenzione interna, nella tabella vanno riportate **solamente** le azioni di **modifica** del documento, non le fasi di verifica ed approvazione. La tabella non deve essere registrata nell'indice delle tabelle ma deve essere inserita nella sezione **registro delle modifiche** subito dopo l'indice del documento. La tabella deve registrare le seguenti informazioni:

1. **Versione del file**;
2. **Data di rilascio**;
3. **Autore**;
4. **Ruolo**;
5. **Descrizione**: un riassunto delle modifiche apportate.

Per creare la tabella nel file `../Common/Intestazione.tex` è stato definito un ambiente LaTeX chiamato `changelog` che genera la tabella e un nuovo comando `\changelogline` che crea una nuova riga della tabella. Per esempio un registro delle modifiche potrebbe avere un sorgente così strutturato:

```
1 \begin{changelog}
2     \changelogline{0.0.1}{01/11/2022}{Alessandro
        ↪ Baldissera}{Amministratore}{Definita struttura
        ↪ del documento}
3 \end{changelog}
```

3.1.12 Numeri di versione

Sia per i documenti che per i prodotti software è utile che vengano segnalati con numero di versione. Nella fattispecie, lo schema per indicare la versione di un prodotto è la seguente: `X.Y.Z`, dove:

- **X** incrementa quando il *Responsabile* accetta il prodotto;
- **Y** incrementa quando un *Verificatore* revisiona con esito positivo il prodotto;
- **Z** incrementa quando vengono introdotte modifiche significative al prodotto.

Quando i valori di **X** e **Y** incrementano faranno tornare a zero i numeri successivi.

Avanzamento di versione: avviene tramite una richiesta di modifica del *Verificatore* o *Responsabile* e da fare prima dell'accettazione della pull request$_G$. Chi ha effettuato l'ultima modifica registrata nel documento dovrà incrementare di una unità il valore di **X** o **Y** se la richiesta arriva rispettivamente da *Responsabile* o *Verificatore* e azzerare le cifre successive.

3.1.13 Nomi dei documenti e revisioni abbreviati

Nei documenti in cui vi sono molte ripetizioni di nomi di documenti si possono utilizzare le abbreviazioni che si trovano nelle seguenti tabelle:

Nome documento	Nome abbreviato
Piano di Progetto	*PdP*
Piano di Qualifica	*PdQ*
Way of Working	*WoW*
Glossario	*G*
Manuale Utente	*MU*

Tabella 2: Tabella riassuntiva delle abbreviazioni dei nomi dei documenti

Nome revisione	Nome abbreviato
Requirements and Technology Baseline	*RTB*
Product Baseline	*PB*
Customer Acceptance	*CA*

Tabella 3: Tabella riassuntiva delle abbreviazioni dei nomi delle revisioni

3.1.14 Tabelle

Tutte le tabelle che si inseriscono nel documento devono essere provviste di descrizione tramite il comando `\caption` e, quando nel documento si trova almeno una tabella, il documento deve essere provvisto di indice delle tabelle. Inoltre devono rispettare i seguenti stili di base:

- Le righe e colonne di intestazione devono avere un colore diverso dalle righe e colonne che contengono informazioni;
- Devono essere centrate sul foglio;
- Non devono essere di tipo 'flottante' in modo che la tabella venga inserita nel punto nel quale è stata inserita nel sorgente;
- La descrizione (o 'caption') della tabella deve trovarsi sotto.

3.1.15 Immagini

Tutte le immagini che si inseriscono nel documento devono essere provviste di descrizione tramite il comando `\caption` e, quando nel documento è presente almeno un'immagine, il documento deve essere provvisto di indice delle immagini. Inoltre devono rispettare i seguenti stili di base:

- Le figure devono essere centrate nel foglio;
- Non devono essere di tipo 'flottante' in modo che l'immagine venga inserita nel punto nel quale è stata inserita nel sorgente;
- La descrizione (o 'caption') dell'immagine deve trovarsi sotto.

3.1.16 Riferimenti cliccabili all'interno del documento

È possibile inserire dei riferimenti cliccabili all'interno del documento per una migliore navigabilità. Per fare ciò si utilizzano le `\label`$_G$ come segnaposto e `\hyperref` per creare un link. I link devono seguire le seguenti regole:

- Devono essere sottolineati;
- Non devono avere un colore diverso dal resto del testo;
- Non devono usare font monospaziato in quanto riservato ai link a risorse esterne.

3.1.17 Glossario

Il ***Glossario*** è un documento che contiene tutte le definizioni dei termini per cui si ritiene che sia necessario specificarne o disambiguarne il significato. Tutti i membri del gruppo sono liberi di aggiungere termini e definizioni al ***Glossario***, impegnandosi a controllare prima che il termine non sia già presente.

Nei documenti i termini definiti nel ***Glossario*** vengono indicati con una G a pedice dopo la parola che si vuole indicare. Non è obbligatorio che tutte le parole presenti nel ***Glossario*** siano segnalate ma almeno la prima occorrenza deve essere segnalata per ogni documento.

3.1.18 Verbali

I verbali sono dei documenti che seguono una struttura semplificata rispetto ad altri documenti. In particolare, come descritto in Prima pagina e in Indice, i verbali sono provvisti di intestazione più breve e adatta al tipo di documento e sprovvisti di indice. Inoltre i verbali non sono documenti versionati, quindi non necessitano di tabelle di versionamento. Le informazioni generali sono riportate in prima pagina, nell'intestazione e comprendono:

- Riunione interna o esterna;
- Luogo;
- Data;
- Ora di inizio;
- Ora di fine;
- *Responsabile*;
- *Scriba*;
- *Verificatore*;

- Partecipanti: saranno scritti nome e cognome di tutti i singoli partecipanti.

I contenuti del verbale sono così organizzati:

- **Punti di discussione**: riporta, tramite un elenco puntato, i punti di discussione, riflessione o domande da porre al gruppo e/o al proponente;
- **Decisioni**: riporta, tramite elenco puntato, le decisioni prese e le risposte alle domande poste;
- **Extra**: una sezione facoltativa in cui inserire eventuali note sull'incontro.

3.2 Verifica

3.2.1 Scopo

Tutto ciò che viene prodotto dal team, dalla documentazione al software, deve essere verificato e validato da una persona indipendente da colui che ha creato il prodotto. La verifica permette di non introdurre modifiche al prodotto che rischiano una regressione o uno stagnamento del prodotto. Il gruppo si doterà sia di **verifiche manuali**, svolte dai *Verificatori*, sia di verifiche automatiche quali **Unit Test**$_G$, **Integration Test**$_G$, e ogni altra verifica automatica considerata utile.

3.2.2 Quando un prodotto può essere verificato

I prodotti possono essere verificati quando la persona identificata ad apportarne modifiche ha terminato le issue$_G$ relative all'attività e ritiene di aver effettuato un buon lavoro. In questo momento l'assegnatario può aprire una **pull request**$_G$ dal branch$_G$ della feature$_G$ al branch$_G$ **devel**, attivando lo stato **Da revisionare** del ticket$_G$ associato al branch della feature$_G$. A quel punto un *Verificatore* si incarica di verificare il ticket$_G$ spostandolo nella colonna **In revisione**.

3.2.3 Verifica dei documenti

La verifica dei documenti deve assicurarsi che:

- Tutti i ticket$_G$ creati dal ***Responsabile*** associati al documento siano stati completati;
- Il documento prodotto segua la struttura decisa durante la sua progettazione;
- Tutti i contenuti siano consoni;
- Vengano rispettate le regole definite per la stesura dei documenti (vedi sezione Documentazione);
- Non ci siano errori grammaticali;
- La compilazione dei file sorgenti del documento abbia successo;
- Il documento finale non presenti artefatti grafici che ne precludano la consultazione.

3.2.4 Verifica del documento *Analisi dei requisiti*

Il documento ***Analisi dei requisiti*** richiede oltre alla checklist vista nella sezione precedente alcuni ulteriori passi:

- Tutti i casi d'uso identificati devono essere rimandabili ad un requisito identificato nel capitolato;
- Tutti i casi d'uso identificati devono essere forniti di almeno:
  - Titolo;
  - Descrizione;
  - Scenario;
  - Attore principale;
  - Precondizioni;
  - Postcondizioni.
- Per ogni requisito identificato deve comparire nella tabella di tracciamento.

3.2.5 Verifica del codice sorgente

Quando un'attività viene creata, si apre una pull request$_G$ verso il branch$_G$ **devel** e il *Verificatore* può iniziare la fase di verifica, composta dalle seguenti fasi:

1. Controllo che tutte le verifiche automatiche diano esito positivo, per fare ciò si apre la pull request$_G$ su GitHub$_G$ e, nella sezione **check**, i test non devono dare esito negativo;
2. Controllo manuale del sorgente nella sezione **Files changed** avviando una revisione, assicurandosi che i cambiamenti rispettino le regole definite e lasciando dei commenti, evidenziando le righe di codice di riferimento, se qualcosa non è conforme o genera dei dubbi; nel caso un file non presenti problemi può essere indicato come **Viewed** nella revisione;
3. Infine, se le due fasi precedenti vengono completate senza riscontrare problemi da risolvere, la pull request$_G$ può essere accettata e integrata.

3.3 Automazione

3.3.1 Scopo

L'automazione è una fase molto importante nella realizzazione efficiente ed efficace di un progetto. Permette di rendere automatico o semi-automatico un compito ripetitivo e/o tedioso, indifferentemente dalla difficoltà di tale compito. Inoltre permette di eliminare il fattore umano dal compito che potrebbe generare errori involontari.

3.3.2 Automazione project board di Jira

Il software Jira consente di automatizzare i cambiamenti di stato delle attività di progetto basandosi su azioni compiute sui ticket$_G$ o sugli epic$_G$ relativi. L'automazione è una funzionalità di Jira che svincola l'utente dall'azione di spostare manualmente un ticket$_G$ all'interno della board per farlo cambiare di stato: si abbassa così il rischio di avere incongruenza tra lo stato effettivo del progetto e ciò che è rappresentato tramite Jira. Le situazioni in cui l'automazione offerta da Jira è stata modellata sono le seguenti:

- Il passaggio di un ticket$_G$ dallo stato **Da completare** a **In corso** aggiorna i campi **Assegnatario** e **Realizzatore**;
- L'apertura di una pull request$_G$ fa passare un ticket$_G$ da **In corso** a **Da revisionare**;
- La presa in carico di un *Verificatore* fa avanzare il ticket$_G$ nello stato **In revisione**;
- La chiusura della pull request$_G$ fa completare il ticket$_G$;
- I ticket$_G$ in ritardo finiranno nella colonna **In ritardo** in qualunque stato essi si trovino.

3.3.3 Automazione con script

Lo 'scripting' è un'attività di scrittura di codice utilizzata per automatizzare un'attività. Normalmente si crea uno script$_G$ per creare uno strumento estremamente flessibile ed adattato allo use-case preso in considerazione, rispetto ad uno strumento già esistente. Il gruppo non mette dei vincoli sul linguaggio da utilizzare per creare lo script$_G$, tuttavia utilizzare linguaggi e strumenti di base permette a tutti di poter utilizzare lo script$_G$ senza nessuna difficoltà. In ogni caso, chi crea uno script$_G$ deve tener presente che:

- Va documentato annotando il codice e aggiornando l'apposita sezione Script$_G$ custom;
- Il codice deve essere inserito nel repository$_G$ e quindi versionato;
- Lo script$_G$ deve essere specifico per una precisa attività, quindi deve avere dimensioni ridotte.

Documentazione di uno script Per documentare un nuovo script$_G$ si deve:

- Indicare su quale repository$_G$ si trova;
- Indicare il nome dello script$_G$;
- Descriverne il funzionamento e l'utilizzo, specificando ove necessario la lista dei parametri.

3.3.4 Automazione con GitHub Actions

Le GitHub Actions$_G$ sono il sistema di automazione proprietario di GitHub$_G$. Esse permettono di eseguire script$_G$ di shell o software forniti da uno store interno per eseguire attività di ogni genere in modo automatizzato ad ogni evento che accade nel repository$_G$. Ogni membro del gruppo può creare una nuova Action a patto che venga documentata in questo documento nella sezione dedicata GitHub Actions$_G$.

Documentazione di una GitHub Action Per documentare un nuova GitHub Action$_G$ si deve:

- Indicare su quale repository$_G$ si trova;
- Indicare il nome della Action;
- Descriverne il funzionamento ed eventualmente come modificarla.

3.3.5 Script custom

Pulizia dai file temporanei di LaTeX

- Repository$_G$: `linvteam/documentazione`
- Nome: `cleanup.ps1`

Descrizione e utilizzo Lo script$_G$ si occupa di rimuovere tutti i file temporanei creati da LaTeX. Spesso è necessario rimuovere quei file per pulire il repository$_G$ o per far generare il PDF da zero. Per rendere il processo automatico ed evitare di commettere errori causati dalla distrazione si è pensato di creare questo script$_G$ che implementa delle funzioni di sicurezza, ad esempio ignorare la cartella `Common` o `Immagini`.

Per utilizzarlo necessita dei seguenti parametri:

1. Directory target della pulizia o `--all` per far pulire tutte le cartelle del repository$_G$;
2. `--no-pdf` per ignorare l'eliminazione del file PDF generato da LaTeX;
3. `--recursive` o `-r` per attivare la modalità ricorsiva, rispetta comunque le directory ignorate;
4. `--help` o `-h` per vedere un messaggio di aiuto e una lista delle directory ignorate.

Per aggiungere altre cartelle da ignorare, basta aprire lo script e aggiungerle in coda alla lista `$ignoreDirectories`.

Pulizia dai file temporanei di LaTeX per sistemi Linux-based e MacOS

- Repository: `linvteam/documentazione`
- Nome: `cleanup.sh`

Descrizione e utilizzo Lo script è il risultato del processo di porting dello script cleanup.ps1 per sistemi Linux-based ed MacOS; ha come scopo la rimozione dei file di compilazione di documenti scritti in linguaggio LaTeX. Allo stesso modo dello script cleanup.ps1, `cleanup.sh` ignora le cartelle `Common` ed `Immagini` in quanto in esse sono salvati file funzionali alla compilazione di altri documenti. Per utilizzarlo necessita dei seguenti parametri:

1. Directory target della pulizia o `--all` per far pulire tutte le cartelle del repository;
2. `--no-pdf` per ignorare l'eliminazione del file PDF generato da LaTeX;

3. `--recursive` o `-r` per attivare la modalità ricorsiva, rispetta comunque le directory ignorate;

4. `--help` o `-h` per vedere un messaggio di aiuto e una lista delle directory ignorate.

Per aggiungere altre cartelle da ignorare occorre modificare le righe 116 e 183 dello script.

Pubblicazione automatica dei documenti

- Repository$_G$: `linvteam/documentazione`
- Nome: `docpublisher`
- Eseguibile: `index.js`

Descrizione ed utilizzo Lo script si occupa di pubblicare in automatico sul repository `linvteam/SmartLog` i documenti prodotti previa approvazione del *Responsabile*. È stato scelto di eseguire questa operazione in automatico per evitare errori umani, quali dimenticanze o errori di battitura nel processo di pubblicazione. Lo script inizialmente cerca nella repository `linvteam/documentazione` i file PDF dei documenti compilati e, sapendo che si trovano nella stessa cartella del sorgente con lo stesso nome, andrà anche a leggere il contenuto del file sorgente per estrarne alcune informazioni necessarie alla pubblicazione. Una volta lette queste informazioni, lo script si occuperà di compilare il file `README.md` della repository `linvteam\SmartLog`. Lo script in particolare non si occupa né della compilazione né della pubblicazione dei documenti sulla repository, questo compito è delegato alle Github Actions$_G$. Il sorgente dello script è diviso in 4 file:

- `documentFinder.js`: insieme di funzioni che si occupano di individuare i documenti ed estrarne le informazioni necessarie;
- `index.js`: il file principale dello script che crea il nuovo file `README.md`;
- `markdownCreator.js`: insieme di funzioni che si occupano di generare del markdown generico;
- `README-template.js`: il template di partenza per il file `README.md`.

Lo script eseguirà la ricerca direttamente dalla **directory di lavoro**, di conseguenza il comando da eseguire sarà: `node docpublisher/index.js` dalla root della repository; Il risultato sarà il file `OUTPUT.md` da copiare al posto del file `README.md` nella repository `linvteam/SmartLog`.

Lo script può anche essere eseguito in locale per verificare l'output prima di pubblicare i documenti nella repository pubblica.

3.3.6 GitHub Actions

Build e Deploy dei documenti Latex

- Repository: `linvteam/documentazione`
- Nome: `Build and Deploy Latex document`

Descrizione e modifica Questa GitHub Action è composta da due fasi distinte:

- Compilazione automatica dei documenti e caricamento di un archivio `.zip` nell'artifact repository;
- Caricamento automatico dei documenti prodotti nella repository pubblica `linvteam/SmartLog` su approvazione del *Responsabile*.

Per modificare questa Action si deve editare il file `.github/workflows/build.yml`, andando ad aggiungere il nome del file sorgente del documento (inteso come il 'main' del documento) e il nome del file PDF prodotto nei relativi punti indicati dai commenti che si trovano sul file stesso.

La seconda fase al contrario non è necessario che venga modificata per ogni nuovo documento creato. La fase di pubblicazione dei documenti si compone di diverse sottoazioni che vengono eseguite solo se la fase precedente dà esito positivo:

- Scaricamento dei file del repository `linvteam/documentazione`;
- Scaricamento ed estrazione del file `Documenti.zip` dall'artifact repository;
- Scaricamento dei file del repository `linvteam/SmartLog` in una sottodirectory;
- Esecuzione dello script `docpublisher`;
- Copia dei nuovi file PDF dei documenti e dell'output dello script precedente sul repository `linvteam/SmartLog`;
- Commit e push delle modifiche sul repository `linvteam/SmartLog`.

Questa fase viene eseguita solo quando si esegue un'operazione di push sul branch `main` di `linvteam/documentazione`. Questa GitHub Actions per funzionare necessita di un Secret$_G$ di nome `UPLOAD_TOKEN` per eseguire il commit e push delle modifiche nel repository `linvteam/SmartLog`.

3.4 Accettazione

3.4.1 Scopo

Lo scopo della fase di accettazione è rendere consapevole il *Responsabile* del prodotto che si sta rilasciando al *Committente*.

3.4.2 Quando si attiva l'Accettazione

L'accettazione si attiva nel momento in cui i *Verificatori* ritengono che le attività svolte e integrate nel branch$_G$ **devel** dei repository$_G$ si trovano in uno stato sufficientemente maturo e coerente per essere presentate al *Committente*. Un altro momento in cui si deve attivare la fase di accettazione è allo scadere prossimo di un ticket$_G$, il quale prevede la pubblicazione del materiale prodotto.

3.4.3 Come attivare l'Accettazione

Per attivare l'Accettazione i *Verificatori* devono aprire una pull request$_G$ su GitHub$_G$ dal branch$_G$ **devel** verso il branch$_G$ **main**, impostando come Reviewers il *Responsabile*.

3.4.4 Regole di accettazione dei prodotti

Il *Responsabile* deve assicurarsi che:

- I prodotti, visti come black-box$_G$, siano coerenti con gli obbiettivi previsti;
- I prodotti rispettino i vincoli e i requisiti obbligatori imposti dal *Committente*.

3.5 Rilascio

3.5.1 Scopo

Lo scopo della fase di rilascio è rendere disponibile a chiunque (in particolare al *Committente* ed al *Proponente*) una baseline approvata del prodotto: la pubblicazione della baseline avviene in un sito GitHub Pages$_G$ che si basa sul repository$_G$ contenente la documentazione pubblica.

3.5.2 Quando si attiva il Rilascio

L'attivazione del rilascio ha come precondizione l'esecuzione del processo di accettazione sopra descritto: tale processo deve dare esito positivo, ovvero il *Responsabile* deve dare conferma della bontà del prodotto, chiudendo la pull request$_G$ verso il branch$_G$ **main**.

3.5.3 Come attivare il Rilascio

Per attivare il rilascio è necessaria l'accettazione del prodotto da parte del *Responsabile*, con conseguente **chiusura** della pull request$_G$ verso il branch$_G$ **main**.

3.5.4 Regole di Rilascio della documentazione

Tale processo è stato automatizzato tramite lo script$_G$ custom **docpublisher**. Per una corretta esecuzione dello script$_G$, nelle **lettere** deve essere utilizzato il comando `\letter` il quale indica che la tipologia del documento è "lettera" (serve per la suddivisione dei contenuti all'interno del sito GitHub Pages$_G$). Il comando appena citato necessita di un parametro che indica a quale **revisione** è destinata la lettera:

- **Candidatura**;
- **Requirements and Technology Baseline**;
- **Product Baseline**;
- **Customer Acceptance**.

A pubblicazione avvenuta, tutti i documenti saranno disponibili ed organizzati a questo link: `https://linvteam.github.io/SmartLog/`.

4 Processi organizzativi

Il seguente capitolo descrive i ruoli di progetto e tutte le modalità di comunicazione interne ed esterne, oltre a presentare le modalità di ricezione di comunicazioni automatiche generate dagli strumenti utilizzati.

4.1 Ruoli di progetto

Per tutta la durata del progetto, i membri del gruppo assumeranno diversi **ruoli**, ovvero assumeranno le responsabilità e le mansioni di figure professionali in ambito informatico. Le figure professionali disponibili per il progetto sono:

- *Responsabile*: è la figura di riferimento per il progetto e si occupa di:
  - Gestire le relazioni con l'esterno (con gli *stakeholders* in generale);
  - Gestire le risorse (umane ed economiche);
  - Pianificare le attività da svolgere;
  - Preventivare i costi da sostenere;
  - Affrontare scelte chiave relative alle caratteristiche del prodotto;
  - Valutare i rischi delle scelte da effettuare;
  - Eseguire un'opera di retrospettiva in modo continuo nel tempo per migliorare l'economicità (efficienza ed efficacia) delle azioni compiute dal gruppo di lavoro;
  - Redigere il documento ***Piano di Progetto***.
- *Amministratore*: è la figura di riferimento per quanto riguarda le norme di progetto, ovvero le regole di lavoro che il team deve mettere in atto; si occupa di:
  - Selezionare l'uso delle tecnologie che il gruppo dovrà utilizzare nel progetto;
  - Stabilire i protocolli di comunicazione del gruppo (interni ed esterni);
  - Normare le tecnologie ed i processi in uso nel documento ***Way of Working***.
- *Analista*: è la figura di riferimento per quanto riguarda la fase di **Analisi dei requisiti**; si occupa di:
  - Comprendere a fondo i bisogni espressi dal *Proponente* (bisogni "lato utente");
  - Analizzare e scomporre i bisogni "lato utente" fino ad ottenere, per fasi successive, i bisogni "lato soluzione": essi sono le specifiche per il prodotto, ovvero rispondono alla domanda "cosa deve fare il prodotto per soddisfare i bisogni dell'utente?";
  - Redigere il documento ***Analisi dei Requisiti***.
- *Progettista*: è la figura di riferimento per quanto riguarda le scelte progettuali derivate dai requisiti "lato soluzione"; si occupa di:
  - Costruire l'architettura di prodotto tramite scelte architetturali su più livelli di specificità (progettazione logica e progettazione di dettaglio) che soddisfino i requisiti "lato soluzione";

  - Assicurarsi che l'architettura sia di qualità, ovvero che sia dotata di caratteristiche che le consentono di avere certe proprietà desiderabili.
  - Tracciare la corrispondenza tra i requisiti del prodotto (elencati nel documento ***Analisi dei Requisiti***) e le scelte progettuali che li soddisfano.

- *Programmatore*: è la figura di riferimento per la codifica del software di prodotto; si occupa di:
  - Seguire le direttive del *Progettista* codificando le varie parti dell'architettura elaborata;
  - Scrivere i test che il codice deve superare per essere considerato "corretto";
  - Tracciare la corrispondenza tra i requisiti del prodotto (elencati nel documento ***Analisi dei Requisiti***) e le unità di codice che li soddisfano.

- *Verificatore*: è la figura di riferimento per il controllo della correttezza del prodotto; si occupa di:
  - Verificare che le modifiche apportate ai prodotti rispettino il livello di qualità stabilito nel documento ***Piano di Qualifica***;
  - Segnalare possibili correzioni all'*Assegnatario* di un compito.

4.1.1 Rotazione ruoli

Avviene durante la cerimonia di **pianificazione** dello sprint_G (vedi sezione relativa alla Gestione delle attività): il gruppo, attraverso una discussione collettiva, analizza la situazione e valuta quale possa essere la migliore assegnazione dei ruoli ai singoli membri basandosi su:

- **Vincoli di progetto**: ogni membro deve assumere ogni ruolo per una durata in linea con il preventivo orario presentato durante la candidatura;
- **Impegni personali**: qualora un membro del gruppo avesse degli impegni personali che ne pregiudicano la disponibilità per la durata di uno sprint_G, a tale membro verrebbero assegnati compiti (e di conseguenza ruoli) per i quali non è indispensabile un quantitativo di ore superiore alla disponibilità offerta.

Ogni membro del gruppo potrà assumere all'occorrenza più di un ruolo durante ogni sprint_G, evitando accuratamente situazioni in cui si potrebbero avere conflitti di interesse (un membro non può verificare delle modifiche al prodotto se ne è stato l'autore).

4.2 Comunicazioni interne al gruppo

Il gruppo si impegna a mantenere delle comunicazioni attive tra i componenti in modalità sia sincrone che asincrone a seconda delle necessità. Anche in questo ambito l'automazione può essere applicata per veicolare informazioni: le misure adottate dal gruppo sono riportate nella sezione Notifiche automatiche.

4.2.1 Comunicazioni sincrone

Le comunicazioni sincrone avverranno nelle seguenti modalità:

- **Presenza** sia negli intervalli tra le lezioni che in incontri organizzati;
- **Discord**$_G$: che consente di effettuare chiamate, videochiamate e di condividere il contenuto dello schermo dei partecipanti, consentendo inoltre lo scambio di informazioni testuali e multimediali tramite chat.

Gli incontri devono essere pianificati, possibilmente di settimana in settimana in modo tale che i vari componenti del gruppo si possano organizzare per essere presenti attivamente all'incontro. Nell'impossibilità di un componente di prendere parte all'incontro verrà comunque stilato un verbale che sintetizzi i punti di discussione e le relative decisioni prese nell'incontro. Tale verbale sarà reso disponibile il prima possibile nella cartella condivisa su Google Drive$_G$ e successivamente verrà esportato in formato LaTeX e caricato nel repository$_G$ della documentazione privata.

4.2.2 Comunicazioni asincrone

Le comunicazioni asincrone avvengono secondo le seguenti modalità:

- **Gruppo Telegram** da utilizzare per comunicazioni che non richiedono risposta immediata e possono essere recepite in qualsiasi momento da tutti i membri del gruppo;
- **Chat privata Telegram** per comunicazioni dirette e private tra due componenti del gruppo.

Ad ogni modo, Telegram non deve essere inteso come canale di comunicazione formale.

4.2.3 Comunicazioni formali

Le comunicazioni tecniche e formali avvengono secondo le seguenti modalità:

- **Jira** con i metodi descritti nella sezione Gestione delle attività;
- **GitHub Discussions**$_G$ nella quale si può discutere di tematiche rigorosamente tecniche, le quali possono anche essere rese disponibili al committente.

4.3 Comunicazioni esterne al gruppo

Il gruppo è intenzionato a mantenere comunicazioni attive e costanti con il proponente secondo le metodologie descritte di seguito.

4.3.1 Comunicazioni asincrone

Le comunicazioni asincrone avvengono secondo le seguenti modalità:

- **E-Mail** per le comunicazioni formali, come l'organizzazione di incontri o review;
- **Gruppo chat Microsoft Teams**$_G$ per le comunicazioni formali brevi.

4.3.2 Comunicazioni sincrone

Le comunicazioni sincrone avvengono secondo le seguenti modalità:

- **Microsoft Teams**$_G$ per incontri in videochiamata.

4.4 Notifiche automatiche

Per tenere sempre aggiornati tutti i membri del gruppo sugli eventi che accadono nei repository$_G$ del progetto, è attivo il sistema di notifiche automatico di GitHub$_G$ che, attraverso un webhook$_G$, invierà un messaggio in un canale dedicato su Discord$_G$. Queste notifiche riguarderanno nuovi commit$_G$, apertura di pull request$_G$ e revisioni del codice, risultati sul sistema di GitHub Actions$_G$. Per quanto riguarda l'aggiornamento dei ticket$_G$ di Jira vengono utilizzate le notifiche tramite E-Mail, in questo caso verranno notificati solo i diretti interessati.

4.5 Organizzazione delle attività

Le attività da svolgere durante il progetto saranno suddivise in sprint$_G$ aperti e chiusi da due importanti **cerimonie**:

- **Pianificazione**: consiste nell'individuazione delle attività da svolgere nello sprint$_G$, preventivandone costi temporali, eventuale scadenza e priorità;
- **Retrospettiva**: consiste nell'analisi dello sprint$_G$ trascorso dando enfasi alla differenza tra i costi preventivati e quelli effettivamente sostenuti (e le motivazioni dietro ad eventuali differenze); in tale cerimonia si dà rilievo alle opinioni del gruppo di lavoro, ottenendo informazioni che potrebbero essere utili per la pianificazione di futuri sprint$_G$.

Tali cerimonie avranno luogo ogni due settimane, durante le riunioni interne (sezione Svolgimento incontri formali interni).

4.5.1 Forma delle attività

Le attività da svolgere durante il progetto assumeranno la forma di ticket$_G$ del software Jira$_G$: tale software è in grado di fornire una **visione di insieme** delle attività di progetto, tracciandone lo stato. Ogni ticket$_G$ è dotato, in seguito a configurazione effettuata da alcuni membri del gruppo, di alcuni campi dati di utilità per le operazioni di tracciamento e pianificazione:

- **Durata**: indica la durata preventivata per lo svolgimento del compito;
- **Realizzatore**: indica la persona che sta svolgendo l'attività;
- **Verificatore**: indica la persona che deve verificare che l'attività sia stata svolta in modo corretto.

Quando viene individuata un'attività da svolgere, il relativo ticket$_G$ viene:

1. Prelevato dal **backlog**$_G$;

2. Aggiunto alla **project board**: simula una lavagna in cui i post-it sono le attività, incolonnate in base al loro stato.

La board è stata configurata in modo tale da supportare gli sprint$_G$: le attività possono essere organizzate e visualizzate per periodi di tempo personalizzabili (gli sprint$_G$ stessi). Il gruppo cercherà di automatizzare quanto più possibile il cambiamento di stato dei ticket$_G$ in base allo stato effettivo delle relative attività (Vedi Automazione board di Jira).

4.5.2 Ciclo di vita delle attività

I possibili **stati** del ciclo di vita delle attività sono i seguenti:

- **Da completare**: indica un'attività pianificata ma che deve essere presa in carico da un membro del gruppo;
- **In corso**: l'attività è stata assegnata ad un membro del gruppo (vale anche l'auto-assegnazione);
- **Da revisionare**: l'assegnatario dichiara di aver portato a termine il compito assegnatogli ed attende il giudizio di una persona esterna (*Verificatore*);
- **In revisione**: il *Verificatore* sta controllando il lavoro svolto dall'assegnatario, controllo che genererà un **feedback** in base al quale l'assegnatario potrebbe dover correggere il proprio operato (iterando anche la fase di verifica);
- **Completato**: quando il **feedback** del *Verificatore* è positivo, l'attività può dirsi conclusa.
- **In ritardo**: l'attività, avente data di massima registrata nel relativo ticket$_G$, non è stata completata entro la scadenza prefissata.

I ticket$_G$ possono avere una **checklist** associata, definita a priori nella pianificazione dell'attività, per dare una linea guida oggettiva all'*Assegnatario* ed al *Verificatore* durante l'esecuzione del compito e la sua verifica.

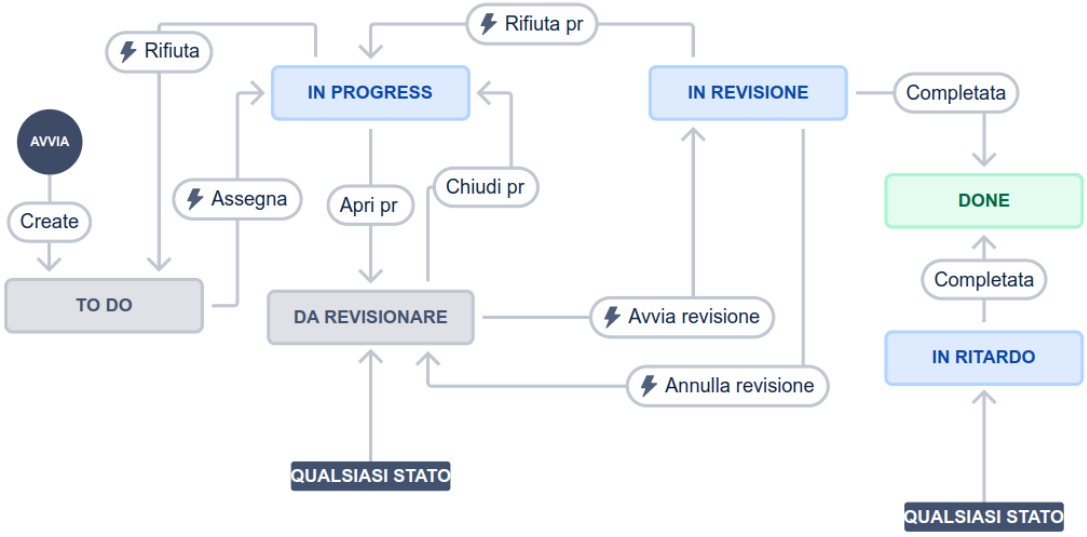

Figura 1: Grafico del flusso delle attività.

4.5.3 Assegnazione delle attività

L'assegnazione delle attività è un'operazione tipicamente asincrona resa visibile a tutto il gruppo di lavoro tramite i campi **Assegnatario** e **Verificatore** dei ticket$_G$. I singoli membri potranno decidere quali attività svolgere in maniera indipendente (coerentemente con il ruolo a loro assegnato al momento della scelta delle attività da svolgere), garantendo un'equa distribuzione del carico di lavoro in base alle disponibilità orarie ed allo stesso tempo il completamento delle attività con priorità maggiore.

4.5.4 Tracciamento ore

Per ogni attività verrà stimata la durata (tramite il campo **Durata** del ticket$_G$) durante ogni cerimonia di **pianificazione**: tale procedimento supporta la stesura del preventivo dei costi di periodo, effettuata tramite un documento Google Sheets da ogni membro in modo tempestivo. Al termine del periodo di sprint$_G$, i membri del gruppo dovranno aver compilato un secondo documento Google Sheets con le ore di lavoro (suddivise per ruolo) effettivamente sostenute, su cui la cerimonia di **retrospettiva** ed il consuntivo di periodo si baseranno.

4.6 Utilizzo dei repository

Il gruppo utilizzerà GIT$_G$, GitHub$_G$ come sistema di versionamento$_G$ e Jira come ITS$_G$ (issue$_G$ tracking system). Allo scopo di separare i contenuti (e facilitare la scrittura del manuale utente del codice) si utilizzeranno 3 repository$_G$, ognuno con il proprio utilizzo:

- **Documentazione privata**: che conterrà tutti i sorgenti LaTeX dei documenti, oltre alla documentazione interna al team;
- **Documentazione pubblica**: che conterrà tutti i documenti da condividere con il committente;
- **Software**: che conterrà tutto il software prodotto nelle varie fasi di sviluppo.

4.6.1 Uso dei branch

Nei vari repository$_G$ utilizzeremo una versione di Gitflow semplificata su 3 livelli di branch$_G$:

- **main** dove verranno convogliate tutte le modifiche per le varie release;
- **devel** derivato da **main**, dove verranno convogliate tutte le attività sviluppate e confermate dai verificatori;
- per ogni ticket$_G$ di **Jira** un branch$_G$ derivante da **devel** dedicato, il nome del branch$_G$ dovrà iniziare con l'identificatore del ticket$_G$ a cui si vuole contribuire.

4.6.2 Convenzioni sui commit

Qualsiasi codice che viene caricato sul repository$_G$ condiviso deve almeno compilare e passare una buona parte dei test scritti. Nel messaggio di commit$_G$ dovrà essere contenuto l'identificatore del ticket$_G$ a cui si riferisce. Il messaggio di commit$_G$ dovrà essere breve ma conciso, senza perdersi in dettagli, eventualmente da descrivere nella issue$_G$ relativa all'attività che si sta svolgendo o in altra sede consona.

Commit verificati Ogni commit$_G$ entrante nel repository$_G$ dovrà essere **firmato** tramite chiave SSH: ogni membro del gruppo, dopo aver creato una chiave SSH per ogni device in uso per il progetto, assocerà le chiavi al proprio account GitHub$_G$. La firma dei commit$_G$ è automatica se preceduta da un'adeguata configurazione del device; GitHub$_G$ garantirà l'autenticazione invertendo la procedura di firma.

4.7 Svolgimento di incontri formali interni

Durante lo sviluppo del progetto saranno necessari degli incontri sincroni e formali nei quali discutere di diverse tematiche sia interne al gruppo che riguardanti lo sviluppo del progetto stesso. Per questo motivo gli incontri formali seguiranno le seguenti regole:

1. Il *Responsabile* trova la data migliore per il prossimo incontro chiedendo ai membri la disponibilità;
2. Una volta trovato il giorno e l'ora dell'incontro, il *Responsabile*, nella cartella condivisa su Google Drive$_G$ dedicata ai verbali, crea un Google Doc$_G$ a partire dal template e si compilano i vari campi dell'intestazione;
3. I membri del gruppo sono liberi di inserire i punti di discussione all'interno del documento;
4. Il *Responsabile* organizza i punti di discussione in base all'urgenza e al tempo a disposizione;
5. Il *Responsabile* dirige la discussione seguendo i punti inseriti precedentemente;
6. Lo *Scriba* eletto dal *Responsabile* trascrive per ogni punto di discussione una breve conclusione;
7. Lo *Scriba* ufficializza il verbale formattandolo in LaTeX e caricandolo nel repository$_G$ della documentazione.

4.7.1 Riunione di aggiornamento e sincronizzazione

Ogni venerdì mattina alle ore 09:00 avrà luogo, sulla piattaforma Discord$_G$, la riunione di **aggiornamento e sincronizzazione**: in tale occasione, i membri del gruppo discutono di situazioni accadute successivamente alla riunione precedente e di eventuali incombenze. Qualora si presentassero situazioni in cui la riunione settimanale non potesse essere svolta (vincoli temporali, personali o semplicemente per mancanza di utilità in un dato istante) le comunicazioni sui canali asincroni dovranno essere tempestive.

4.8 Svolgimento di incontri formali esterni

Durante lo sviluppo del progetto saranno necessari molti incontri con il proponente per discutere di diverse tematiche e fare il punto della situazione sullo sviluppo del prodotto. Gli incontri seguiranno le seguenti regole:

1. Il *Proponente* o il *Responsabile* richiedono di organizzare un incontro;

2. Una volta organizzato l'incontro, il *Responsabile*, nella cartella condivisa su Google Drive$_G$ dedicata ai verbali, crea un Google Doc$_G$ a partire dal template e si compilano i vari campi dell'intestazione;

3. I membri del gruppo sono liberi di inserire i punti di discussione all'interno del documento;

4. Il *Responsabile* organizza i punti di discussione in base all'urgenza e al tempo a disposizione;

5. Il *Responsabile* espone i punti di discussione al proponente, lasciando la parola ai membri del gruppo interessati dove necessario;

6. Lo *Scriba* eletto dal *Responsabile* trascrive per ogni punto di discussione una breve conclusione;

7. Lo *Scriba* ufficializza il verbale formattandolo in LaTeX e caricandolo nel repository$_G$ della documentazione.