Destinatari

Prof. Tullio Vardanega
Prof. Riccardo Cardin

Redattori

Pietro Busato
Linda Barbiero
Guglielmo Barison
Veronica Tecchiati
Davide Donanzan
Oscar Konieczny

Verificatori

Guglielmo Barison
Veronica Tecchiati
Davide Donanzan
Linda Barbiero

Norme di Progetto

nan1fyteam.unipd@gmail.com

Registro delle Modifiche

Versione	Data	Descrizione	Redattore	Verificatore
2.0.0	2024-08-18	**Approvazione per PB**		
1.2.0	2024-08-24	Ultime correzioni.	Pietro Busato	Davide Donanzan
1.1.1	2024-08-18	Correzioni e sistemazioni finali.	Davide Donanzan	Linda Barbiero
1.2.0	2024-08-18	Correzione metriche qualità di processo e di prodotto.	Pietro Busato	Linda Barbiero
1.1.0	2024-08-16	Aggiunta sezione 2.1.4.9 e aggiunta verbali mancanti.	Pietro Busato	Linda Barbiero
1.0.1	2024-08-09	Correzioni generali.	Pietro Busato	Linda Barbiero

Tabella 1: Registro delle modifiche.

Versione	Data	Descrizione	Autore	Ruolo
1.0.0	2024-06-04	**Approvazione per RTB**		
0.11.0	2024-06-04	Verifica completa, correzioni varie.	Guglielmo Barison, Veronica Tecchiati, Davide Donanzan	Verificatore
0.10.0	2024-05-30	Stesura sezione 5.	Pietro Busato, Davide Donanzan	Redattore
0.9.0	2024-05-30	Stesura sottosezione 3.1.	Pietro Busato	Redattore
0.8.1	2024-05-30	Correzioni minori.	Oscar Konieczny, Guglielmo Barison	Redattore
0.8.0	2024-05-29	Stesura sottosezione 2.1.4.1.	Oscar Konieczny, Pietro Busato	Redattore
0.7.1	2024-05-29	Aggiunta diagrammi UML[G] in sezione 2.2.2.	Veronica Tecchiati	Redattore
0.7.0	2024-05-29	Aggiunta sottosezioni 3.2 e 3.3.	Oscar Konieczny	Redattore
0.6.1	2024-05-22	Aggiunta descrizione dettagliata e eventuale formula per ogni metrica.	Linda Barbiero	Redattore
0.6.0	2024-05-21	Stesura sottosezione 2.2.3 e 2.2.4.	Linda Barbiero	Redattore
0.5.0	2024-05-21	Stesura sottosezione 3.4.3.1.	Linda Barbiero	Redattore
0.4.0	2024-04-30	Stesura sottosezioni 4.1.2.2 e 4.1.2.3.	Linda Barbiero	Redattore
0.3.0	2024-04-29	Continuazione sottosezione 4.1.1 e stesura sezioni 4.3 e 4.4	Linda Barbiero	Redattore
0.2.0	2024-04-28	Stesura sottosezione 4.1.1 e 4.1.2.	Linda Barbiero	Redattore
0.1.0	2024-04-27	Stesura sottosezione 4.2.	Linda Barbiero	Redattore
0.0.0	2024-04-27	Stesura del file.	Linda Barbiero	Redattore

Tabella 2: Registro delle modifiche.

Indice

Elenco delle figure

Elenco delle tabelle

1 Introduzione

1.1 Scopo del documento

Il seguente documento ha come scopo quello di elencare le norme che ogni membro del gruppo NaN1fy è tenuto a rispettare durante lo svolgimento del progetto SyncCity presentato dall'azienda Proponente[G] SyncLab.
Inoltre, si delineano le convenzioni relative all'utilizzo dei diversi strumenti selezionati per l'implementazione del prodotto, esponendo dettagliatamente i procedimenti adottati.

1.2 Obiettivi del prodotto

L'obiettivo del progetto SyncCity è quello di creare una piattaforma atta al monitoraggio di sensori sparsi geograficamente nel territorio di una città. I sensori in questione permettono la misurazione e segnalazione di dati real-time[G] riguardanti le più disparate caratteristiche e necessità del territorio quali temperatura ed umidità esterna, occupazione di stalli di parcheggio, funzionamento o guasto elettrico di colonnine di ricarica, traffico stradale e via dicendo. La Proponente[G] richiede la simulazione di alcuni dei sensori nominati nonchè la gestione dei dati, della loro persistenza e della loro rappresentazione grafica attraverso widget[G] e grafici.

SyncCity permetterà un miglioramento della qualità[G] dei servizi offerti dalla città attraverso il continuo monitoraggio della stessa, ottenendo, gestendo e successivamente condividendo i dati con gli utenti.

Il prodotto si struttura nelle seguenti funzionalità[G] principali:

- Raccolta dati;
- Persistenza e strutturazione dati;
- Rappresentazione grafica dati.

1.3 Glossario

Al fine di ovviare a possibili ambiguità dovute al linguaggio e ai termini utilizzati nel seguente documento, viene fornito un *Glossario v2.0.0* contenente le definizioni dei termini utilizzati aventi un significato specifico. Tali termini saranno evidenziati dalla presenza di una G ad apice.

1.4 Riferimenti

1.4.1 Riferimenti Normativi

- *Norme di Progetto v2.0.0*;
- *Verbale Esterno 2024-03-12*;
- *Verbale Esterno 2024-04-19*;
- *Verbale Esterno 2024-04-03*;
- Presentazione e documentazione del capitolato[G] d'appalto C6 - SyncCity:
  - https://www.math.unipd.it~tullio/IS-1/2023/Progetto/C6p.pdf (Ultimo accesso: 26 agosto 2024)
  - https://www.math.unipd.it/~tullio/IS-1/2023/Progetto/C6.pdf (Ultimo accesso: 26 agosto 2024)
- Regolamento progetto didattico:
  - https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/PD2.pdf (Ultimo accesso: 26 agosto 2024)

1.4.2 Riferimenti Informativi

- Documentazione git:[G]
  - https://git-scm.com/docs (Ultimo accesso: 26 agosto 2024)
- Documentazione GitHub:[G]
  - https://docs.github.com/en (Ultimo accesso: 26 agosto 2024)
- Documentazione LaTeX:[G]
  - https://www.guitex.org/home/documentazione (Ultimo accesso: 26 agosto 2024)

2 Processi Primari

2.1 Fornitura

2.1.1 Introduzione

Il processo[G] primario di fornitura, definito dallo standard[G] ISO[G]/IEC[G] 12207-1995 (con ultima versione datata 2017), individua l'insieme di attività atte alla realizzazione di un prodotto software che soddisfi pienamente, a partire dalla formulazione della proposta al Committente[G] fino alla sua ultimazione e consegna, i requisiti e le necessità a cui deve rispondere, garantendo un percorso consono, strutturato ed efficiente, oltre che efficace.

2.1.2 Attività

Tali sono le attività definite nel processo primario di fornitura:

- **Acquisizione e preparazione:** si individuano le necessità del cliente e vengono definiti eventuali requisiti, con associate analisi dei costi pecuniari e temporali (preventivo[G]), e valutazione delle varie opzioni;
- **Contrattazione:** vengono negoziati tra fornitore e cliente i termini e le condizioni contrattuali, e vengono stipulati di comune accordo obiettivi, costi, tempistiche e responsabilitá di entrambe le parti;
- **Pianificazione:** vengono pianificate le attività e le stesure dei documenti utili alla realizzazione del progetto nel rispetto degli accordi fatti;
- **Attuazione e controllo:** vengono eseguite le attività pianificate, controllando regolarmente e consistentemente lo stato di avanzamento e il rispetto degli impegni prefissati conformemente al costo, alle tempistiche e ai requisiti accordati in precedenza;
- **Revisione e valutazione:** vengono effettuate revisioni periodiche e confronti con il cliente, per assicurare il corretto svolgimento del progetto secondo i termini prefissati e risolvere dubbi, incertezze e rischi occorsi;
- **Completamento e consegna:** una volta completato il progetto, viene consegnato al cliente il prodotto finale, secondo quanto stipulato nel contratto.

2.1.3 Rapporti con la Proponente[G]

L'azienda SyncLab si è resa disponibile per realizzare un canale di comunicazione costante e tempestivo, attraverso il quale è possibile colloquiare con i responsabili del progetto affidato al team per dubbi, consigli, organizzazione; tale canale, istituito sulla piattaforma Discord[G], è accompagnato anche dalla possibilità di comunicare con la Proponente tramite posta elettronica. È stato concordato con i responsabili di organizzare periodicamente delle riunioni di Stato Avanzamento Lavori (SAL[G]), fissate solitamente ogni due settimane, il venerdì alle 15:00 (con variazione di orario/data in base a eventuali inconvenienze riscontrate): tali riunioni, che coincidono con la fine dello Sprint[G] corrente e l'inizio del successivo, consistono nell'esposizione dell'operato svolto dal team durante lo Sprint medesimo, con feedback da parte dell'azienda e conseguente discussione di eventuali dubbi o problemi riscontrati e pianificazione del prossimo Sprint. La Proponente mette anche a disposizione la possibilità di effettuare incontri di formazione riguardo alle tecnologie proposte, tenuta da membri esperti dell'azienda e con struttura "deep dive", nei quali chiarire dubbi e delucidare alcuni aspetti delle tecnologie utilizzate. Possono essere infine richieste delle riunioni speciali per risolvere tempestivamente possibili problematiche relative ai requisiti di progetto e ai vincoli imposti dalla Proponente, o per aggiornare la stessa.

2.1.4 Documentazione fornita

A corredo delle attività volte alla realizzazione del progetto, vengono stesi e resi disponibili all'azienda propronente SyncLab e ai Committenti Prof. Vardanega e Prof. Cardin i seguenti documenti:

2.1.4.1 Analisi dei Requisiti

Il documento *Analisi dei Requisiti v2.0.0* illustra e descrive in dettaglio i casi d'uso e i requisiti del progetto, nonchè le funzionalità[G] che ci si aspetta che il prodotto finale abbia, sulla base degli obiettivi posti riguardo al progetto. Tale documento funge quindi anche da base preliminare per la progettazione del software, contenente:

- Insieme dei casi d'uso, ovvero di tutti gli scenari possibili di utilizzo del software da parte degli utenti. Per ogni caso d'uso si individuano e si analizzano:
  - Scenario;
  - Attori;
  - Azioni.
- Lista dei requisiti e dei vincoli definiti e concordati con l'azienda Proponente[G], volti alla realizzazione del prodotto finale.

2.1.4.2 Dichiarazione Impegni

Tale documento manifesta la volontà, da parte del team di sviluppo, di impegnarsi nella realizzazione del prodotto di un dato capitolato[G] (in particolare nel nostro caso di SyncCity, proposto dall'azienda SyncLab); tale documento fornisce inoltre:

- Una suddivisione del monte ore e dei singoli ruoli tra i vari membri del gruppo;
- Una descrizione dei ruoli e del loro rapporto relativo alle specifiche del progetto;
- Una analisi preliminare dei rischi e dei dubbi sorti durante il processo di vaglio dei capitolati;
- Preventivo preliminare dei costi;
- Scadenza prefissata previsa;

2.1.4.3 Glossario

Ai fini di disambiguare ogni possibile dubbio o incertezza, nonchè per rispondere preventivamente e in maniera esaustiva a possibili domande di natura tecnico-linguistica/etimologica da parte dei membri del gruppo, o di chiunque abbia la possibilità e la necessità in futuro di leggere i documenti relativi a tale progetto, viene istituito un *Glossario v2.0.0*, contenente una definizione chiara e univoca di tutti quei termini rilevanti che portano con sè un significato specifico, onde evitare interpretazioni arbitrarie e fraintendimenti riguardo tali specifici concetti.

2.1.4.4 Lettera di Presentatione

La *Lettera di Presentazione* è un documento che accompagna la documentazione e il prodotto forniti all'azienda proponente durante le fasi di revisione del progetto, il cui scopo è quello di dare un veloce contesto intorno allo stato di avanzamento dei lavori (o del loro inizio, per quanto concerne la lettera di presentazione per i capitolati), e fornire una breve panoramica della documentazione redatta fino a quel momento.

2.1.4.5 Manuale Utente

Il documento noto come *Manuale Utente v1.0.0* descrive i requisiti minimi e le istruzioni utili all'installazione/operazione del prodotto finale. Descrive inoltre le funzionalità del prodotto e di come l'utente può usufruirne.

2.1.4.6 Norme di Progetto

La finalità del documento noto come *Norme di Progetto v2.0.0* è quello di redigere e definire un insieme di standard[G] e regole rigurado ai processi e la loro normazione (o Way of Working[G]), da seguire imperativamente da parte del team di sviluppo, durante tutto il ciclo di vita del progetto, in modo da garantirne la qualità[G] e la conformità agli obiettivi e requisiti prefissati con il cliente. I contenuti sono divisi in:

- Processi primari;
- Processi di supporto;
- Processi organizzativi;
- Metriche di qualità.

2.1.4.7 Piano Di Progetto

Il documento *Piano di Progetto v2.0.0*, redatto dal responsabile del team, si occupa di delineare la pianificazione e la gestione delle attività volte alla realizzazione del progetto; tra queste si ha in particolare:

- Analisi dei rischi e delle eventuali problematiche sorte durante lo sviluppo del software, con conseguenti metodi per la loro possibile risoluzione o mitigazione;
- Modello di sviluppo adottato dal team, ovvero descrizione della metodologia e dell'organizzazione delle attività e dei processi utilizzate durante tutto lo sviluppo del software;
- Preventivo dei costi stimati per ciascun periodo e relativo consuntivo[G] con conseguente;
- Aggiornamento costante sullo stato di avanzamento dei lavori, corredato di diagrammi di Gantt[G] che descrivono lo stato di ogni singolo documento o attività relativa al progetto successivo allo Sprint[G] appena percorso.

2.1.4.8 Piano di Qualifica

Il documento *Piano di Qualifica v2.0.0* offre una panoramica dettagliata delle strategie di verifica e validazione adottate per assicurare la qualità[G] del prodotto e dei processi del progetto in questione. Costantemente aggiornato per riflettere l'evoluzione del progetto, si pone come documento dinamico e incrementale che illustra le pratiche per il controllo di qualità degli artefatti e dei processi, con particolare enfasi sulle metriche di valutazione del prodotto. Progettato per guidare l'adozione di processi mirati al miglioramento continuo, si compone di:

- **Qualità di processo:** di fondamentale importanza è assicurarsi, attraverso controlli rigorosi e regolari, che i processi che supportano il prodotto siano ottimali. Applicato a tutte le attività, pratiche e metodologie, svoltesi durante l'intero ciclo di vita del software, si mira a integrare la qualità nel prodotto stesso;
- **Qualità di prodotto:** caratteristiche del prodotto che individuano la sua capacità di soddisfare le esigenze e le richieste del progetto. Concentrandosi su aspetti quali affidabilità, funzionalità[G], manutenibilità e usabilità, si assicura di garantire che il software non solo soddisfi le richieste del cliente e funzioni correttamente, ma che lo faccia conformemente agli standard di qualità[G] adottati;
- **Test[G]:** piano di testing che verrà utilizzato per garantire la correttezza finale del prodotto, comprendente test di unità, di integrazione, di sistema[G] e di accettazione.

2.1.4.9 Specifica Tecnica

Il documento *Specifica Tecnica v2.0.0* viene scritto al fine di descrivere e far comprendere gli aspetti tecnici chiave del progetto, oltre che a fungere da guida per la codifica e la manutenzione dello stesso. La sua finalità primaria è dunque fornire una descrizione dettagliata e approfondita dell'architettura[G] implementativa del sistema[G], fino alle sue radici più profonde, andando quiindi ad esporre dettagliatamente anche il codice e i design pattern[G] utilizzati. Tale documento ha inifine anche il compito di monitorare la copertura dei requisiti identificati nel documento *Analisi dei Requisiti*.

2.1.4.10 Valutazione capitolati

Il documento *Valutazione Capitolati v1.0.0* ha il puro scopo informativo di elencare le motivazioni per le quali si è scelto di proporsi per un determinato progetto tra i vari selezionati, soppesando i requisiti della Proponente e ponderando le criticità del progetto per ognuno di questi.

2.1.4.11 Valutazione dell'utilizzo di Flink

Tale documento, di natura descrittivo-narrativa, spiega il contesto e le motivazioni che hanno portato all'implementazione dello stream processing[G] tramite Apache Flink, e di come esso sia stato implementato.

2.1.4.12 Verbali interni

Viene riportata, sottoforma di verbale, la documentazione concernente le riunioni interne del team di sviluppo, attuate tramite la piattaforma comunicativa Discord[G] a cadenza settimanale, il cui scopo è quello di fissare per iscritto:

- Riassunto dell'andamento dell'ultimo periodo;
- Discussioni, dubbi, proposte, eventuali problemi riscontrati;
- Organizzazione per il prossimo periodo.

2.1.4.13 Verbali esterni

Viene riportata, sottoforma di verbale, la documentazione concernente le riunioni tenutesi con i rappresentanti dell'azienda proponente (SAL[G]), attuate tramite la piattaforma comunicativa Google Meet ogni due settimane, il cui scopo è di fissare per iscritto:

- Resoconto del lavoro svolto durante l'ultimo Sprint[G] con feedback della controparte;
- Discussioni, dubbi, proposte, eventuali problemi riscontrati;
- Organizzazione e obiettivi per il prossimo Sprint. Al contrario dei verbali interni, i verbali esterni verranno, una volta stesi, inviati telematicamente alla Proponente[G] per una loro verifica e convalida del documento.

2.1.5 Strumenti

Di seguito viene riportata una lista di strumenti utilizzati per i processi di fornitura:

- **Discord[G]:** come piattaforma per comunicare da remoto tra membri del gruppo e per raccogliere informazioni e dati utili, come link, appunti et al., oltre che per comunicare in maniera asincrona con l'azienda proponente;
- **Github[G]:** come piattaforma dove condividere il codice del prodotto e i documenti da stendere;
- **Gmail:** come mezzo aggiuntivo e alternativo per la comunicazione asincrona con l'azienda;
- **Google Calendar:** utilizzato dalla Proponente[G] per segnare le date dei vari incontri fissati;
- **Google Meet:** come piattaforma dove vengono svolti gli incontri con l'azienda e i responsabili del progetto;
- **Microsoft Excel:** dove vengono inseriti e organizzati i dati relativi a preventivi e consuntivi dei vari periodi;
- **Telegram[G]:** come piattaforma per comunicare in maniera asincrona tra membri del gruppo.

2.2 Sviluppo

2.2.1 Introduzione

Il processo[G] di sviluppo comprende tutte le attività da svolgere, da parte dei membri del team di sviluppo, il cui fine è quello di ottenere un prodotto in linea con le esigenze e i requisiti fissati dalla Proponente[G]; in particolare si compone di:

- Analisi dei requisiti;
- Progettazione;
- Codifica.

2.2.2 Analisi dei Requisiti

2.2.2.1 Descrizione

Lo scopo del documento di *Analisi dei Requisiti v2.0.0* è quello di definire esaustivamente i casi d'uso, i requisiti e i vincoli concordati insieme alla Proponente[G], per garantire una corretta codifica e implementazione del prodotto software da parte del team.
L'*Analisi dei Requisiti v2.0.0*, redatta dagli Analisti, contiene:

- Una descrizione esplicita delle funzionalità[G] attese dal prodotto;
- L'insieme degli attori, ovvero gli utilizzatori del prodotto finale;
- L'insieme dei casi d'uso, ovvero le interazioni tra gli attori e il sistema[G];
- L'insieme dei requisiti e delle caratteristiche da soddisfare.

2.2.2.2 Obiettivo

Tale processo si propone di comprendere in maniera ultima le esigenze degli utenti, le caratteristiche del prodotto e le condizioni in cui esso dovrà operare.
Questa attività di analisi comporta:

- L'identificazione, in armonia con le esigenze e le proposte dell'azienda, degli obiettivi e delle finalità del prodotto che si intende sviluppare;
- La fornitura di una base chiara e comprensibile ai progettisti per definire più facilmente l'architettura e il design del sistema;
- La facilitazione della comunicazione tra fornitori e stakeholder[G].

2.2.2.3 Diagrammi UML[G] dei casi d'uso

Un diagramma di caso d'uso è uno strumento di modellazione utilizzato nella documentazione e descrizione delle funzionalità di un sistema. La sua utilità si manifesta nella possibilità di rappresentare visivamente, e quindi in maniera veloce e comprensibile, l'interazione tra utente e sistema in uno specifico scenario. Gli scenari d'uso descrivono le azioni atte a consentire all'utente di eseguire con successo una specifica attività; essi sono interconnessi mediante linee. La rappresentazione fornita dai diagrammi dei casi d'uso non si occupa di spiegare eventuali dettagli implementativi, poichè esula dallo scopo principale del documento.

I diagrammi dei casi d'uso sono composti da:

- **Attore:** agente estraneo che interagisce con il sistema per soddisfare una sua necessità;

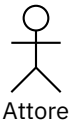

Figura 1: Rappresentazione UML di un attore.

- **Caso d'uso:** funzionalità[G] messa a disposizione da parte del sistema, con la quale l'attore può interagire;

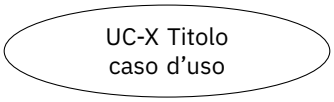

Figura 2: Rappresentazione UML di un caso d'uso.

- **Sottocasi:** i sottocasi espongono in maniera più dettagliata e specifica alcune istanze di casi d'uso più generali;

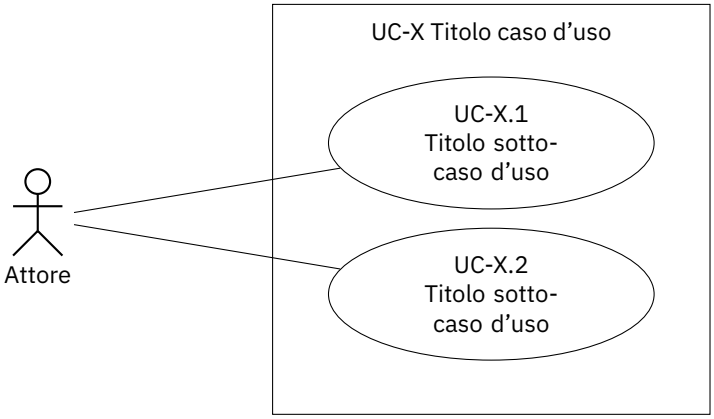

Figura 3: Rappresentazione UML sottocasi d'uso.

- **Sistema**:[G] identifica il prodotto software in produzione, e contiene al suo interno i casi d'uso; al di fuori di esso saranno invece posizionati gli attori (esterni);

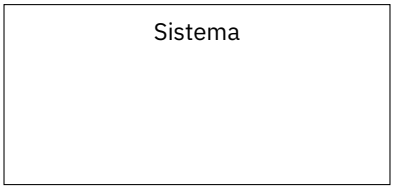

Figura 4: Rappresentazione UML di un sistema.

- **Relazione tra attori e casi d'uso:** tale relazione esplica l'interazione tra un attore, ovvero l'utilizzo di una funzionalità[G] del sistema, e un caso d'uso specifico, ovvero la funzionalitá stessa;

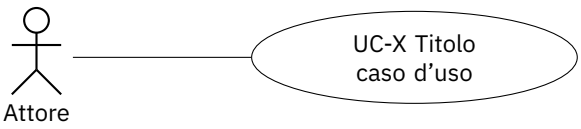

Figura 5: Rappresentazione UML relazione di associazione.

- **Generalizzazione tra attori:** tale relazione identifica un rapporto di tipo ereditario tra due o più attori, uno dei quali (attore specializzato) ottiene caratteristiche e comportamenti da parte dell'attore "padre" (attore base), stabilendo una gerarchia tra gli attori facenti parte dell'interazione con il sistema;

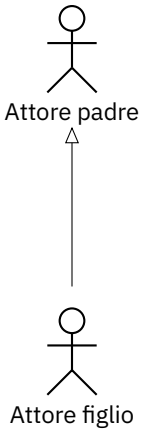

Figura 6: Rappresentazione UML generalizzazione tra attori.

- **Relazioni tra casi d'uso:**
  - **Inclusione:** un caso d'uso viene definito "includente" quando, nell'interazione tra un attore e tale caso d'uso, viene eseguito come parte integrante del processo[G] un altro caso d'uso, detto "incluso". Attraverso questa relazione si evita il problema della duplicazione di casi d'uso superflui;

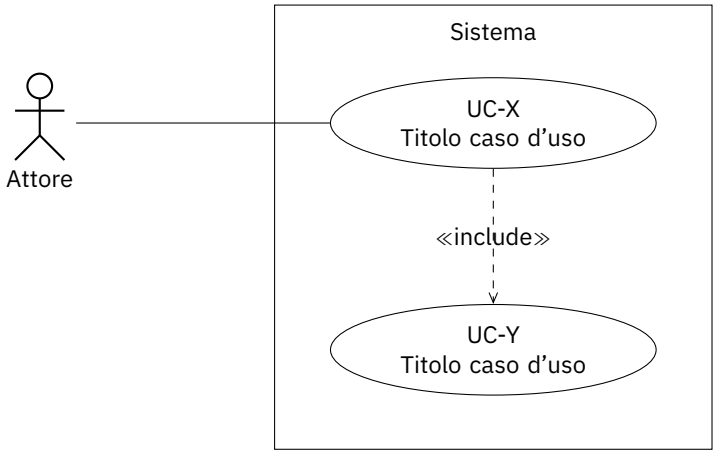

Figura 7: Rappresentazione UML relazione di inclusione.

  - **Estensione:** un caso d'uso viene definito "estendente" quando, nell'interazione tra un attore e tale caso d'uso, al verificarsi di determinate condizioni può essere eseguito un altro caso d'uso, detto "esteso", a complemento o arricchimento del processo in atto. Attraverso tale relazione si è in grado di gestire situazioni particolari senza satollare lo scenario principale di eccezioni e condizioni particolari;

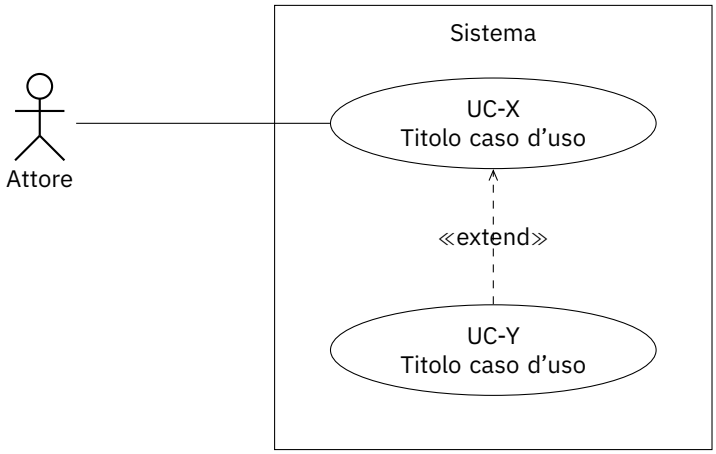

Figura 8: Rappresentazione UML relazione di estensione.

– **Generalizzazione:** nel contesto di una generalizzazione un caso d'uso specifico eredita il proprio comportamento da parte di un altro caso d'uso piú generico.

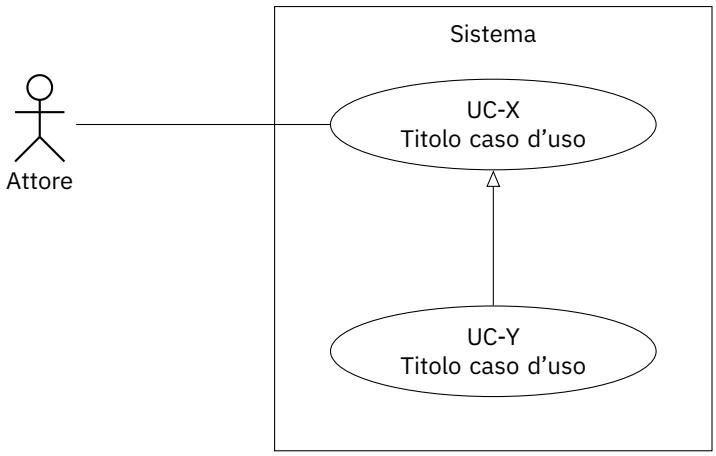

Figura 9: Rappresentazione UML relazione di generalizzazione.

2.2.2.4 Codice dei casi d'uso

Ad ogni caso d'uso è associato un codice univoco definito nel seguente formato:

UC-[Numero].[Specializzazione]

Dove **Numero** è un identificativo e **Specializzazione** si riferisce ad un caso specifico dello stesso caso d'uso.

2.2.2.5 Requisiti

I requisiti di un prodotto software, concordati a inizio progetto insieme alla Proponente, identificano in maniera dettagliata le funzionalità[G], le prestazioni, i vincoli e tutte le specifiche che tale prodotto deve soddisfare. Fungono quindi da guida per lo sviluppo, il testing e la valutazione finale del prodotto, garantendo la sua conformità alle esigenze e agli obiettivi prefissati.
Di seguito vengono elencati i vari tipi di requisito:

- **Funzionali:** questa tipologia di requisiti definisce le funzionalità del sistema[G], ovvero i servizi che esso deve essere in grado di fornire al fine di soddisfare le esigenze dell'utente. Un requisito funzionale descrive quindi il comportamento del sistema;
- **Qualità:** questa tipologia di requisiti definisce lo standard[G] di qualità che il prodotto deve rispettare;
- **Vincolo:** questa tipologia di requisiti indica i limiti imposti dal capitolato[G] che il prodotto deve rispettare;
- **Prestazionali:** questa tipologia di requisiti specifica le prestazioni che il sistema deve avere.

2.2.2.6 Codice dei requisiti

A ciascun requisito è abbinato un codice univoco definito nel seguente modo:

R[Tipologia]-[Numero]

Dove **Tipologia** rappresenta il tipo di requisito e **[Numero]** è un identificativo progressivo.

2.2.3 Progettazione

Lo scopo della progettazione è definire una soluzione possibile ai requisiti dichiarati nell'*Analisi dei Requisiti v2.0.0*.

2.2.3.1 Descrizione

Svolta dai Progettisti, la progettazione è soggetta a due parti:

- **Progettazione logica**: tecnologie, framework[G] e librerie[G], utilizzate per la realizzazione del prodotto, dimostrando la sua adeguatezza utilizzando il PoC[G].
  - Framework e tencologie utilizzate;
  - Proof of Concept[G];
  - Diagrammi UML[G].
- **Progettazione di dettaglio**: base architetturale del prodotto basandosi sulla Progettazione logica.
  - Diagrammi delle classi;
  - Tracciamento delle classi;
  - Design Pattern;
  - Test di unità per componente.

La progettazione risulta anche nella stesura del documento *Specifica Tecnica v1.0.0*, contenente:

- Descrizione approfondita delle tecnologie utilizzate;
- Descrizione dell'architettura implementata;
- Spiegazione motivazioni alla base delle scelte di design;
- Panoramica dei requisiti e dell'ottemperanza degli stessi.

2.2.3.2 Vincoli di qualità[G]

Vengono di seguito elencate le best practice[G] utilizzate durante la progettazione e lo sviluppo del prodotto:

- **Analisi:** si assicura che il prodotto venga progettato e sviluppato soltanto dopo una attenta e approfondita analisi dei requisiti, funzionali e non, stabiliti insieme alla Proponente[G];
- **Affidabilità:** si assicura che il software svolga correttamente e in maniera prevedibile e verificabile le proprie funzioni;
- **Coesione:** si assicura un ottimizzazione dei singoli componenti del software a livello di coesione, garantendo il più possibile che essi lavorino su una singola responsabilità ben definita;
- **Chiarezza:** si assicura che sia il prodotto sia la documentazione di sostegno siano facilmente comprensibili, chiari e completi di immagini, diagrammi UML[G], descrizioni, spiegazioni e commenti;
- **Decoupling:** si assicura un ottimizzazione dei singoli componenti del software a livello di accoppiamento, garantendo il più possibile che essi lavorino indipendentemente gli uni con gli altri, riducendo interazioni e dipendenze, migliorando così manutenibilità e flessibilità del sistema[G];
- **Flessibilità:** si assicura la possibilità di adattare o estendere facilmente il prodotto per soddisfare nuovi requisiti senza dover modificare in maniera estensiva il prodotto;
- **Incapsulamento:** si assicura l'occultamento, all'interno del prodotto, dei dettagli implementativi dei singoli componenti, se visti dall'esterno, consentendo l'accesso e l'utilizzo solo in maniere predefinite;
- **Modularità:** si assicura la suddivisione dell'architettura in moduli indipendenti e ben definiti, in maniera tale da consentire coesione e decoupling e facilitando scalabilità, flessibilità e manutenzione;
- **Riusabilità:** si assicura, attraverso astrazione, incapsulamento et similia la possibilità di poter riutilizzare in contesti diversi e non taluni parti di codice o architettura, per quanto possibile;
- **Scalabilità:** si assicura la progettazione dell'architettura in maniera tale da poter gestire, senza problemi e in modo efficiente, aumenti di carico e aggiunte di nuove funzionalità[G];
- **Semplicità:** si assicura un ottimizzazione dei singoli componenti e del prodotto a livello di semplicità, garantendo il più possibile un codice e un'architettura puliti e privi di sezioni nebulose o inutilmente complesse;
- **Sufficienza:** si assicura l'adempimento del più alto numero possibile di requisiti, funzionali e non, pattuiti con l'azienda Proponente[G].

2.2.3.3 Diagrammi delle classi

Vengono utilizzati, a complemento della progettazione del prodotto, i diagrammi delle classi, necessari per definire caratteristiche e relazioni tra componenti del sistema.

Una classe viene rappresentata graficamente come un rettangolo, diviso in tre sezioni:

1. **Nome della classe:** nome identificativo della classe; esso viene rappresentato in grassetto per la convenzione PascalCase, ad eccezione dei nomi delle classi astratte, scritti invece in corsivo. Il nome deve essere chiaro e indicare le funzionalità/responsabilità della classe;
2. **Attributi:** campi dati della classe, vengono rappresentati singolarmente, divisi in righe, secondo il seguente formato:

   ```
   Visibilità Nome:  Tipo<T> [Molteplicità] = Valore/i di Default
   ```

   dove:

   - **Visibilità:** precede ogni attributo, e segue questi indicativi:
     - —: visibilità privata;

  - $\#$: visibilità protetta;
  - $+$: visibilità pubblica;
  - $\sim$: visibilità di package.
- **Nome:** nome univoco dell'attributo, viene scritto in camelCase per il linguaggio Java[G], e in underscore per il linguaggio Python[G]. Nel caso di costanti, il nome è scritto in Maiuscolo;
- **Tipo:** indica il tipo, primitivo o meno, dell'attributo;
- $<$**T**$>$**:** indica il tipo di oggetto contenuto nel caso di sequenze o container di elementi, come liste o array;
- **Molteplicità:** indica la lunghezza nel caso di sequenze o container di elementi, come liste o array; se non si conosce a priori la lunghezza degli elementi, si indica con "[*]". In caso di singolo elemento, questo dato è opzionale;
- **Valore/i di Default:** ogni campo dati può essere inizializzato con un valore di default.

3. **Firme dei Metodi:** descrivono il comportamento generale dei singoli metodi. Tutti i metodi, anch'essi rappresentati singolarmente in righe separate, seguono il seguente formato:

   ```
   Visibilità Nome (Parametri Formali):  Tipo di Ritorno
   ```

   dove:

   - **Visibilità:** precede ogni metodo, e segue questi indicativi:
     - $-$: visibilità privata;
     - $\#$: visibilità protetta;
     - $+$: visibilità pubblica;
     - $\sim$: visibilità di package.
   - **Nome:** rappresenta l'identificativo del metodo e descrive in maniera esplicita il suo obiettivo o funzionalità;
   - **Parametri formali:** separati tramite virgole, rappresentano i vari parametri richiesti dal metodo;
   - **Tipo di Ritorno:** determina l'eventuale tipo di ritorno del metodo.

Per convenzione, la maggior parte dei metodi getter/setter e dei costruttori non vengono rappresentati, e l'assenza di attributi o metodi indica una loro assenza all'interno della classe o, nel caso di eredità, eventuali metodi ripetuti che si possono ignorare.

2.2.3.4 Relazioni tra classi

Le classi sono collegate tra loro mediante frecce di varia natura, che rappresentano i rispettivi tipi di relazione e dipendenza che le classi possono avere.
Qui di seguito vengono individuate le possibili relazioni:

- **Dipendenza:**
  si dice che una classe dipende da un'altra classe esterna quando la prima utilizza servizi, metodi o membri forniti dalla seconda classe; tale relazione viene rappresentata con una freccia tratteggiata, che può presentare, in taluni contesti, i testi:
  - «use»: quando la classe chiama semplicemente metodi o utilizza attributi dell'altra classe;
  - «create»: quando la classe, prima di chiamare metodi o utilizzare attributi dell'altra classe, istanzia un costruttore per tale classe.

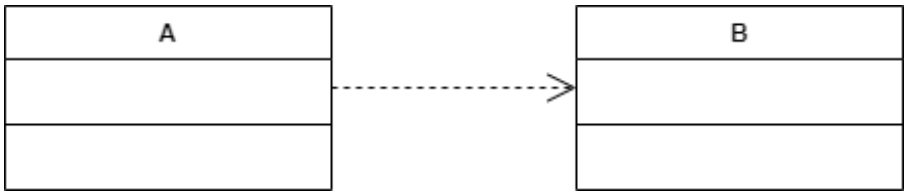

Figura 10: Esempio di dipendenza.

- **Associazione:**
  si dice che una classe è associata ad un'altra quando la prima contiene campi o istanze della seconda; le molteplicità possono venire espresse tramite valori posizionati di fianco alla freccia, che risulta essere una linea continua e orientata; è da notare che se la moltiplicità è uno(1), essa può venire ignorata nel diagramma;

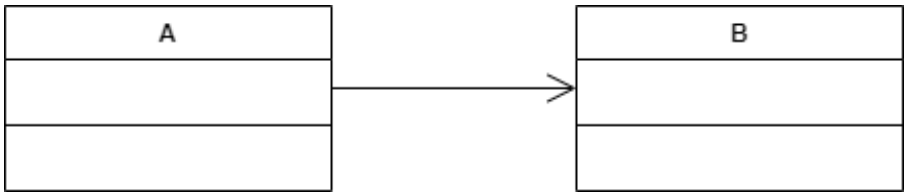

Figura 11: Esempio di associazione.

- **Aggregazione:**
  si dice che una classe è aggregata ad un'altra quando la seconda ha una relazione "Part of" rispetto alla prima (ovvero quest'ultima è composta anche dalla prima), ma le parti possono esistere indipendentemente dalla classe principale. L'aggregazione viene rappresentata tramite una freccia che inizia con un diamante vuoto;

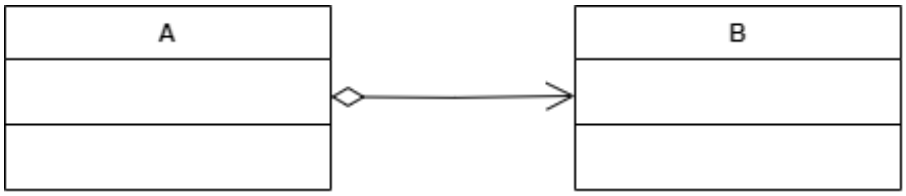

Figura 12: Esempio di aggregazione.

- **Composizione:**
  si dice che una classe è composita con un'altra classe quando la seconda è "Part of" della prima, ma le classi di cui la prima è composta sono dipendenti da essa e non possono esistere indipendentemente. La composizione viene indicata tramite freccia che inizia con un diamante pieno;

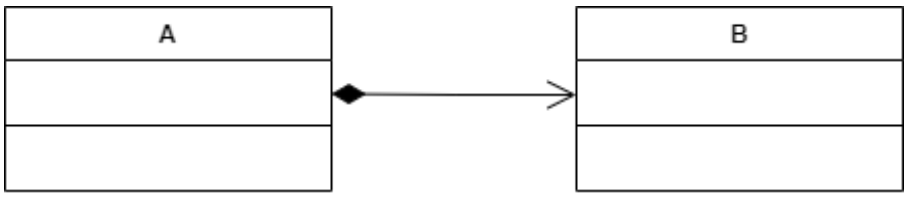

Figura 13: Esempio di composizione.

- **Generalizzazione:**
  si dice che una classe(detta sottoclasse o figlia) generalizza un'altra (eredita da essa, detta superclasse o genitore) quando la prima ha una relazione "Is a" rispetto alla seconda, acquisendone attributi e metodi. La generalizzazione viene indicata con una freccia continua vuota;

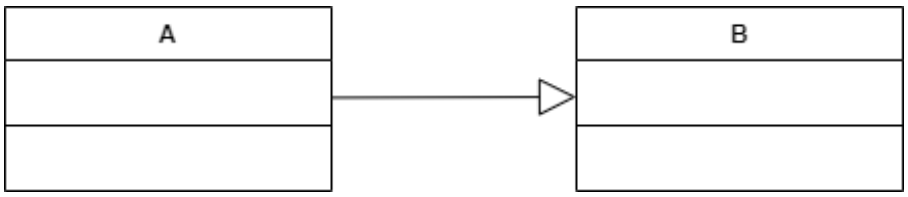

Figura 14: Esempio di generalizzazione.

- **Interface Realization:**
  si dice che una classe realizza un'interfaccia qunado essa fornisce l'implementazione dei metodi definiti in tale specifica interfaccia. Questa relazione è importante quando si vuole mostrare come una classe concreta soddisfi i requisiti di un'interfaccia definendo e implementando i suoi metodi. La classe interessata viene collegata tramite una semplice linea con l'interfaccia, che viene invece rappresentata da un cerchio.

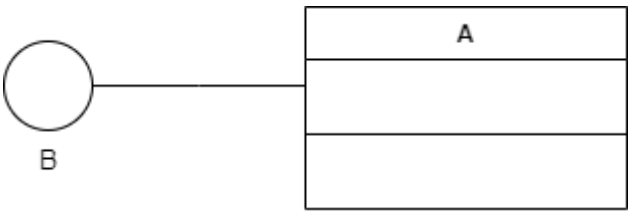

Figura 15: Esempio di realizzazione di interfaccia.

2.2.3.5 Design Pattern

Un design pattern rappresenta una soluzione consolidata e riconosciuta per problemi comuni di progettazione, tendenti a presentarsi durante tutto il processo di sviluppo software. Esso costituisce un modello guida che offre un metodo ben organizzato e testato per risolvere un determinato problema di progettazione, che può essere riutilizzato in diverse situazioni. I design pattern sono sviluppati su base empirica e sono progettati per migliorare efficienza, qualità e manutenibilità del software su cui vengono utilizzati. Ogni design pattern deve includere una descrizione della sua importanza e utilità all'interno dell'architettura, per un'esegesi completa sul suo ruolo e utilizzo.

2.2.3.6 Test

L'insieme complessivo dei test ha il ruolo di assicurare la qualità del prodotto, sia durante le sue fasi di progettazione e implementazione, sia nello stadio finale. Durante la stesura dei test, vengono delineati i requisiti per il testing, definiti i casi di test e i criteri di accettazione, che servono a validare il software. L'obiettivo principale dei test è quello di individuare e correggere errori, bug e difetti presenti nel software prima che esso venga rilasciato. Inoltre, il processo di testing è essenziale per garantire che il software soddisfi le specifiche e i requisiti delineati con il cliente.

2.2.4 Codifica

Lo scopo della codifica, lavoro dei Porgrammatori, è implementare le specifiche individuate concretizzandole su un prodotto utilizzabile. Tale prodotto dovrà poi soddisfare taluni punti, quali:

- Test di unità;
- Conformità di utilizzo dei metodi di codifica e degli standard applicati;
- Test di integrazione;
- Operatività e manutenibilità del prodotto finale.

2.2.4.1 Stile di Codifica

Di seguito vengono riportate le norme adottate dal team di sviluppo durante la stesura del codice del prodotto:

- La lingua adottata per il codice è, ovviamente, l'inglese, mentre per i commenti è consigliato, sebbene non obbligatorio, l'utilizzo dell'italiano;
- Nel caso in cui sia necessario documentare porzioni di codice tramite commenti mirati, è d'obbligo che essi siano concisi e chiari;
- I nomi dei file devono essere univoci ma soprattutto esplicativi, dichiarando con parole chiave il loro contenuto;
- I nomi delle classi e dei metodi (ad eccezione dell'overriding) devono essere univoci; per quanto riguarda la tipologia di nomenclatura, essa varia in base al linguaggio utilizzato (underscore per Python[G], camelCase per Java[G]);
- L'indentazione è fissa a 4 spazi per ogni sorgente;
- La nomenclatura per le costanti è tutta in maiuscolo, con gli spazi sostituiti da underscore;
- Il corpo delle classi e dei metodi deve rimanere il più pulito, compatto e comprensibile possibile;
- L'iterazione è preferita alla ricorsione poichè rende più facile il lavoro di verifica del codice;
- Ogni metodo può contenere un numero massimo di 3 cicli annidati, per non appesantire e complicare il codice.

3 Processi di Supporto

3.1 Documentazione

3.1.1 Introduzione

Viene denominato “documentazione” l’insieme di tutti i documenti a corredo del progetto software, che forniscono tutte le informazioni e i dettagli utili a sviluppatori, utenti e distributori riguardanti il prodotto. Utilizzata prevalentemente dal team di sviluppo per facilitare l’organizzazione e svolgimento delle attività durante il ciclo di vita del software, essa viene mantenuta coerente grazie al tracciamento di tutti i processi e attività che la coinvolgono, in maniera tale da migliorarne costantemente lo stato e semplificare la manutenzione.

3.1.2 Lista dei documenti

Di seguito viene elencato l’insieme dei documenti redatti:

- *Analisi dei Requisiti v2.0.0*;
- *Dichiarazione Impegni v2.0.0*;
- *Glossario v2.0.0*;
- *Lettera di Presentatione*;
- *Manuale Utente v1.0.0*;
- *Norme di Progetto v2.0.0*;
- *Piano di Progetto v2.0.0*;
- *Piano di Qualifica v2.0.0*;
- *Specifica Tecnica v1.0.0*;
- *Valutazione Capitolati v1.0.0*;
- *Valutazione dell’utilizzo di Flink v1.0.0*;
- Verbali Esterni;
- Verbali Interni.

3.1.3 Documentation as Code

Riguardo la documentazione viene adottata la Way of Working[G] del “Documentation as Code”, che prevede la stesura, la gestione e la distribuzione dei documenti utilizzando pratiche e strumenti solitamente utilizzati nello sviluppo di codice o, più in generale, di software. I punti principali di tale Way of Working sono:

- Automazione;
- Collaborazione;
- Distribuzione;
- Integrazione incrementale;
- Versionamento[G].

3.1.4 Ciclo di vita dei documenti

Ogni documento viene redatto e manutenuto secondo questo processo:[G]

1. Viene creata una branch[G] per la stesura del documento nella repository[G] docs: se il file è un verbale, allora viene clonato il template relativo nel branch;
2. Vengono assegnati i membri del gruppo che dovranno redigere il documento;
3. Ogni volta che una sezione o comunque parte del documento viene completata, viene aggiunto nel registro delle modifiche del file una riga in coda, nel formato **[versione x.y.z]**, **[data]**, **[oggetto]**, **[membro del team]**, **[ruolo]**(1);
4. Viene poi caricata nella repository tramite commit[G] sul branch adeguato;
5. Può venire richiesta una pull request[G] per fondere le modifiche fatte fino a quel momento nel branch "main", in maniera tale da costituire una milestone[G] per i singoli documenti, senza ovviamente chiudere il branch;
6. Una volta completato il documento(2), viene richiesta la verifica del contenuto e della forma da parte di almeno due membri del gruppo, prima di procedere con il merge[G] finale e la chiusura della branch;
7. A richiesta dei verificatori, possono essere ripetute parti del processo, in vista di una ottimizzazione del documento;
8. Una volta dichiarato finito il documento, la sua branch viene chiusa e la rispettiva issue[G] viene marcata come "closed";
9. A discrezione del team di sviluppo, la branch di un documento che si ritiene debba essere aggiornato ulteriormente, manutenuto o modificato, previa valida motivazione, può essere riaperto, e il processo ripreso.

Note:

1. Dalla conclusione dell'RTB[G] questo metodo è deprecato; ora il campo **[membro del team]** è sostituito da **[Redattori]**, mentre il campo **[ruolo]** viene sostituito da **[Verificatori]**;
2. Il periodo massimo consentito per l'ultimazione di un documento non mantenuto nel tempo o per la modifica di un qualsiasi documento è della durata di una settimana, se non altrimenti specificato.

Il ciclo di vita dei documenti e delle loro issue[G] consiste nei seguenti quattro stati:

- **Stato "to-do":** il documento deve essere iniziato;
- **Stato "in progress":** il documento sta venendo scritto, completato o aggiornato;
- **Stato "verify":** il documento è in attesa di essere posto sotto scrutinio;
- **Stato "done":** il documento è completo e verificato;

Un documento che deve passare il vaglio per una milestone[G] principale, quale RTB[G] o PB[G] (o eventi straordinari che assumono la stessa "gravitas"), viene controllato e verificato un'ultima volta, in separata sede, da tutti i membri del team. In corrispondenza di questo ultimo controllo, il documento subisce uno scatto di versione (da 0.x.y a 1.x.y, o 2.x.y, e cosí via) che determina la versione definitiva di quel periodo.

3.1.5 Template Verbali

Per semplificare, velocizzare e standardizzare la stesura del documento piú comune prodotto dal team, ovvero i verbali, indipendentemente dal loro essere esterni o interni, è stato sviluppato un template in LaTeX[G], con il quale essi vengono omologati e verificati molto facilmente.

3.1.6 Nomenclatura

Per riferirsi a un documento prodotto dal team, la formulazione del nome è la seguente:

[nome_del_file] + _ + **[versione]** (aggiunta in automatico).

Per i verbali viene utilizzato invece una nomenclatura del tipo:

[VE] (se esterno) / **[VI]** (se interno) + _ + **[data_verbale]**

(con data verbale nel formato "YYYY_MM_DD", dove YYYY indica l'anno, MM il mese e DD il giorno).

3.1.7 Versionamento[G]

Il versionamento, necessario per il tracciamento delle modifiche dei documenti, è strutturato nel formato **X.Y.Z**, come convenzione della versione dei documenti, dove:

- **X:** cifra che viene incrementata quando avviene un rilascio, che nel caso del progetto corrisponde ai raggiungimenti di RTB[G], PB[G] e CA[G];
- **Y:** rappresenta un'aggiunta o modifica sostanziale, come ad esempio l'aggiunta di una sezione;
- **Z:** indica una piccola modifica, come per esempio la correzione di errori o aggiunte di piccole dimensioni.

Un documento parte sempre dalla versione 0.0.0. Ogni modifica di un numero di versione, comporta l'azzeramento di tutti i numeri alla sua destra.

3.1.8 Struttura

Ogni file segue una rigorosa struttura delle pagine, organizzata come segue.

3.1.8.1 Prima Pagina

Nella prima pagina di ogni documento sono presenti, partendo dall'alto e proseguendo verso destra:

- Destinatari del documento;
- Redattori;
- Verificatori;
- Nome del file;
- Logo del team;
- Logo dell'università di Padova;
- Mail ufficiale del team.

Al contenuto di ogni pagina, esclusa la prima, precede un'intestazione con nome del file a sinistra e logo del team a destra, mentre segue un piè di pagina contenente semplicemente il numero della pagina.

3.1.8.2 Registro delle modifiche

Alla prima pagina segue la pagina contenente il changelog del documento; esso è strutturato sottoforma di tabella, ogni riga della quale contiene i seguenti dati:

- Versione del documento;
- Data dell'evento;
- Breve descrizione dell'evento;
- Membri che hanno funto da redattori;
- Membri che hanno funto da verificatori.

Nel caso di una revisione per milestone[G] prinicipale (RTB, PB et similia), essendo che tutti i membri del gruppo vi partecipano, non vengono specificati né redattori né verificatori.

3.1.8.3 Indice
Nella pagina successiva al Registro Modifiche, viene reso disponibile, grazie alla scrittura in LaTeX[G], un indice interattivo che segna ogni sezione e sottosezione del contenuto del documento, comprendente la possibilità di navigare nel file tramite click sulla sezione interessata.

3.1.8.4 Contenuto del documento
Le restanti pagine del documento sono riservate al contenuto del documento stesso, che non segue una precisa organizzazione trasversale tra i documenti, data la loro eterogeneità di contenuti e forma. Unica eccezione sono i verbali, interni ed esterni, che presentano una struttura omologata e omogenea:

- **Informazioni generali:** riguardanti l'incontro e i partecipanti all'incontro;
- **Ordine del giorno:** in cui si descrive brevemente lo scopo dell'incontro;
- **Sintesi dell'incontro:** in cui si descrive ciò che è stato detto durante l'incontro;
- **Conclusioni:** in cui si espone le scelte decise durante l'incontro;
- **Attività da svolgere:** in cui si dichiara gli impegni e le attività da svolgere decise sulla base delle conclusioni tratte dall'incontro.

Nell'ultima pagina dei verbali viene poi apposta, nel corrispettivo spazio dedicatogli, la firma da parte di un rappresenante della Proponente[G], da porre dopo la visione e conferma di tale documento della controparte.

3.1.9 Convenzioni stilistiche

3.1.9.1 Registro modifiche
All'interno della sezione "Descrizione", nel registro delle modifiche di un documento, ogni entry deve terminare con un punto.

3.1.9.2 Elenchi puntati
Le voci di ogni elenco, salvo eccezioni, iniziano con lettera maiuscola e terminano con punto e virgola ';', eccetto l'ultima voce che termina con punto normale '.'.

3.1.9.3 Descrizione immagini
Ogni immagine o tabella presenta una descrizione associata, utile a fornire una breve descrizione o spiegazione del contenuto visivo.

3.1.9.4 Formato delle date
Viene adottato il formato **YYYY-MM-DD**, dove YYYY indica l'anno (4 cifre), MM il mese (2 cifre), e DD il giorno (2 cifre).

3.1.9.5 Abbreviazioni
Numerosi sono i casi in cui vengono utilizzate sigle relative a entità e artefatti relativi al progetto. Tra questi ci sono:

- **Concetti chiave del progetto:**
  - **CA[G]:** Customer Acceptance;
  - **PoC[G]:** Proof of Concept[G];
  - **MVP[G]:** Minimum Viable Product;
  - **PB:[G]** Product Baseline[G];
  - **RTB:[G]** Requirements and Technology Baseline.
- **Ruoli del progetto:**

- **Am:** Amministratore;
- **An:** Analista;
- **Pr:** Programmatore;
- **Pt:** Progettista;
- **Re:** Responsabile;
- **Ve:** Verificatore.

3.1.10 Strumenti

Di seguito vengono elencati tutti gli strumenti unitariamente utilizzati dal team per la stesura, scrittura, manutenzione e verifica dei documenti:

- **GitHub[G]:** piattaforma di hosting di codice sorgente, utilizzato per la condivisione dei file, della loro verifica e del loro versionamento[G], nonché per la automatizzazione di taluni processi, in conformità al concetto di "Documentation as Code";
- **LaTeX:** linguaggio di markup, molto famoso in ambito accademico/scientifico, utilizzato per la stesura dei documenti in maniera pulita ed efficiente.

3.2 Gestione della configurazione

3.2.1 Introduzione

Attuato durante tutto il ciclo di vita di un progetto software, il processo di gestione della configurazione norma il tracciamento e il controllo delle modifiche a documenti e codice prodotti, in maniera da rendere organizzata la procedura di modifica di tali artefatti e la loro evoluzione.

3.2.2 Tecnologie Utilizzate

- **Git:** software utilizzato per il controllo di versione dei Configuration Item;
- **GitHub:** piattaforma web per il controllo di versione utilizzata per l'hosting e coordinamento delle operazioni. Offre anche un Issue[G] Tracking System.

3.2.3 Versionamento[G]

Il versionamento è necessario per il tracciamento delle modifiche che avvengono ai documenti. Grazie al versionamento è possibile visualizzare le modifiche che un file ha subito e, nel caso fosse necessario, far regredire il documento ad una versione precedente.
Il gruppo utilizza il formato X.Y.Z come convenzione della versione dei documenti, dove:

- **X:** cifra che viene incrementata quando avviene un rilascio, che nel caso del progetto corrisponde ai raggiungimenti di RTB[G], PB[G] e CA[G];
- **Y:** rappresenta un'aggiunta o modifica sostanziale, come ad esempio l'aggiunta di una sezione;
- **Z:** indica una piccola modifica, come per esempio la correzione di errori o aggiunte di piccole dimensioni.

Un documento parte sempre dalla versione 0.0.0.
Ogni modifica di un numero di versione, comporta l'azzeramento di tutti i numeri alla sua destra.

3.2.4 Repository[G]

Il gruppo NaN1fy utilizza le seguenti repository facenti parte della organizzazione NaN1fy in Github[G].

3.2.4.1 Lista repository

- **docs:** repository dedicata alla documentazione del progetto;
- **SyncCity:** repository dedicata alla scrittura e implementazione del progetto software;
- **NaN1fy.github.io:** repository del sito vetrina.

3.2.4.2 Struttura repository docs

Questo repository è suddiviso in due branch[G] principali:

- **main:** branch contenente tutti la documentazione prodotta in formato .pdf;
- **sources:** branch contenente i file .tex dei rispettivi documenti della repository precedente. Quando vengono aggiunti o modificati dei file in questa branch, vengono automaticamente compilati, caricando il risultato nella branch main, cosicchè possano essere visualizzati da chiunque.

Di seguito viene riportata la struttura della repository, i termini in **grassetto** indicano il nome di una directory:

- **PB:**
  - **Esterni:**
    - **Verbali**;
    - Analisi dei Requisiti (v2.0.0);
    - Manuale Utente (v1.0.0);
    - Piano di Progetto (v2.0.0);
    - Piano di Qualifica (v2.0.0);
    - Specifica Tecnica (v1.0.0);
    - Valutazione dell'utilizzo di Flink (v1.0.0).
  - **Interni:**
    - **Verbali**;
    - Glossario (v2.0.0);
    - Norme di Progetto (v2.0.0).
- **RTB:**
  - **Esterni:**
    - **Verbali**;
    - Analisi dei Requisiti (v1.0.0);
    - Piano di Progetto (v1.0.0);
    - Piano di Qualifica (v1.0.0).
  - **Interni:**
    - **Verbali**;
    - Glossario (v1.0.0);
    - Norme di Progetto (v1.0.0).
- **Candidatura:**
  - **Verbali:**
    - **Esterni**;
    - **Interni**;
  - Dichiarazione impegni (v2.0.0);
  - Lettera di presentazione;
  - Valutazione capitolati (v1.0.0).

3.2.4.3 Struttura Repository SyncCity

Di seguito viene riportata la struttura della repository, i termini in **grassetto** indicano il nome di una directory:

- **ClickHouse[G]:** directory contenente i config e codice necessario per l'utilizzo di ClickHouse;
- **Grafana[G]:** directory contenente i config di Grafana;
- **Flink[G]:** directory contenente i config e codice necessario per l'utilizzo di Flink;
- **PyMockSensors:** directory contenente il codice sorgente di PyMockSensors, ovvero il generatore simulato di dati provenienti da sensori;
- docker-compose.yaml: file di configurazione per l'utilizzo di Docker Compose[G].

3.2.4.4 Sito Vetrina

La repository NaN1fy.github.io (raggiungibile all'omonimo link) contiene il codice del sito del progetto, il cui scopo è quello di fornire una veloce e intuitiva interfaccia, nella quale poter visualizzare ergonomicamente e in maniera organizzata i documenti relativi al progetto stesso.
Il sito vetrina propone anche la visualizzazione diretta del *Glossario v2.0.0*, contenente una definizione chiara e univoca di tutti i termini rilevanti, onde evitare interpretazioni arbitrarie e fraintendimenti riguardo tali specifici concetti.
Il codice del sito comprende una funzionalità[G] di auto aggiornamento dei documenti contenuti nella vetrina e nel *Glossario v2.0.0*.

3.2.5 Sincronizzazione

Attraverso la piattaforma GitHub[G], ogni attività prevista viene affiancata da una issue[G] a cui corrisponde una branch[G], separata e parallela alle altre; tale suddivisione permette lo svolgimento in maniera autonoma e sicura di ogni singola attività, garantendo un'avanzamento simultaneo dei lavori.

3.2.5.1 Branching

La suddivisione in branch[G], già precedentemente esposta, presuppone l'utilizzo della metodologia single-purpose, che prevede l'utilizzo di una branch solo ed esclusivamente per lo svolgimento di una singola attività. In tale modo, il lavoro viene parcelizzato, garantendo un flusso stabile di lavoro.
Una volta che una data attività viene portata a termine, la corrispondente branch viene unita alla branch principale e successivamente cancellata.

3.2.5.2 Pull request[G]

Al termine di un'attività (o al completamento di una sua parte), il membro a cui è stata assegnata questa attività (o uno di essi), si assume la responsabilità di aprire una Pull Request indicando i verificatori. Questi ultimi, dopo aver confermato la correttezza delle modifiche, hanno il compito di fare il merge della pull request, chiudendo poi la issue.

3.3 Verifica

3.3.1 Introduzione

La verifica è un processo[G] svolto, caso per caso, da una parte dei membri del team, con lo scopo di garantire l'efficienza e la correttezza di ogni attività sottoposta a scrutinio.
È un processo presente per l'intera durata del ciclo di vita del software e abbraccia, nella sua esecuzione, sia la documentazione che il prodotto in sè: la sua attuazione, difatti, non è organizzata ricorrentemente, ma viene effetuata in occasione del completamento di un'attività, documento o parte significativa del prodotto.
Questo processo trae le sue basi dai vincoli di qualità[G] e dalle linee guida individuate all'interno del documento *Piano di Qualifica v2.0.0*, basi che il verificatore è tenuto a rispettare per garantire uniformità, coerenza e ripetibilità al processo di verifica.

3.3.2 Analisi statica

L'analisi statica è una modalità di analisi che prevede una verifica del prodotto e/o dei documenti senza la necessità di esecuzione o di mediazione di terze parti.
Le due metodologie principali per condurre l'analisi statica, sono: l'inspection e il walkthrough.

3.3.2.1 Inspection
Tale metodologia utilizza un'approccio metodologico e ben strutturato per rilevare possibili difetti nel prodotto e nella documentazione; questi difetti vengono definiti e specificati a priori in delle liste, cosidette di controllo, le quali verranno poi utilizzate in maniera programmatica nella valutazione del documento e codice del prodotto.

3.3.2.2 Walkthrough
Contrariamente al metodo inspection, il walkthrough si preoccupa della verifica del documento o del codice tramite un maggiore dialogo tra il verificatore e l'autore, eseguendo una verifica più sostanziale e profonda, non alla ricerca di problemi specifici, bensì di una correttezza più generica e adattiva. In tale maniera, pur risultando in un maggiore impiego di risorse, garantisce una correttezza maggiore.
Per questo progetto il team NaN1fy si propone di utilizzare per la documentazione un'approccio di questo tipo, principalmente considerando che l'approccio di tipo inspection è fin troppo rigido per poter garantire una corretta valutazione di un'insieme così eterogeneo e variegato di dati, quale è la documentazione del progetto.

3.3.3 Analisi dinamica

La metodologia analisi dinamica pone l'attenzione sul comportamento del codice e suoi eventuali difetti durante l'esecuzione dello stesso.
Tale metodologia è specifica per ogni progetto in sè, in quanto l'insieme di attività di verifica, composte da test[G], dipendono esclusivamente dal contenuto del codice e dai requisiti del progetto.
Tali test garantiscono, grazie alla loro ripetibilità, una valutazione oggettiva del codice e delle funzionalità del prodotto, in quanto in grado, dato un'insieme di caratteristiche, di generare lo stesso risultato più volte, indipendentemente da fattori esterni o casuali. La definizione e l'esecuzione dei test seguono i principi del Modello a V[G].

3.3.3.1 Test di unità
Test che hanno come obiettivo la verifica delle singole unità del sistema[G], che possono essere funzioni o metodi esulando dal resto del sistema.
I test di unità si dividono in due categorie:

- **Test funzionali:** verificano che l'output corrisposta al valore atteso;
- **Test strutturali:** verificano la struttura interna dell'unità e il flusso dei dati.

3.3.3.2 Test di integrazione
Test che hanno come obiettivo la valutazione del comportamento delle unità quando vengono combinate tra di loro identificando eventuali problemi dell'interazione tra le componenti integrate e verificando l'efficacia e il soddisfacimento dei requisiti posti dal progetto.

3.3.3.3 Test di sistema
Test che hanno come obiettivo di valutare il sistema come singola e unica entità verificando la corretta esecuzione e completezza del prodotto, in linea con i requisiti fissati nel documento di *Analisi dei Requisiti v2.0.0*.

3.3.3.4 Test di accettazione
Test il cui obiettivo è di mostrare la conformità del software rispetto le richieste e aspettative della Proponente[G], garantendo il soddisfacimento di quest'ultima riguardo il prodotto finale.

3.3.3.5 Identificazione dei test

Ogni singolo test possiede un codice univoco identificativo con il seguente formato:

T[Tipologia]-[Numero]

Dove **Tipologia** indica la tipologia del test:

- **U:** di unità;
- **I:** di integrazione;
- **S:** di sistema;
- **A:** di accettazione.

3.3.3.6 Stato dei test

I test possiedono anche uno stato:

- **V:** Verificato. Il test ha esito positivo;
- **NV:** Non Verificato. Il test ha esito negativo;
- **NI:** Non Implementato.

3.4 Gestione Qualità[G]

L'obiettivo è assicurarsi che i processi e il prodotto rispettino le esigenze del cliente e lo facciano con il massimo livello di qualità possibile, monitorando anche futuri progressi attraverso verifiche retrospettive.

3.4.1 Piano di Qualifica

Il documento *Piano di Qualifica v2.0.0* è fondamentale per il completamento degli obiettivi di questo processo[G].
Il documento comprende:

- Fissare gli obiettivi di qualità;
- Definire le metriche di visione quantitativa;
- Definire test di qualità e funzionamento e relativa documentazione;
- Avere una visione dello stato attuale del prodotto e del progetto;
- Fornire margine di retrospettiva ai fini di miglioramento.

3.4.2 Testing

Il documento *Piano di Qualifica v2.0.0* fornisce obiettivi di qualità sia del processo che del prodotto. Le metriche relative garantiscono la verifica sugli aspetti di accessibilità; i test invece garantiscono la qualità generale del software.
Le categorie interessate sono le seguenti:

- **Test di unità:** si verifica il corretto funzionamento delle unità componenti il sistema[G]. Un'unità rappresenta un elemento indivisibile e indipendente del sistema[G];
- **Test di integrazione:** si verifica il corretto funzionamento di più unità che cooperano per svolgere uno specifico compito (tali unità devono certamente aver superato i loro test di unità precedentemente);
- **Test di sistema[G]:** si verifica il corretto funzionamento del sistema[G] nella sua interezza. I requisiti funzionali obbligatori, di vincolo, di qualità[G] e di prestazione, precedentemente concordati con la Proponente[G] mediante stipulazione del contratto, devono essere soddisfatti per intero;
- **Test di accettazione:** si verifica il soddisfacimento della Proponente rispetto al prodotto software. Il loro superamento permette di procedere con il rilascio del prodotto.

Per ogni test vengono indicati i risultati del test di qualità del processo e della qualità del prodotto software.

3.4.3 Metriche

Il documento *Piano di Qualifica v2.0.0* fornisce le metriche da applicare all'esecuzione dei test di qualità. Ogni metrica è codificata come segue:

M[Tipologia metrica]-[Sigla identificativa Metrica]

in particolar modo:

- **Tipologia metrica:** l'effettiva categoria di appartenenza della metrica[G];
  - **PC:** Processo;
  - **PD:** Prodotto.
- **Sigla identificativa metrica:** sigla di indentificazione della specifica metrica.

3.4.3.1 Gestione della qualità[G]

Metrica	Sigla	Descrizione	Formula
MPC-QMS	Quality Metrics Satisfied (QMS)	Numero di metriche di qualità soddisfatte.	-

Tabella 3: Valori accettabili e ottimi per ogni metrica riguardante il processo di gestione della qualità.

3.4.4 Aspettative

A seguito di questo processo, le aspettative attese sono le seguenti:

- Ottima qualità del prodotto realizzato;
- Ottima qualità dei processi di sviluppo e di comunicazione;
- Buona visione quantitativa dello stato di avanzamento dei lavori;
- Test con alta frequenza e predicibili;
- Miglioramento generale costante;
- Soddisfazione delle aspettative del Proponente[G].

4 Processi Organizzativi

4.1 Gestione di Processo

4.1.1 Coordinamento

4.1.1.1 Comunicazioni interne

Le comunicazioni interne avvengono principalmente tra membri del gruppo di pari livello. Si rimanda alla sezione 4.2 per approfondire gli strumenti utilizzati.

- **Discord[G]**
  - **Informazioni:** comunicazioni semi-formali per la condivisione di risorse;
  - **Canali Testuali:** comunicazioni semi-formali per la condivisione di appunti sugli incontri;
  - **Canali Vocali:** comunicazioni semi-formali per le riunioni interne; informali per argomenti non inerenti al progetto.
- **Telegram[G]**
  - **Chat di gruppo:** comunicazioni semi-formali, brevi, inerenti al progetto;
  - **Chat individuali:** comunicazioni informali, inerenti al progetto.
- **Google Calendar**
  - Comunicazioni informali, brevi, inerenti all'evento di calendario relativo.

4.1.1.2 Comunicazioni esterne

Le comunicazioni esterne con il Committente[G] e Proponente[G] vengono considerate ovviamente di importanza maggiore, dunque trattate col rispetto dovuto.
Il registro utilizzato è esclusivamente formale, cercando di adottare vocaboli consoni e concisi.

- Discord;
- Google Meet;
- Gmail: indirizzo e-mail condiviso nan1fyteam.unipd@gmail.com.

4.1.1.3 Riunioni interne

Le riunioni interne avvengono tra i membri del gruppo, settimanalmente, con cadenza al giovedì ore 15 esclusi casi eccezionali, con durata media di 1 ora / 1 ora e mezza. Dalla fine dell'RTB le riunioni interne sono state spostate, di conseguenza alla variazione delle date del SAL con la **Proponente**, al martedì, con orario invariato.
In caso di mancata partecipazione da parte di uno dei membri, oltre alla sezione di appunti presente nei canali di comunicazione, è comunque possibile accedere ai verbali degli incontri.
I verbali vengono redatti dai membri a rotazione, così come la verifica di esse.
Ogni riunione ha un tema principale, "l'ordine del giorno", su cui implicitamente si basano le discussioni principali. Oltre a ciò, generalmente le riunioni avvengono nella seguente modalità:

- Discussione da parte di ognuno sulle proprie task[G] assegnate con retrospettiva in merito (quel che si è fatto e quel che c'è da fare);
- Discussione su temi venuti fuori durante lo svolgimento delle task;
- Controllo generale della task board[G] e delle scadenze imminenti.

4.1.1.4 Riunioni esterne

Le riunioni esterne avvengono prevalentemente tra committente[G] e Proponente[G] ogni due settimane. Vengono utilizzate come incontri di SAL[G], e come deadline degli Sprint[G].
La durata media è intorno ai 40 minuti, ed è compito dei membri del gruppo esporre in modo conciso e preparato lo stato di avanzamento dei lavori ed eventuali dubbi sorti.
Vengono successivamente redatti i relativi verbali, consegnati all'azienda per poter essere validati, approvati e firmati.

4.1.1.5 Reperibilità

Ciascun componente del gruppo gode dell'autonomia di pianificare il proprio orario di lavoro individuale in modo asincrono, conformemente agli obblighi accademici, personali e alle disposizioni stabilite nel programma preventivamente concordato. In un accordo mirato a bilanciare l'efficacia della comunicazione asincrona con la tutela del tempo personale, i partecipanti si impegnano a garantire la propria disponibilità per questioni inerenti al progetto didattico durante il seguente intervallo orario: dal lunedì al venerdì, dalle 9:00 alle 13:00 e dalle 15:00 alle 19:00. Qualsiasi modifica agli orari o ai giorni di disponibilità può essere concordata previamente tra i membri del gruppo. È importante sottolineare che questo intervallo di disponibilità non deve essere interpretato come tempo di lavoro attivo, ma un limite temporale entro il quale i membri si impegnano a essere raggiungibili per eventuali necessità connesse al progetto.

4.1.2 Pianificazione

4.1.2.1 Ruoli del Progetto

I ruoli sono stati distribuiti equamente tra tutti i componenti del gruppo e saranno:

- **Responsabile del progetto:** Guida il team nel rispetto delle scadenze, nell'allocazione delle risorse e nella pianificazione generale, assicurando che il progetto proceda in modo efficiente e soddisfi gli obiettivi.
  - Pianifica lo Sprint[G] definendo le task relative, definendo il preventivo ore e costi;
  - Calcola il consuntivo[G] delle ore e costi alla fine dello Sprint;
  - Tiene traccia dello stato generale di progresso dell'intero progetto;
  - Fa da intermediario tra il gruppo e l'azienda Proponente[G].
- **Amministratore:** Gestisce l'infrastruttura e le risorse necessarie per il progetto, inclusi gli strumenti e le tecnologie che definiscono il modo di lavorare del team.
  - Gestisce l'infrastruttura del progetto e i suoi strumenti;
  - Automatizza i processi e ne individua punti di miglioramento;
  - Si occupa dell'effettiva redazione dei documenti che definiscono il Way of Working[G] del gruppo.
- **Programmatore:** Responsabile della scrittura del codice seguendo le specifiche del progetto e traducendo i requisiti in un'applicazione funzionante.
  - Si occupa della stesura del codice in conformità ai requisiti e alla sua manutenibilità;
  - Scrive i test[G] per il codice prodotto;
  - Si occupa della documentazione relativa alla comprensione del codice, sia da parte dell'utente che da parte del programmatore.
- **Progettista:** Definisce l'architettura del software, pianifica la sua struttura e l'organizzazione nel dettaglio.
  - Sviluppa l'architettura in conformità ai requisiti e alla sua manutenibilità, al minimo livello di dipendenze possibili;
  - Approfondisce le conoscenze e strumenti tecnici utili allo sviluppo.
- **Verificatore:** Garantisce la qualità[G] del software eseguendo test e controlli per assicurare il corretto funzionamento e il rispetto degli standard[G] di qualità.
  - Verifica il livello atteso e il rispetto della qualità in ambito tecnico;
  - Verifica il livello atteso e il rispetto della qualità in ambito funzionale;
  - Verifica il livello atteso e il rispetto della qualità in ambito organizzativo.
- **Analista:** Si concentra sull'analisi dei requisiti, aiuta a definire le funzionalità[G] del software e si assicura di comprendere i bisogni del cliente.
  - Valuta il dominio applicativo delle richieste del Proponente[G];
  - Scompone le richieste ed esigenze del Proponente in sotto attività così da poter essere risolte individualmente e/o parallelamente.

4.1.2.2 Gestione delle task[G]

Durante la fase di Sprint[G] Planning[G] vengono definite tutte le attività, le quali saranno poi associate alle relative task.
Ognuna di esse viene assegnata ad almeno un membro, il quale si occuperà del suo ciclo di vita. Vengono assegnate in modo da poter essere svolte il più parallelamente e asincronamente possibile.
Per le task di processi primari (sviluppo) e di supporto (documentazione) viene utilizzato GitHub[G]. Il ciclo di vita di una task è come segue:

- **Creazione:** la task definita viene aperta come issue[G] su GitHub;
- **Assegnazione:** la task viene assegnata ad uno o più membri del gruppo;
- **Completamento:** la task viene completata, prevalentemente su un branch[G] distinto dal principale;
- **Pull request:** viene fatta una pull request[G] del branch[G] di sviluppo dell'attività completata, collegando la richiesta alla relativa issue[G];
- **Verifica:** almeno due verificatori effettuano il controllo qualità[G];
- **Accettazione:** quando la fase di verifica è conclusa senza intoppi, viene approvata la pull request, si chiude la issue relativa e viene cancellato il branch relativo all'attività.

Per ulteriori dettagli si rimanda alla sezione 4.2. La dimensione e importanza dell'attività dipendono dal processo[G] (primario, di supporto o organizzativo) di cui fa parte. All'apertura della sua task[G] relativa viene valutato il tempo ragionevole di svolgimento e La tracciabilità dei cambiamenti, eventuali commenti e il generale stato dell'attività sono presenti nella piattaforma, nelle pagine della issue collegata.

4.1.2.3 Metodo di Lavoro

Per lo svolgimento dell'attività il gruppo ha scelto di adottare la modalità Agile SCRUM[G].
Ciò permette la suddivisione del tempo di lavoro in intervalli di tempo (Sprint[G]), in modo da definire attività da svolgere, quel che è stato fatto e si è fatto, e avere scadenze tangibili durante lo svolgimento del progetto. Ogni Sprint, della durata media di almeno due settimane, prevede le seguente fasi:

- **Sprint Planning:** Pianificazione dello Sprint, in concomitanza con il suo inizio. Gli incontri di SAL[G] vengono utilizzati come punto di riferimento di conclusione dello Sprint e inizio del successivo.
  - Discussione sui nuovi obiettivi post incontro di SAL da completare;
  - Discussione di quel che rimane da fare dal precedente Sprint e con quale priorità;
  - Definizione di obiettivi concreti e di issue[G] relative, definendole su GitHub[G];
  - Preventivo ore da parte di ogni membro, in base alle attività da svolgere e i ruoli da assegnare, e preventivo dei costi in base alle ore dichiarate.
- **Sprint Review:** Revisione dello Sprint nel suo ultimo giorno.
  - Consuntivo[G] ore secondo la produttività individuale, sia in difetto che in eccesso, e consuntivo costi in base alle ore dichiarate;
  - Discussione sugli obiettivi raggiunti e non di ogni membro. Gli obiettivi non raggiunti verranno presentati allo Sprint successivo.
- **Sprint Retrospective:** Retrospettiva effettivamente conclusiva dello Sprint, valutandone l'andamento generale. Viene valutato ciò che è andato positivamente e cosa negativamente, così da poter avere delle linee guida sul come proseguire al meglio;
  - **Good:** ciò che effettivamente è andato bene durante lo svolgimento dello Sprint;
  - **To Improve:** ciò che invece va migliorato per i Sprint successivi.

4.2 Infrastruttura

Fanno parte dell'infrastruttura organizzativa tutti gli strumenti che permettono al gruppo di attuare in modo efficace ed efficiente i processi organizzativi. In particolare tali strumenti permettono la **comunicazione**, il **coordinamento** e la **pianificazione**.

4.2.1 Strumenti

4.2.1.1 GitHub[G]

È il principale servizio di hosting della repository[G] di gruppo e di controllo della versione distribuita.

Generalmente il workflow adottato dal gruppo è il GitHub Flow, che sinteticamente segue il seguente schema:

- Riallineamento della repository locale con quella remota;
- Creazione di un branch[G] locale su cui effettuare le modifiche;
- Push del branch locale verso repository remota;
- Creazione di una pull request[G];
- Verifica da parte di due membri del gruppo e successivo merge del branch con le modifiche;
- Eliminazione del branch utilizzato dalla repository remota.

Per i dettagli consultare la documentazione ufficiale:

https://docs.github.com/en/get-started/quickstart/github-flow (ultimo accesso: 26 agosto 2024)

Viene utilizzato anche il sistema[G] di **project management** fornito.

La **board[G] principale** è divisa nelle seguenti liste:

- **Todo:** contiene task[G] da fare in base allo Sprint[G] corrente. Durante la fase di Sprint Planning[G] sono state definite ed ognuna preassegnata ad almeno un membro, il quale si occuperà del suo ciclo di vita;
- **In Progress:** contiene task in corso. Ogni task che non sia relativa alla stesura dei verbali viene assegnata ad almeno due membri, così che possano lavorarci asincronamente;
- **Verify:** contiene task completate che necessitano di verifica. Qualsiasi membro del gruppo può autonomamente diventare verificatore di una task aggiungendosi a essa, con la precondizione che non sia la stessa persona che l'ha svolta. Questa fase avviene in parallelo con la relativa Pull Request[G], dove ulteriori due membri devono verificare ed approvare lo svolgimento della task. Nel caso di modifiche lievi sarà discrezione del verificatore apportarle; nel caso di modifiche più importanti sarà premura del Responsabile valutare la situazione.
  In generale, in caso di esito negativo la pull request viene annullata e si ritorna in fase "In Progress". Nel caso invece sia conforme viene approvata e spostata in fase "Done";
- **Done:** contiene task completate, verificate e accettate. In prossimità del prossimo Sprint, vengono archiviate le task completate relative a Sprint precedenti così da non Sprint la board[G].

Per quanto riguarda le **task**, ognuna è costituita da:

- **Titolo:** breve descrizione sintentica su cosa consiste la task;
- **Descrizione:** opzionale e breve per dettagli importanti;
- **Membro/i:** elementi del gruppo coinvolti nella task;
- **Milestone[G]:** a quale milestone fanno riferimento.

Si vuole far notare che, nonostante tutto, le task non vengono mai cancellate ma soltanto archiviate.
Per navigare più facilmente nella bacheca è possibile impostare dei filtri, ad esempio per membro o milestone.

4.2.1.2 Discord[G]
Principale strumento di **comunicazione interna sincrona** e **asincrona**. Vengono utilizzati 3 categorie di canali:

- **Informazioni:** prevalentemente utilizzato per la condivisione di risorse;
- **Canali Testuali:** utilizzato per la condivisione di appunti su incontri interni ed esterni, e simili;
- **Canali Vocali:** comunicazioni vocali tra i membri del gruppo, con possibilità di condivisione schermo.

Viene utilizzato questo strumento anche per la **comunicazione asincrona** con l'azienda Proponente[G] nel loro canale personale.

4.2.1.3 Telegram[G]
Principale strumento di **comunicazione interna testuale asincrona**. Viene utilizzato in due modalità:

- **Gruppo:** chat condivisa utilizzata, con parsimonia, per comunicazioni rivolte a tutti i membri;
- **Individuale:** ogni membro del gruppo può essere contattato singolarmente.

4.2.1.4 Google Calendar
Calendario condiviso del gruppo utilizzato per tenere traccia di:

- **Meeting Esterni:** con Proponente[G] o Committente[G];
- Qualsiasi altra attività o evento che può essere collocato in un tempo specifico.

4.2.1.5 Google Drive
Strumento utilizzato come:

- **Directory condivisa** dai membri del gruppo per documenti temporanei o non ufficiali;
- Accesso alla **suite Google:** Docs, Sheets, Slides.

4.2.1.6 Google Meet
Strumento di videochiamata utilizzato principalmente per la comunicazione esterna con committente e Proponente[G].

4.2.1.7 Google Mail
Utilizzo dell'indirizzo e-mail condiviso nan1fyteam.unipd@gmail.com per le **comunicazione esterna** come gruppo con i proponenti e il committente.

4.3 Miglioramento

Nel corso della redazione della documentazione e dello sviluppo del software, il gruppo si impegnerà a perseguire un miglioramento costante delle attività, con l'obiettivo di evitare di ripetere errori precedentemente commessi e di fornire soluzioni ottimali, con un riguardo particolari sui temi quali:

- Organizzazione;
- Ruoli;
- Strumenti di lavoro.

4.4 Formazione

Al fine di promuovere un ambiente di lavoro asincrono efficiente e equo, garantendo un progresso organizzato e uniforme senza lasciare alcun membro indietro, si richiede a ciascun componente del gruppo di assumersi autonomamente la responsabilità di colmare eventuali lacune relative agli strumenti e alle tecnologie impiegate per la documentazione e lo sviluppo del progetto. Questo può avvenire mediante lo studio individuale o, alternativamente, con la condivisione delle proprie conoscenze con gli altri membri al fine di accelerare il processo[G] di apprendimento.
Di seguito sono elencati gli strumenti e le tecnologie utilizzati, insieme ai principali riferimenti adottati dal gruppo:

- **LaTeX:** https://www.overleaf.com/learn (ultimo accesso: 26 agosto 2024);
- **Git[G]:** https://docs.github.com/en/get-started/using-git/about-git (ultimo accesso: 26 agosto 2024);
- **GitHub[G]:** https://docs.github.com (ultimo accesso: 26 agosto 2024);
- **GitHub Flow:** https://docs.github.com/en/get-started/quickstart/github-flow (ultimo accesso: 26 agosto 2024).

5 Metriche per il prodotto

5.1 Introduzione

Al fine di garantire una valutazione oggettiva e misurabile del prodotto e dei processi che lo compongono rispetto agli standard[G] fissati, oltre che per fornire una guida atta al miglioramento del prodotto in sè e dei suoi processi, verrano utilizzate le seguenti metriche di product management.

5.2 Metriche per la qualità di processo[G]

- **Fornitura**
  - **MPC-EV:** Earned Value - Rappresenta il valore del lavoro prodotto fino ad un dato momento;

$$EV = BAC^{G} \times \%lavoro\_prodotto \tag{1}$$

  - **MPC-PV:** Planned Value - Rappresenta il valore del lavoro pianificato fino ad un dato momento;

$$PV = BAC^{G} \times \%lavoro\_pianificato \tag{2}$$

  - **MPC-AC:** Actual Cost - Rappresenta il costo effettivamente sostenuto fino ad un dato momento. Il suo valore è reperibile nel *Piano di Progetto v2.0.0*;
  - **MPC-CPI:** Cost Performance Index - Rappresenta l'indice di produzione rispetto al costo sostenuto;

$$CPI = \frac{EV}{AC} \tag{3}$$

  - **MPC-EAC:** Estimate At Completion - Rappresenta il valore stimato per il completamento del progetto in un dato momento;

$$EAC = \frac{BAC^{G}}{CPI} \tag{4}$$

  - **MPC-ETC:** Estimate To Completion - Rappresenta il valore stimato per completare il progetto;

$$ETC = EAC - AC \tag{5}$$

  - **MPC-VAC:** Variance At Completion - Rappresenta la variazione relativa del budget stimato rispetto al budget pianificato;

$$VAC = \frac{BAC - EAC}{BAC} \tag{6}$$

  - **MPC-SV:** Schedule Variance - Rappresenta la variazione relativa del valore prodotto rispetto a quello pianificato;

$$SV = \frac{EV - PV}{BAC} \tag{7}$$

  - **MPC-BV:** Budget Variance - Rappresenta la variazione relativa tra il valore pianificato e i costi sostenuti.

$$BV = \frac{PV - AC}{BAC} \tag{8}$$

- **Sviluppo**
  - **MPC-RSI:** Requirements Stability Index - Rappresenta l'inclinazione del progetto di subire modifiche alla quantità di requisiti, espresso in percentuale.
- **Documentazione**
  - **MPC-IG:** Indice Gulpease - Indice di leggibilità di un testo tarato sulla lingua italiana, considera la lunghezza delle parole e della frase rispetto al numero di lettere nella frase stessa.

$$GULPEASE = 89 + \left(\frac{300 \times \text{numero\_frasi} - 10 \times \text{numero\_lettere}}{\text{numero\_parole}}\right) \tag{9}$$

    Il risultato è un numero compreso tra 0 e 100, dove "100" indica la leggibilità più alta e "0" quella più bassa. Vi sono alcune soglie che sono ufficialmente considerate rilevanti, in particolare:
    * Testi con risultato inferiore all'80 sono considerati di difficile lettura per chi possiede la licenza elementare;
    * Testi con risultato inferiore all'60 sono considerati di difficile lettura per chi possiede la licenza media;
    * Testi con risultato inferiore all'40 sono considerati di difficile lettura per chi possiede la licenza superiore.
  - **MPC-CO:** Correttezza Ortografica - Numero di errori ortografici e grammaticali presenti nel documento.
- **Verifica**
  - **MPC-CC:** Code Coverage - Rappresenta la percentuale di codice eseguito durante l'esecuzione di un cluster di test.
- **Gestione della qualità**
  - **MPC-QMS:** Quality Metrics Satisfied - Rappresenta il numero di metriche che sono state soddisfatte in un determinato momento.
- **Gestione dei processi**
  - **MPC-NR:** Non-calculated Risks - Rappresenta il numero di rischi non calcolati incontrati fino ad un dato momento.

5.3 Metriche per la qualità di prodotto

- **Funzionalità:**
  - **MPD-ROS:** Requisiti Obbligatori Soddisfatti - Rappresenta la percentuale di requisiti obbligatori soddisfatti;
  - **MPD-RDS:** Requisiti Desiderabili Soddisfatti - Rappresenta la percentuale di requisiti desiderabili soddisfatti;
  - **MPD-ROPS:** Requisiti Opzionali Soddisfatti - Rappresenta la percentuale di requisiti opzionali soddisfatti.
- **Affidabilità:**
  - **MPD-BC:** Branch Coverage - Rappresenta la percentuale di percorsi possibili eseguiti almeno una volta nel codice;
  - **MPD-SC:** Statement Coverage - Rappresenta la percentuale di istruzioni eseguite nel codice;
  - **MPD-FD:** Failure Density - Rappresenta il rapporto tra la quantità di guasti già presenti nel codice rispetto alla quantità presente all'inserimento di nuovo codice;
  - **MPD-PTCP:** Passed Test Cases Percentage - Rappresenta la percentuale di test eseguiti con esito positivo.

- **Usabilità:**
  - **MPD-FU:** Facilità di Utilizzo - Rappresenta il numero medio di click necessario per raggiungere ogni funzionalità;
  - **MPD-TA:** Tempo di Apprendimento - Rappresenta il numero di minuti necessario all'apprendimento del prodotto.
- **Efficienza:**
  - **MPD-UR:** Utilizzo Risorse - Rappresenta la quantità di risorse utilizzate per l'esecuzione del Software (CPU, RAM, TE).
- **Manutenibilità:**
  - **MPD-CC:** Complessità Ciclomatica - Rappresenta la complessità di programma determinata dal numero di cammini linearmente indipendenti attraversabili.