Norme di Progetto

SnakeByte (Gruppo 1):
Valeria Baleanu, Leonardo Pellizzon, Filippo Venzo, Giuseppe De Fina,
Francesco Pasqual, Christian Libralato, Luca Granziero
(2109911, 2111006, 2113705, 2113187, 2103119, 2101047, 2075512)

Informazioni documento			
Versione	**Data**	**Stato**	**Destinatari**
2.0.0	13/04/2026	Approvato	SnakeByte, prof. Vardanega Tullio, prof. Cardin Riccardo

Contatti: snakebyteteam@gmail.com

Registro delle modifiche					
Versione	**Data**	**Autore**	**Verifica**	**Approvazione**	**Descrizione**
2.0.0	05/04/2026	-	-	C. Libralato	Approvazione
1.2.0	04/04/2026	V. Baleanu	G. De Fina	-	Ristrutturazione uniforme di tutti i processi.
1.1.0	31/03/2026	L. Granziero	G. De Fina	-	Aggiunta sottosezioni in Fornitura e Sviluppo, sezione Codifica, controllo della configurazione e valutazione.
1.0.0	13/02/2026	-	-	F. Venzo	Approvazione
0.9.0	10/02/2026	G. De Fina	L. Granziero F. Pasqual	-	Aggiunto elenco delle tabelle.
0.8.0	09/02/2026	G. De Fina	L. Granziero F. Pasqual	-	Aggiunte "generalizzazioni" nella tabella per l'Analisi dei Requisiti, aggiunto comando *needspace*.
0.7.0	08/02/2026	G. De Fina	V. Baleanu	-	Modificata redazione verbali (responsabile), aggiunta sezione Gestione e tracciamento dei test, distinzione test funzionali/strutturali, aggiornamento strumento comunicazioni esterne a Microsoft Teams.
0.6.0	05/2/2026	G. De Fina	V. Baleanu	-	Completati Processi di Supporto con Garanzia della qualità, Revisioni congiunte, Audit e Risoluzione dei problemi.
0.5.0	05/02/2026	G. De Fina	V. Baleanu	-	Completati Processi Organizzativi con sezioni su Controllo e monitoraggio, Gestione sprint, Tracciamento attività e Gestione rischi.
0.4.0	11/01/2026	V. Baleanu	F. Venzo	-	Aggiunta processo di Verifica e Validazione.
0.3.0	01/01/2026	V. Baleanu	F. Venzo	-	Aggiunta metriche di qualità di processo e di prodotto.
0.2.0	08/12/2025	C. Libralato	L. Pellizzon	-	Aggiunte a processo di verifica e approvazione dei documenti (sez. 3.1.4, 3.1.5). Aggiunta Sviluppo con Analisi Requisiti ai processi primari (sez. 2.2)
0.1.6	29/11/2025	F. Pasqual	V. Baleanu	-	Aggiunto processo di verifica dei documenti tramite *pull request*

Versione	Data	Autore	Verifica	Approvazione	Descrizione
0.1.5	08/11/2025	F. Venzo	L. Granziero	-	Aggiunta tabella attività completate nella struttura dei verbali
0.1.4	30/10/2025	F. Venzo	L. Granziero	-	Modifica struttura tabella attività, modifiche tipografiche
0.1.3	26/10/2025	F. Venzo	L. Granziero	-	Aggiunta sezioni e termini glossario, modifica convenzione date nei nomi dei file
0.1.2	23/10/2025	F. Venzo	L. Granziero	-	Correzione errori ortografici e aggiunta link
0.1.1	22/10/2025	F. Venzo	L. Granziero	-	Aggiunta sezioni e sotto sezioni
0.1.0	17/10/2025	F. Venzo	L. Granziero	-	Prima stesura

Indice

1 Introduzione **7**
1.1 Finalità del documento . . . 7
1.2 Glossario . . . 7
1.3 Scopo del Prodotto . . . 7
1.4 Riferimenti Normativi . . . 7
1.5 Riferimenti Informativi . . . 7

2 Processi primari **8**
2.1 Fornitura . . . 8
2.1.1 Attività . . . 8
2.1.2 Procedure . . . 8
2.1.2.1 Documentazione risultante . . . 8
2.1.3 Strumenti . . . 9
2.2 Sviluppo . . . 9
2.2.1 Attività . . . 9
2.2.2 Procedure . . . 9
2.2.2.1 Analisi dei Requisiti . . . 9
2.2.2.2 Documentazione risultante . . . 10
2.2.2.3 Codifica . . . 10
2.2.3 Strumenti . . . 11

3 Processi di supporto **11**
3.1 Documentazione . . . 11
3.1.1 Attività . . . 11
3.1.2 Procedure . . . 11
3.1.2.1 Struttura generale dei documenti . . . 11
3.1.2.2 Struttura dei verbali . . . 12
3.1.2.3 Tabella delle decisioni . . . 12
3.1.2.4 Tabella delle attività da svolgere . . . 12
3.1.2.5 Tabella delle attività svolte . . . 12
3.1.2.6 Struttura dei diari di bordo . . . 12
3.1.2.7 Redazione dei documenti . . . 13
3.1.2.8 Verifica dei documenti . . . 13
3.1.2.9 Approvazione dei documenti . . . 13
3.1.2.10 Manutenzione dei documenti . . . 14
3.1.2.11 Nomenclatura dei documenti . . . 14
3.1.3 Strumenti . . . 14
3.2 Gestione della configurazione . . . 14
3.2.1 Attività . . . 14
3.2.2 Procedure . . . 15
3.2.2.1 Numeri di versione dei documenti . . . 15
3.2.2.2 Controllo della configurazione . . . 15
3.2.2.3 Valutazione della configurazione . . . 15
3.2.2.4 Repository . . . 15
3.2.3 Strumenti . . . 15
3.3 Verifica . . . 15
3.3.1 Attività . . . 16
3.3.2 Procedure . . . 16
3.3.2.1 Analisi statica . . . 16
3.3.2.2 Analisi dinamica . . . 16
3.3.2.3 Nomenclatura dei test . . . 16
3.3.2.4 Stato dei test . . . 17
3.3.2.5 Gestione e tracciamento dei test . . . 17

3.3.3 Strumenti . . . 17
3.4 Validazione . . . 17
3.4.1 Attività . . . 17
3.4.2 Procedure . . . 17
3.4.2.1 Collaudo . . . 17
3.4.3 Strumenti . . . 18
3.5 Garanzia della qualità . . . 18
3.5.1 Attività . . . 18
3.5.2 Procedure . . . 18
3.5.2.1 Responsabilità del Verificatore . . . 18
3.5.2.2 Metriche di qualità . . . 18
3.5.3 Strumenti . . . 18
3.6 Revisioni congiunte . . . 18
3.6.1 Attività . . . 18
3.6.2 Procedure . . . 19
3.6.2.1 SAL – Stato Avanzamento Lavori . . . 19
3.6.3 Strumenti . . . 19
3.7 Audit . . . 19
3.7.1 Attività . . . 19
3.7.2 Procedure . . . 19
3.7.2.1 Audit di baseline . . . 19
3.7.3 Strumenti . . . 19
3.8 Risoluzione dei problemi . . . 20
3.8.1 Attività . . . 20
3.8.2 Procedure . . . 20
3.8.2.1 Workflow di risoluzione . . . 20
3.8.3 Strumenti . . . 20

4 Processi organizzativi 20
4.1 Processo di gestione . . . 20
4.1.1 Attività . . . 20
4.1.2 Procedure . . . 21
4.1.2.1 Assegnazione delle responsabilità . . . 21
4.1.2.2 Responsabile . . . 21
4.1.2.3 Amministratore . . . 21
4.1.2.4 Progettista . . . 21
4.1.2.5 Analista . . . 21
4.1.2.6 Verificatore . . . 22
4.1.2.7 Programmatore . . . 22
4.1.2.8 Coordinamento . . . 22
4.1.2.9 Gestione degli sprint . . . 22
4.1.2.10 Tracciamento delle attività . . . 22
4.1.2.11 Pianificazione e monitoraggio delle ore . . . 23
4.1.2.12 Gestione dei rischi . . . 23
4.1.3 Strumenti . . . 23
4.2 Processo di miglioramento . . . 23
4.2.1 Attività . . . 23
4.2.2 Procedure . . . 24
4.2.2.1 Ciclo di miglioramento . . . 24
4.2.3 Strumenti . . . 24
4.3 Processo di formazione . . . 24
4.3.1 Attività . . . 24
4.3.2 Procedure . . . 24
4.3.2.1 Tecnologie da apprendere . . . 24
4.3.2.2 Risorse per la formazione . . . 25

4.3.2.3 Piano per la formazione . . . 25
4.3.3 Strumenti . . . 25

5 Metriche di qualità di processo **25**
5.1 Processi primari . . . 25
5.1.1 Fornitura . . . 25
5.1.1.1 MPC1 - Planned Value (PV) . . . 25
5.1.1.2 MPC2 - Earned Value (EV) . . . 26
5.1.1.3 MPC3 - Actual Cost (AC) . . . 26
5.1.1.4 MPC4 - Cost Performance Index (CPI) . . . 26
5.1.1.5 MPC5 - Schedule Performance Index (SPI) . . . 27
5.1.1.6 MPC6 - Estimate At Completion (EAC) . . . 27
5.1.1.7 MPC7 - Estimate To Complete (ETC) . . . 27
5.1.2 Sviluppo . . . 27
5.1.2.1 MPC8 - Requirements Stability Index (RSI) . . . 27
5.2 Processi di supporto . . . 28
5.2.1 Documentazione . . . 28
5.2.1.1 MPC9 - Indice di Gulpease (IG) . . . 28
5.2.1.2 MPC10 - Indice di Frammentazione (IF) . . . 28
5.2.2 Verifica . . . 29
5.2.2.1 MPC11 - Test Success Rate (TSR) . . . 29
5.2.3 Gestione della qualità . . . 29
5.2.3.1 MPC12 - Percentuale Metriche Soddisfatte (PMS) . . . 29
5.3 Processi organizzativi . . . 30
5.3.1 Gestione dei processi . . . 30
5.3.1.1 MPC13 - Percentuale Rischi Inattesi (PRI) . . . 30
5.3.1.2 MPC14 - Labor Efficiency (LE) . . . 30

6 Metriche di qualità di prodotto **31**
6.1 Funzionalità . . . 31
6.1.0.1 MPD1 – Percentuale Requisiti Obbligatori Soddisfatti (PROS) . . . 31
6.1.0.2 MPD2 – Percentuale Requisiti Opzionali Soddisfatti (PRPS) . . . 31
6.1.0.3 MPD3 – Percentuale Requisiti Desiderabili Soddisfatti (PRDS) . . . 32
6.2 Affidabilità . . . 32
6.2.0.1 MPD4 - Line Coverage (LC) . . . 32
6.2.0.2 MPD5 - Branch Coverage (BC) . . . 32
6.2.0.3 MPD6 - Statement Coverage (SC) . . . 33
6.3 Usabilità . . . 33
6.3.0.1 MPD7 - Profondità Massima di Navigazione (PMN) . . . 33
6.4 Efficienza . . . 34
6.4.0.1 MPD8 - Tempo Medio di Risposta (TMR) . . . 34
6.5 Manutenibilità . . . 34
6.5.0.1 MPD9 - Complessità Ciclomatica (CC) . . . 34
6.6 Portabilità . . . 35
6.6.0.1 MPD10 - Compatibilità Cross-Browser (CCB) . . . 35

Elenco delle tabelle

1 Struttura dei casi d'uso . . . 10
2 Acronimi dei documenti . . . 14
3 MPC1 - Planned Value (PV) . . . 25
4 MPC2 - Earned Value (EV) . . . 26
5 MPC3 - Actual Cost (AC) . . . 26
6 MPC4 - Cost Performance Index (CPI) . . . 26
7 MPC5 - Schedule Performance Index (SPI) . . . 27
8 MPC6 - Estimate At Completion (EAC) . . . 27
9 MPC7 - Estimate To Complete (ETC) . . . 27
10 MPC8 - Requirements Stability Index (RSI) . . . 28
11 MPC9 - Indice di Gulpease (IG) . . . 28
12 MPC10 - Indice di Frammentazione (IF) . . . 29
13 MPC11 - Test Success Rate (TSR) . . . 29
14 MPC12 - Percentuale Metriche Soddisfatte (PMS) . . . 30
15 MPC13 - Percentuale Rischi Inattesi (PRI) . . . 30
16 MPC14 - Labor Efficiency (LE) . . . 31
17 MPD1 - Percentuale Requisiti Obbligatori Soddisfatti (PROS) . . . 31
18 MPD2 - Percentuale Requisiti Opzionali Soddisfatti (PRPS) . . . 32
19 MPD3 - Percentuale Requisiti Desiderabili Soddisfatti (PRDS) . . . 32
20 MPD4 - Line Coverage (LC) . . . 32
21 MPD5 - Branch Coverage (BC) . . . 33
22 MPD6 - Statement Coverage (SC) . . . 33
23 MPD7 - Profondità Massima di Navigazione (PMN) . . . 34
24 MPD8 - Tempo Medio di Risposta (TMR) . . . 34
25 MPD9 - Complessità Ciclomatica (CC) . . . 35
26 MPD10 - Compatibilità Cross-Browser (CCB) . . . 35

1 Introduzione

1.1 Finalità del documento

Il presente documento fissa le norme che il gruppo *SnakeByte* è tenuto a rispettare durante l'intero ciclo di vita del progetto didattico, al fine di garantire efficienza ed efficacia nello svolgimento delle attività. Il documento è strutturato secondo lo Standard ISO/IEC 12207:1995 e il *Regolamento del progetto didattico (A.a. 2025/2026)*. Per ciascun processo vengono descritte le attività che lo compongono, le procedure operative da seguire e gli strumenti da utilizzare. Il documento è in continua evoluzione e viene aggiornato incrementalmente per tutta la durata del progetto.

1.2 Glossario

Il documento cita alcuni termini la cui definizione può risultare ambigua. Per questo, è possibile consultare il *glossario*$_G$, il quale contiene le definizioni di tali espressioni, marcate da una lettera G a pedice.

1.3 Scopo del Prodotto

Il presente progetto, proposto dall'azienda Vimar S.p.A. attraverso il capitolato View4Life, si inserisce nel contesto della domotica applicata alle residenze protette per anziani. Una residenza protetta è una struttura autonoma per persone anziane perlopiù autosufficienti, che necessitano di un ambiente sicuro, monitorato e confortevole, gestibile anche a distanza da parte del personale sanitario.

L'obiettivo è progettare e realizzare View4Life, una piattaforma web per la gestione completa degli impianti smart nelle residenze protette. La piattaforma si interfaccerà con dispositivi domotici Vimar tramite le API REST KNX IoT 3rd-party, consentendo il controllo di attuatori per l'illuminazione e gli accessi, comandi per tapparelle, sensori UWB per il rilevamento di presenza e cadute, e termostati smart.

Il sistema si compone di due macro-componenti principali:

- un'infrastruttura *Cloud* containerizzata, progettata secondo il principio di Infrastructure as Code (IaC), che ospita tutte le funzioni dell'applicativo;
- un applicativo *web responsive* destinato al personale sanitario, che include un cruscotto informativo, un sistema di gestione degli allarmi, una sezione analytics con suggerimenti per il risparmio energetico e la visualizzazione dello stato dei dispositivi dell'impianto.

Il gruppo SnakeByte si impegna a consegnare il prodotto entro il **5 aprile 2026**, con un budget complessivo preventivato di **euro 13.045**.

1.4 Riferimenti Normativi

- **Standard ISO/IEC 12207:1995**:
  https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf
  (consultato il 04/04/2026);
- **Regolamento del progetto didattico**:
  https://www.math.unipd.it/~tullio/IS-1/2025/Dispense/PD1.pdf
  (consultato il 04/04/2026).

1.5 Riferimenti Informativi

- **Sito dedicato alla documentazione**: https://snakebyteteam.github.io
  (consultato il 04/04/2026)
- **Glossario**: https://snakebyteteam.github.io/glossary.html
  (consultato il 04/04/2026)

2 Processi primari

Le attività che compongono i processi primari considerate in questa sede sono un sottoinsieme proprio delle attività previste dallo Standard ISO/IEC 12207:1995.

2.1 Fornitura

Il processo di fornitura definisce le attività che il gruppo SnakeByte svolge in qualità di fornitore del prodotto *software*, dalla candidatura fino alla consegna finale.

2.1.1 Attività

Il processo di fornitura si compone delle seguenti attività, da eseguire nell'ordine indicato:

1. **Avviamento**: il gruppo esamina i capitolati proposti. Per i capitolati di maggiore interesse vengono inviate comunicazioni via email alla Proponente per richiedere approfondimenti;
2. **Preparazione della risposta**: il gruppo sceglie il capitolato per cui candidarsi e redige la documentazione di candidatura;
3. **Contrattazione**: il gruppo presenta a Vimar S.p.A. le risposte elaborate, concordando i requisiti definitivi e le modalità di collaborazione;
4. **Pianificazione**: il gruppo stabilisce il modello di ciclo di vita, individua le risorse necessarie e identifica i rischi, producendo la relativa documentazione;
5. **Esecuzione e controllo**: il gruppo realizza quanto pianificato, monitorando la qualità dei prodotti e l'avanzamento rispetto agli obiettivi;
6. **Revisione e valutazione**: il gruppo mantiene contatti periodici con Vimar S.p.A. tramite i SAL (Stato Avanzamento Lavori), presentando gli avanzamenti e raccogliendo *feedback*;
7. **Consegna e completamento**: ultimato il prodotto, il gruppo provvede alla consegna di quanto realizzato ai committenti e all'azienda Proponente, garantendo il supporto necessario nelle fasi conclusive.

2.1.2 Procedure

2.1.2.1 Documentazione risultante

La documentazione prodotta durante le attività di fornitura, consegnata ai committenti (prof. Tullio Vardanega e prof. Riccardo Cardin) e alla Proponente, è la seguente:

- **Valutazioni dei capitolati**, contenente:
  - titolo del capitolato e nome dell'azienda proponente;
  - breve descrizione del capitolato e dei suoi obiettivi;
  - considerazioni del gruppo.
- **Dichiarazione degli impegni**, contenente:
  - impegni orari e suddivisione dei ruoli;
  - preventivo dei costi totali del progetto (calcolato secondo il *Regolamento del progetto didattico*);
  - data prevista di consegna.
- **Lettera di presentazione**, contenente:
  - scelta del capitolato;
  - motivazione della scelta;
  - riassunto costi complessivi e data prevista di consegna.

2.1.3 Strumenti

- **Google Calendar**: per condividere tra i membri del gruppo gli appuntamenti previsti (riunioni interne, incontri con la proponente, scadenze);
- **GitHub**: per la gestione delle *Issue* e del Backlog, tracciando le attività da svolgere;
- **Discord**: per le comunicazioni sincrone interne, in particolare per le riunioni interne;
- **WhatsApp**: per le comunicazioni asincrone interne (aggiornamenti rapidi, comunicazioni di servizio);
- **Google Mail**: per le comunicazioni formali scritte verso l'esterno, tramite l'indirizzo *snakebyteteam@gmail.com*;
- **Microsoft Teams**: per le comunicazioni sincrone e asincrone con Vimar S.p.A.;
- **Figma**: per la realizzazione di bozzetti grafici e wireframe dell'applicativo web.

2.2 Sviluppo

Il processo di sviluppo comprende le attività necessarie alla realizzazione del prodotto *software*, orientate verso analisi dei requisiti, progettazione, codifica, integrazione e testing.

2.2.1 Attività

Il processo di sviluppo si compone delle seguenti attività:

1. **System requirements analysis**: analisi e documentazione dei requisiti funzionali del sistema;
2. **Software requirements analysis**: analisi e documentazione dei requisiti non funzionali del *software*;
3. **Software architectural design**: progettazione dell'architettura del sistema;
4. **Software detailed design**: progettazione di dettaglio dei componenti;
5. **Software coding and testing**: codifica delle unità e relativa verifica;
6. **Software qualification testing**: test di qualificazione del *software* integrato;
7. **Software installation**: installazione e configurazione del prodotto nell'ambiente di destinazione.

2.2.2 Procedure

2.2.2.1 Analisi dei Requisiti

La redazione dell'Analisi dei Requisiti è affidata al ruolo dell'Analista ed è strutturata come segue:

1. L'Analista conduce sessioni di *brainstorming*$_G$ interne e raccoglie *feedback*$_G$ dalla Proponente per individuare i casi d'uso;
2. Per ogni caso d'uso viene prodotto un diagramma UML e una tabella descrittiva;
3. I requisiti vengono estratti dai casi d'uso, assegnati a un codice univoco e associati al caso d'uso di origine.

La struttura della tabella dei casi d'uso è la seguente:

Campo	Descrizione
Attori	Coloro che partecipano attivamente al caso d'uso per raggiungere un preciso obiettivo
Pre-condizioni	Condizioni che devono essere soddisfatte prima dello scenario descritto dal caso d'uso
Post-condizioni	Condizioni soddisfatte dopo il completamento dello scenario principale. Se viene completato uno scenario alternativo, sono soddisfatte le relative post-condizioni
Scenario principale	Sequenza di passi che l'utente deve seguire per completare il caso d'uso
Scenari alternativi	Scenario divergente dal principale per il verificarsi di una particolare condizione
Estensioni	Casi d'uso ulteriori eseguiti al verificarsi di una particolare condizione nel caso d'uso primario. Modificano Scenario e Post-condizioni
Inclusioni	Casi d'uso ulteriori eseguiti al fine di completare il caso d'uso principale. Vengono eseguiti tutti incondizionatamente
Generalizzazioni	Casi d'uso più specifici che ereditano comportamento e caratteristiche da un caso d'uso più generale, potendo aggiungere o specializzare ulteriori funzionalità

Tabella 1: Struttura dei casi d'uso

La creazione della tabella è facilitata dall'utilizzo del template `templateAdR.sty`. I campi Scenari alternativi, Estensioni, Inclusioni e Generalizzazioni sono presenti solo quando applicabili.

2.2.2.2 Documentazione risultante

La documentazione prodotta durante il processo di sviluppo è:

- **Diagrammi UML$_G$**: realizzati secondo lo standard UML2.5$_G$, a supporto della definizione dei casi d'uso nell'Analisi dei Requisiti;
- **Analisi dei Requisiti**: documento che raccoglie i risultati della *System requirements analysis* e *Software requirements analysis*.

2.2.2.3 Codifica

I programmatori di SnakeByte seguono le seguenti regole durante la codifica:

1. **Formattazione**: utilizzare ***Prettier*** per la formattazione automatica del codice *TypeScript* (*back-end NestJS* e *frontend Angular*) al salvataggio del file;
2. **Analisi della qualità**: *SonarQube*, integrato in CI/CD tramite *GitHub Actions*, verifica la conformità del codice e impedisce l'introduzione di codice non conforme nel *branch* principale;
3. **Buone pratiche**:
   - evitare variabili globali dove possibile;
   - scrivere funzioni brevi con singola responsabilità;
   - utilizzare la lingua italiana per i commenti;
   - assegnare nomi significativi a variabili, funzioni, metodi e classi;
   - applicare un'indentazione di due spazi, secondo lo standard di *Prettier*.
4. **Nomenclatura e posizione dei file**:
   - i file seguono la convenzione *kebab-case* (es. `user.service.ts`);
   - le classi seguono la convenzione *PascalCase* (es. `UserService`);
   - variabili e funzioni seguono la convenzione *camelCase* (es. `getUserById`);
   - le componenti *NestJS* recano i suffissi `Controller`, `Service`, `Module`, `Repository`;
   - i componenti *Angular* recano i suffissi `Component`, `Service`, `Module`, `Guard`.

2.2.3 Strumenti

- **Visual Studio Code**: ambiente di sviluppo per la scrittura e gestione del codice sorgente;
- **Draw.io**: per la realizzazione dei diagrammi UML dei casi d'uso;
- **StarUML**: per la modellazione architetturale e la progettazione del sistema;
- **Prettier**: per la formattazione automatica del codice *TypeScript*;
- **SonarQube**: per l'analisi statica della qualità del codice in CI/CD.

3 Processi di supporto

3.1 Documentazione

Il processo di documentazione definisce le regole per la produzione, verifica, approvazione e manutenzione di tutti i documenti di progetto.

3.1.1 Attività

Il processo di documentazione si compone delle seguenti attività:

1. **Redazione**: creazione o aggiornamento di un documento da parte del responsabile del ruolo competente;
2. **Verifica**: controllo della correttezza formale e contenutistica del documento da parte del Verificatore;
3. **Approvazione**: approvazione del documento da parte del Responsabile prima del rilascio ufficiale;
4. **Manutenzione**: aggiornamento di documenti esistenti a seguito di nuove informazioni o segnalazioni.

3.1.2 Procedure

3.1.2.1 Struttura generale dei documenti

Ogni documento deve rispettare la seguente struttura:

1. **Prima pagina**, contenente dall'alto verso il basso:
   - logo del gruppo SnakeByte;
   - titolo del documento;
   - nome e numero del gruppo SnakeByte;
   - nome, cognome e numero di matricola di ogni componente;
   - versione attuale, data, stato e destinatari del documento;
   - contatto email del gruppo.
2. **Intestazione** di ogni pagina, contenente:
   - nome del gruppo SnakeByte;
   - titolo del documento.
3. **Registro delle modifiche** (dalla seconda pagina), in forma tabellare, contenente:
   - versione, data, autore, verificatore, approvatore e descrizione riassuntiva di ogni modifica.
4. **Indice** con i titoli di tutte le sezioni e sottosezioni, ciascuno con link ipertestuale alla sezione corrispondente (tramite il pacchetto `hyperref`).

3.1.2.2 Struttura dei verbali

I verbali interni ed esterni, oltre alla struttura generale descritta in §3.1.2.1, devono contenere le seguenti sezioni nell'ordine indicato:

1. **Informazioni**: data, ora di inizio, ora di fine, modalità di svolgimento (presenza, online, asincrona);
2. **Presenze**: tabella con nome, cognome, ruolo (ND se non definito) e presenza di ogni membro;
3. **Ordine del giorno**: elenco degli argomenti trattati;
4. **Approfondimento**: sviluppo degli argomenti all'ordine del giorno;
5. **Decisioni**: tabella delle decisioni prese (§3.1.2.3);
6. **Attività da svolgere**: tabella delle attività emerse (§3.1.2.4);
7. **Attività svolte**: tabella delle attività completate (§3.1.2.5).

3.1.2.3 Tabella delle decisioni

Per ogni decisione presa collettivamente riportare:

- identificativo alfanumerico, nella forma `v{i,e}_AAAA_MM_GG.d<numero_decisione>`;
- descrizione testuale della decisione.

3.1.2.4 Tabella delle attività da svolgere

Per ogni attività emersa riportare:

- identificativo alfanumerico, nella forma `v{i,e}_AAAA_MM_GG.a<numero_attività>`;
- descrizione testuale dell'attività;
- id *GitHub Issue* associata ("-" se assente);
- assegnatario;
- scadenza di completamento.

3.1.2.5 Tabella delle attività svolte

Per ogni attività completata riportare:

- identificativo alfanumerico (come definito in §3.1.2.4);
- id *GitHub Issue* associata ("-" se assente);
- data di completamento.

3.1.2.6 Struttura dei diari di bordo

I diari di bordo sono composti dalle seguenti sezioni:

- **Attività svolte**: quanto realizzato nel periodo tra il diario precedente e quello corrente;
- **Obiettivi per il periodo successivo**: attività che il gruppo intende completare nel periodo successivo;
- **Dubbi e difficoltà**: problematiche riscontrate e questioni ancora aperte.

3.1.2.7 Redazione dei documenti

1. La stesura e l'aggiornamento dei documenti è affidata all'Amministratore, che ricava le informazioni dalle riunioni e comunicazioni con il gruppo;
2. La stesura dei verbali (interni ed esterni) è affidata al Responsabile;
3. La stesura dell'Analisi dei Requisiti è affidata all'Analista (§4.1.1.5).

3.1.2.8 Verifica dei documenti

Il processo di verifica tramite *pull request*$_G$ si svolge come segue:

1. **Esecuzione modifiche**: il redattore crea un *branch* locale denominato `modifica-<nome_documento>`. Modifiche a documenti diversi devono essere presentate in *branch* e *pull request* separati;
2. **Apertura della richiesta**: il redattore effettua il *push* del *branch* e apre una *pull request* verso `develop`, assegnando il Verificatore come assegnatario e *reviewer* e collegando la *Issue* di progetto;
3. **Verifica**: il Verificatore esamina le modifiche:
   (a) **Esito positivo**: approva la *pull request*; il *merge* avviene automaticamente e il *branch* remoto viene eliminato automaticamente (*Settings* > *General* > *Automatically delete head branches*). Il redattore elimina manualmente il *branch* locale;
   (b) **Esito negativo**: il Verificatore inserisce commenti *inline* sui punti critici e imposta lo stato su *Request changes*;
4. **Risoluzione e nuova verifica**: in caso di esito negativo, il redattore applica le correzioni in locale, effettua un nuovo *push* sullo stesso *branch*, risponde ai commenti e richiede una nuova revisione (*Re-request review*). Il punto 4 si itera fino a esito positivo.

3.1.2.9 Approvazione dei documenti

L'approvazione avviene per ogni documento in modo progressivo, secondo le tempistiche delle *baseline*. Il processo si svolge come segue:

1. **Aggiornamento versione**: il redattore crea un *branch* locale `approvazione-<nome_documento>` e incrementa esclusivamente l'indice *Major* (X.0.0);
2. **Apertura richiesta**: il redattore effettua il *push* e apre una *pull request* verso `develop`, impostando il Responsabile come assegnatario e *reviewer*;
3. **Approvazione**: il Responsabile esamina il documento:
   (a) **Esito positivo**: approva la *pull request* e il *merge* su `develop`. Il redattore elimina manualmente il *branch* locale;
   (b) **Esito negativo**: comunica le modifiche richieste tramite commenti e imposta lo stato su *Request Changes*;
4. **Risoluzione e nuova approvazione**: il redattore applica le modifiche, ripete il processo di verifica (§3.1.2.7), aggiorna la *pull request* e richiede una nuova approvazione. Il punto 4 si itera fino a esito positivo.

Una volta che tutti i documenti in `develop` sono stati approvati, si procede con il *merge* di `develop` in `main`.

3.1.2.10 Manutenzione dei documenti

Il processo di aggiornamento di un documento esistente segue le stesse regole della prima redazione:

1. creare un *branch* `modifica-<nome_documento>`;
2. applicare le modifiche e sottoporle a verifica tramite *pull request* (§3.1.2.7);
3. le modifiche vengono integrate in `develop` a seguito di verifica positiva e in `main` solo in corrispondenza delle *baseline* di progetto.

3.1.2.11 Nomenclatura dei documenti

La convenzione per la nomenclatura dei file è la seguente (dove X, Y, Z sono i numeri di versione descritti in §3.2.1):

- **Verbali**: `v{i,e}_AAAA_MM_GG_vX.Y.Z`
- **Generale**: `<NOME DOC.>_vX.Y.Z` (dove `<NOME DOC.>` è il nome del documento)

Gli acronimi utilizzati sono:

Abbreviazione	Significato
VI	Verbale interno
VE	Verbale esterno
NdP	Norme di Progetto
PdP	Piano di Progetto
PdQ	Piano di Qualifica
PB	Product Baseline
RTB	Requirements and Technology Baseline

Tabella 2: Acronimi dei documenti

3.1.3 Strumenti

- **LaTeX**: linguaggio di *markup* per la redazione di tutti i documenti, tramite i template `template.sty` e `template` forniti nel *repository* interno;
- **Hunspell**: per il controllo ortografico e lessicale dei documenti;
- **GitHub**: per la redazione, verifica e approvazione dei documenti.

3.2 Gestione della configurazione

Il processo di gestione della configurazione identifica, traccia e controlla le modifiche apportate agli elementi del progetto, garantendone completezza, coerenza e correttezza secondo lo Standard ISO/IEC 12207:1995.

3.2.1 Attività

Il processo si compone delle seguenti attività:

1. **Identificazione della configurazione**: definizione degli elementi da porre sotto controllo di configurazione e assegnazione dei numeri di versione;
2. **Controllo della configurazione**: gestione delle richieste di modifica, che devono essere notificate e approvate prima dell'integrazione nel *branch* principale;
3. **Valutazione della configurazione**: verifica della completezza del *software* prodotto rispetto ai requisiti e al design scelto.

3.2.2 Procedure

3.2.2.1 Numeri di versione dei documenti

La convenzione per il numero di versione è il formato X.Y.Z (*MAJOR.MINOR.PATCH*); ogni documento parte dalla versione 0.1.0.

- **PATCH (Z)**: avanza di un'unità per ogni modifica minore (correzioni grammaticali, integrazioni di informazioni poco significative);
- **MINOR (Y)**: avanza di un'unità per ogni modifica maggiore (aggiunta di sezioni, modifiche sostanziali al contenuto);
- **MAJOR (X)**: avanza di un'unità solo al completamento con successo del processo di approvazione da parte del Responsabile.

Ogni avanzamento di X o Y azzera le cifre a destra.

3.2.2.2 Controllo della configurazione

Ogni modifica da apportare agli artefatti di progetto deve seguire il seguente iter:

1. Aprire una *Issue* su *GitHub* con: numero univoco, titolo descrittivo, label di classificazione, tipo (**Task**, **Feature** o **Bug**), milestone di riferimento e project associato;
2. Assegnare la *Issue* al componente responsabile;
3. Aggiornare lo stato della *Issue* nella *Project Board* (*Backlog* → *Ready* → *In Progress* → *In Review* → *Done*);
4. Chiudere la *Issue* solo dopo che l'attività è stata completata e verificata e le modifiche sono state integrate nel *branch* principale tramite *pull request*.

3.2.2.3 Valutazione della configurazione

Il tracciamento della conformità del prodotto avviene come segue:

1. Ogni requisito definito nell'Analisi dei Requisiti viene associato ai test che ne verificano il soddisfacimento nel *Piano di Qualifica*;
2. Un requisito si intende soddisfatto solo quando tutti i test ad esso associati terminano con esito positivo;
3. Il tracciamento viene mantenuto aggiornato durante l'intero ciclo di sviluppo.

3.2.2.4 Repository

I *repository* adottati per la gestione della configurazione sono:

- **MVP**: *repository* contenente il *Minimum Viable Product* del progetto;
- **snakebyte.github.io**: *repository* contenente il codice sorgente del sito web di documentazione e la documentazione stessa.

3.2.3 Strumenti

- **Git**$_G$: *Version Control System* distribuito e *open source* per il versionamento dei file;
- **GitHub**: servizio di *hosting* per progetti *software* e implementazione di *Git*, utilizzato per *Issue*, *Project Board* e *pull request*.

3.3 Verifica

Il processo di verifica garantisce che i prodotti realizzati siano conformi ai requisiti e agli standard di qualità definiti nel *Piano di Qualifica*.

3.3.1 Attività

Il processo di verifica si compone delle seguenti attività:

1. **Analisi statica**: controllo degli artefatti senza esecuzione del prodotto (documenti e codice);
2. **Analisi dinamica**: esecuzione di test sul prodotto *software* in ambiente controllato;
3. **Gestione e tracciamento dei test**: collegamento dei test ai requisiti e monitoraggio del loro stato.

3.3.2 Procedure

3.3.2.1 Analisi statica

Il gruppo SnakeByte adotta la tecnica di **Inspection**: il Verificatore esamina l'artefatto utilizzando una *checklist* predefinita, identifica anomalie e difetti in modo sistematico, quindi ne discute gli esiti in riunione con l'autore. La *checklist* deve coprire:

- controllo sintattico e semantico;
- conformità agli standard adottati;
- correttezza ortografica (per i documenti);
- rispetto delle convenzioni di codifica (per il codice).

3.3.2.2 Analisi dinamica

I test devono possedere le seguenti caratteristiche: ripetibilità, automatizzazione, determinismo e indipendenza. Il gruppo implementa le seguenti tipologie di test:

- **Test di unità (TU)**: verificano il corretto funzionamento delle singole unità di codice (funzioni, metodi, classi), indipendentemente dalle altre componenti. Si distinguono in:
  - *Test funzionali (Black-box)*: verificano i risultati prodotti per valori non ammissibili inferiori, valori estremi inferiori ammissibili, valori ammissibili, valori estremi superiori ammissibili e valori non ammissibili superiori;
  - *Test strutturali (White-box)*: verificano che tutti gli statement rilevanti del codice siano eseguiti almeno una volta.
- **Test di integrazione (TI)**: verificano la corretta interazione tra le componenti del sistema, adottando una strategia *top-down* o *bottom-up*. Per isolare le unità si utilizzano:
  - *Stub*: componenti semplificati che forniscono risposte predefinite per simulare dipendenze non implementate;
  - *Mock*: oggetti che registrano le chiamate ricevute e validano le interazioni tra moduli.
- **Test di sistema (TS)**: verificano il comportamento dell'intero sistema integrato rispetto ai requisiti funzionali e non funzionali, dal punto di vista dell'utente finale;
- **Test di regressione (TR)**: rieseguono test già validati dopo ogni modifica, per garantire che le funzionalità precedentemente verificate non siano state compromesse.

3.3.2.3 Nomenclatura dei test

Ogni test è identificato da un codice univoco nella forma:

T[Tipo-test]-[X]

dove:

- [**Tipo-test**]: **U** = Unità, **I** = Integrazione, **S** = Sistema, **R** = Regressione;
- [**X**]: numero intero progressivo.

3.3.2.4 Stato dei test

Ogni test può assumere uno dei seguenti stati:

- **NI**: non ancora implementato;
- **S**: eseguito con successo;
- **F**: fallito.

3.3.2.5 Gestione e tracciamento dei test

1. Il Verificatore definisce i test da eseguire e li associa ai requisiti corrispondenti nel *Piano di Qualifica*;
2. Un requisito si intende soddisfatto solo quando tutti i test ad esso associati terminano con esito positivo;
3. I test vengono eseguiti durante la scrittura delle funzionalità, dopo ogni modifica o correzione (tramite test di regressione) e nella fase finale di validazione;
4. Il Verificatore registra gli esiti dei test e segnala le anomalie ai Programmatori;
5. I Programmatori correggono i difetti riscontrati e creano i test di unità.

3.3.3 Strumenti

- **GitHub** (*pull request*): per la verifica dei documenti tramite il *workflow* descritto in §3.1.2.7;
- **SonarQube**: per l'analisi statica automatica del codice in CI/CD tramite *GitHub Actions*.

3.4 Validazione

Il processo di validazione conferma che il prodotto finale soddisfi le aspettative e i bisogni della Proponente.

3.4.1 Attività

Il processo di validazione si compone delle seguenti attività:

1. **Pianificazione del collaudo**: definizione dei casi di test di accettazione e predisposizione dell'ambiente di test;
2. **Esecuzione del collaudo**: esecuzione dei test di accettazione in presenza della Proponente;
3. **Documentazione degli esiti**: registrazione tracciabile dei risultati di ogni test di accettazione e raccolta di feedback.

3.4.2 Procedure

3.4.2.1 Collaudo

Il collaudo si svolge secondo i seguenti passi:

1. Predisporre un ambiente di test rappresentativo dello scenario d'uso reale;
2. Eseguire i test di accettazione basati sui casi d'uso e scenari definiti nell'Analisi dei Requisiti;
3. Documentare in modo completo e tracciabile l'esito di ogni test;
4. Raccogliere feedback dalla Proponente per eventuali correzioni;
5. L'esito positivo della validazione costituisce il presupposto per il rilascio del prodotto.

3.4.3 Strumenti

- **Microsoft Teams**: per lo svolgimento del collaudo in modalità remota con la Proponente;
- **Verbali esterni**: per la documentazione degli esiti e delle decisioni del collaudo.

3.5 Garanzia della qualità

Il processo di garanzia della qualità assicura che i prodotti e i processi del progetto siano conformi agli standard e ai requisiti definiti nel *Piano di Qualifica*.

3.5.1 Attività

Il processo si compone delle seguenti attività:

1. **Controllo di conformità**: verifica che i prodotti rispettino gli standard di qualità adottati;
2. **Monitoraggio dei processi**: verifica che i processi siano applicati secondo quanto definito nelle Norme di Progetto;
3. **Identificazione delle non conformità**: segnalazione delle anomalie rilevate e proposta di azioni correttive;
4. **Tracciabilità delle verifiche**: registrazione degli esiti di tutte le attività di Quality Assurance.

3.5.2 Procedure

3.5.2.1 Responsabilità del Verificatore

Il Verificatore è tenuto a:

1. controllare la conformità di ogni prodotto rispetto ai criteri di qualità definiti;
2. registrare gli esiti di ogni attività di verifica nel *Piano di Qualifica*;
3. segnalare immediatamente le anomalie rilevate e richiedere gli interventi correttivi necessari;
4. garantire completezza e sistematicità nelle verifiche effettuate.

3.5.2.2 Metriche di qualità

Le metriche di qualità di processo e di prodotto sono definite nelle sezioni §5 e §6. Il Verificatore è tenuto a monitorarle costantemente per identificare criticità e intraprendere azioni correttive.

3.5.3 Strumenti

- **GitHub** (*pull request*): per il tracciamento delle verifiche sui documenti;
- **Piano di Qualifica**: documento in cui vengono registrati esiti delle verifiche e stato delle metriche.

3.6 Revisioni congiunte

Il processo di revisioni congiunte prevede incontri periodici con la Proponente per presentare l'avanzamento del progetto, raccogliere feedback e validare le scelte implementative.

3.6.1 Attività

Il processo si compone delle seguenti attività:

1. **Pianificazione del SAL**: concordare con la Proponente la data e l'ordine del giorno dell'incontro;
2. **Svolgimento del SAL**: presentare i progressi, eventuali demo, le problematiche emerse e la pianificazione delle attività successive;
3. **Documentazione**: redigere il verbale esterno che traccia discussioni, decisioni e feedback ricevuti.

3.6.2 Procedure

3.6.2.1 SAL – Stato Avanzamento Lavori

Ogni SAL deve includere:

1. presentazione dei progressi realizzati rispetto alla pianificazione;
2. dimostrazione delle funzionalità implementate (se disponibili);
3. esposizione delle problematiche emerse e delle soluzioni proposte;
4. illustrazione della pianificazione delle attività successive;
5. redazione del verbale esterno al termine dell'incontro.

3.6.3 Strumenti

- **Microsoft Teams**: per gli incontri online con la Proponente;
- **Verbali esterni**: per il tracciamento di contenuti e decisioni dei SAL;
- **Demo**: presentazioni del prodotto in sviluppo per raccogliere *feedback*.

3.7 Audit

Il processo di audit consiste in verifiche formali condotte per accertare che prodotti e processi siano conformi ai requisiti e agli standard adottati, in corrispondenza delle *milestone* di progetto.

3.7.1 Attività

Il processo si compone delle seguenti attività:

1. **Pianificazione dell'audit**: il Responsabile pianifica l'audit in corrispondenza delle *milestone* di progetto;
2. **Esecuzione dell'audit**: verifica della completezza e conformità degli artefatti prima delle consegne ufficiali;
3. **Documentazione degli esiti**: registrazione delle non conformità rilevate e richiesta di azioni correttive.

3.7.2 Procedure

3.7.2.1 Audit di baseline

Prima del raggiungimento di ogni *baseline* (RTB e PB), il Responsabile deve verificare:

1. completezza della documentazione richiesta;
2. conformità dei documenti rispetto ai requisiti del capitolato e alle aspettative della Proponente;
3. tracciabilità tra requisiti, casi d'uso, test e implementazione;
4. rispetto degli standard di qualità definiti nel *Piano di Qualifica*.

Le non conformità rilevate devono essere documentate e risolte prima del rilascio ufficiale.

3.7.3 Strumenti

- **Piano di Qualifica**: per il riscontro degli standard di qualità attesi;
- **GitHub** (*Issue* e *pull request*): per il tracciamento delle non conformità e delle relative azioni correttive.

3.8 Risoluzione dei problemi

Il processo di risoluzione dei problemi definisce le modalità con cui il gruppo identifica, traccia e risolve anomalie, difetti e richieste di miglioramento.

3.8.1 Attività

Il processo si compone delle seguenti attività:

1. **Segnalazione**: identificazione e registrazione del problema;
2. **Analisi e assegnazione**: valutazione di priorità e impatto, assegnazione al membro competente;
3. **Risoluzione**: implementazione della soluzione;
4. **Verifica e chiusura**: controllo della correttezza della soluzione e chiusura della *Issue*.

3.8.2 Procedure

3.8.2.1 Workflow di risoluzione

1. Un membro del gruppo identifica un problema e crea una *Issue* su *GitHub* con titolo, descrizione, label, assegnatario, milestone e priorità;
2. Il Responsabile o l'assegnatario analizza il problema e ne valuta priorità e impatto;
3. Il problema viene assegnato al membro con le competenze appropriate;
4. L'assegnatario implementa la soluzione seguendo il processo di verifica tramite *pull request* (§3.1.2.7);
5. Il Verificatore controlla la correttezza della soluzione;
6. La *Issue* viene chiusa e collegata al commit o alla *pull request* risolutiva.

3.8.3 Strumenti

- **GitHub Issues**: per la segnalazione e il tracciamento centralizzato dei problemi;
- **GitHub Projects** (*Project Board*): per la visualizzazione dello stato di avanzamento delle *Issue*.

4 Processi organizzativi

4.1 Processo di gestione

Il processo di gestione definisce le attività e i compiti di pianificazione, coordinamento, controllo e gestione dei rischi del progetto.

4.1.1 Attività

Il processo di gestione si compone delle seguenti attività:

1. **Pianificazione**: assegnazione dei ruoli e definizione delle responsabilità per ogni sprint;
2. **Coordinamento**: gestione delle comunicazioni interne ed esterne e delle riunioni;
3. **Controllo e monitoraggio**: tracciamento dell'avanzamento delle attività e delle ore lavorate;
4. **Gestione dei rischi**: identificazione, analisi e mitigazione dei rischi durante l'intero ciclo di vita del progetto.

4.1.2 Procedure

4.1.2.1 Assegnazione delle responsabilità

Per tutta la durata del progetto, i membri del gruppo ricoprono a rotazione i seguenti ruoli, con rotazione a inizio di ogni sprint. Ogni componente deve ricoprire almeno una volta ciascun ruolo:

- **Responsabile** (§4.1.2.1): una persona per sprint;
- **Amministratore** (§4.1.2.2): più persone per sprint;
- **Progettista** (§4.1.2.3): più persone per sprint;
- **Analista** (§4.1.2.4): più persone per sprint;
- **Verificatore** (§4.1.2.5): più persone per sprint;
- **Programmatore** (§4.1.2.6): più persone per sprint.

4.1.2.2 Responsabile

Il Responsabile (*Project Manager*) governa il team e rappresenta il progetto verso l'esterno. I suoi compiti sono:

- prendere decisioni nell'interesse del gruppo;
- approvare il lavoro realizzato dagli altri ruoli;
- pianificare le attività e gestire le risorse necessarie;
- coordinare le relazioni con l'esterno.

4.1.2.3 Amministratore

L'Amministratore (*Sysadmin*) definisce, controlla e mantiene l'ambiente informatico di lavoro. I suoi compiti sono:

- definire, selezionare e mettere in opera le risorse informatiche a supporto delle Norme di Progetto;
- gestire le segnalazioni (*ticket*) sul non funzionamento dell'infrastruttura.

4.1.2.4 Progettista

Il Progettista determina le scelte realizzative del prodotto nella fase di sviluppo. I suoi compiti sono:

- progettare l'architettura del prodotto per ottenere la migliore efficienza ed efficacia;
- supervisionare la fase di codifica a supporto dei Programmatori.

4.1.2.5 Analista

L'Analista conosce il dominio del problema e ha competenze avanzate sull'analisi dei requisiti. I suoi compiti sono:

- analizzare i requisiti e il dominio applicativo;
- definire fattibilità e obiettivi delle attività;
- redigere il documento *Analisi dei Requisiti*.

4.1.2.6 Verificatore

Il Verificatore analizza la conformità del lavoro svolto dagli altri ruoli. I suoi compiti sono:

- verificare che gli artefatti *software* e documentali siano aderenti alle Norme di Progetto e agli obiettivi;
- segnalare gli errori da correggere;
- redigere test di verifica e di aderenza ai requisiti per i prodotti *software*.

4.1.2.7 Programmatore

Il Programmatore è responsabile della realizzazione e manutenzione del prodotto *software*. I suoi compiti sono:

- scrivere codice secondo le specifiche del Progettista, perseguendo gli obiettivi definiti dall'Analista;
- redigere la documentazione tecnica (*Manuale*) del prodotto.

4.1.2.8 Coordinamento

Le regole di coordinamento del gruppo sono le seguenti:

1. **Riunione interna fissa**: ogni Lunedì, a partire dalle ore 12:00, si svolge una riunione in presenza o in modalità asincrona. La partecipazione di tutti i membri è desiderabile ma non obbligatoria; la riunione viene annullata in caso di impossibilità della maggior parte del gruppo. Ulteriori riunioni possono essere concordate su disponibilità dei componenti;
2. **Comunicazioni interne**: le comunicazioni sincrone e asincrone interne avvengono tramite **Discord**$_G$; le comunicazioni asincrone rapide tramite **WhatsApp**;
3. **Comunicazioni esterne**: le comunicazioni formali scritte avvengono tramite **Google Mail** all'indirizzo *snakebyteteam@gmail.com*; le comunicazioni sincrone con Vimar S.p.A. avvengono tramite **Microsoft Teams**.

4.1.2.9 Gestione degli sprint

Il gruppo adotta un modello di sviluppo *agile*$_G$ basato su sprint$_G$ di durata fissa pari a due settimane. Ogni sprint si svolge come segue:

1. **Sprint Planning**: all'inizio dello sprint il gruppo si riunisce per definire gli obiettivi, selezionare le attività, stimare l'impegno orario, assegnare i ruoli secondo il principio di rotazione e aggiornare il documento di pianificazione condiviso;
2. **Sviluppo**: ciascun membro svolge le attività assegnate, aggiornando lo stato delle *Issue* nella *Project Board*;
3. **Sprint Review**: al termine dello sprint il gruppo verifica il lavoro completato, valuta il raggiungimento degli obiettivi e raccoglie feedback dalla Proponente quando previsto;
4. **Sprint Retrospective**: il gruppo analizza criticamente il processo, identifica problematiche e propone azioni di miglioramento da documentare nel *Piano di Progetto*.

4.1.2.10 Tracciamento delle attività

Ogni attività di progetto viene gestita come segue:

1. creare una *Issue* su *GitHub* con: titolo descrittivo, assegnatario/i, label, milestone e scadenza;
2. aggiornare regolarmente lo stato della *Issue* nella *Project Board* (Backlog → Ready → In Progress → In Review → Done);
3. commentare sulla *Issue* in caso di difficoltà o variazioni rispetto alla pianificazione.

4.1.2.11 Pianificazione e monitoraggio delle ore

Per ogni sprint, il gruppo utilizza il documento condiviso Google Sheets$_G$ come segue:

1. compilare il **foglio di pianificazione** con l'assegnazione dei ruoli e le ore preventivate per ogni membro;
2. al termine dello sprint, compilare il **foglio di consuntivo** con le ore effettivamente lavorate, specificando ruolo e attività svolte;
3. sincronizzare periodicamente i dati con il *Piano di Progetto*.

4.1.2.12 Gestione dei rischi

1. I rischi vengono identificati e classificati per categoria (tecnologici, di comunicazione, organizzativi, interpersonali) nel *Piano di Progetto*;
2. Per ogni rischio vengono valutati probabilità e impatto e definite le strategie di prevenzione e mitigazione;
3. Durante la *Sprint Retrospective*, il gruppo analizza i rischi manifestatisi, valuta l'efficacia delle strategie adottate e propone azioni correttive per gli sprint successivi.

4.1.3 Strumenti

- **GitHub Projects**: per la gestione delle *Issue* e la visualizzazione dell'avanzamento tramite *Project Board*;
- **Google Sheets**: per la pianificazione dei ruoli e il tracciamento delle ore lavorate;
- **Discord**: per le comunicazioni interne sincrone e asincrone;
- **Microsoft Teams**: per le comunicazioni sincrone con la Proponente;
- **Google Calendar**: per la condivisione di appuntamenti e scadenze.

4.2 Processo di miglioramento

Il processo di miglioramento stabilisce, consolida, misura, controlla e migliora i processi utilizzati durante il ciclo di sviluppo del *software*.

4.2.1 Attività

Il processo prevede le seguenti attività:

1. **Inizializzazione**: stabilire i processi organizzativi e realizzare la documentazione, come definito nel presente documento;
2. **Valutazione**: individuare misurazioni appropriate e controllare periodicamente i dati raccolti (cfr. §5);
3. **Miglioramento**: identificare i processi con criticità, definire azioni correttive e aggiornare la documentazione per riflettere i cambiamenti adottati.

4.2.2 Procedure

4.2.2.1 Ciclo di miglioramento

1. Al termine di ogni sprint, il Responsabile verifica i valori delle metriche di processo definite in §5;
2. Per ogni metrica fuori soglia, viene aperta una *Issue* su *GitHub* con la descrizione della criticità rilevata;
3. Nella Sprint Retrospective, il gruppo definisce le azioni correttive da implementare nello sprint successivo;
4. Le modifiche ai processi vengono documentate nelle Norme di Progetto tramite il consueto processo di manutenzione (§3.1.2.9).

4.2.3 Strumenti

- **Piano di Qualifica**: per il monitoraggio delle metriche di processo;
- **GitHub Issues**: per il tracciamento delle criticità e delle azioni correttive.

4.3 Processo di formazione

Il processo di formazione mantiene i membri di SnakeByte aggiornati sui cambiamenti effettuati e definisce le modalità di acquisizione delle conoscenze necessarie allo svolgimento del progetto.

4.3.1 Attività

Il processo include le seguenti attività:

1. **Implementazione del processo**: realizzare i meccanismi necessari alla formazione del gruppo;
2. **Sviluppo del materiale formativo**: individuare le risorse utili all'apprendimento delle tecnologie adottate;
3. **Implementazione del piano formativo**: definire un piano per formare e mantenere formato il gruppo durante l'intero ciclo di vita del progetto.

4.3.2 Procedure

4.3.2.1 Tecnologie da apprendere

Il gruppo SnakeByte ha individuato le seguenti tecnologie necessarie al completamento del progetto:

- **NestJS**: *framework* backend per Node.js;
- **Angular**: *framework* per lo sviluppo del frontend;
- **PostgreSQL**: sistema di gestione del database relazionale;
- **Docker**: strumento per la containerizzazione dell'infrastruttura;
- **KNX IoT 3rd-party API**: interfaccia REST per la comunicazione con gli impianti Vimar;
- **LaTeX**: linguaggio di markup per la redazione della documentazione.

4.3.2.2 Risorse per la formazione

Per ogni tecnologia, il gruppo adotta le seguenti risorse:

- **NestJS**: documentazione ufficiale sul sito del *framework*;
- **Angular**: documentazione ufficiale sul sito del *framework*;
- **PostgreSQL**: documentazione ufficiale sul sito del progetto;
- **Docker**: sessione formativa tenuta da Vimar S.p.A., integrata dalla documentazione ufficiale;
- **KNX IoT 3rd-party API**: sessione formativa tenuta da Vimar S.p.A., integrata dalla documentazione tecnica fornita dalla stessa;
- **LaTeX**: documentazione ufficiale sul sito del progetto.

4.3.2.3 Piano per la formazione

1. Durante ogni sprint, ciascun membro dedica appositi momenti alla formazione individuale sulle tecnologie assegnate, in particolare nelle fasi iniziali del progetto;
2. Il gruppo può pianificare sprint dedicati esclusivamente allo studio, se ritenuto necessario;
3. Le sessioni formative tenute da Vimar S.p.A. vengono verbalizzate in verbali esterni.

4.3.3 Strumenti

- **Documentazione ufficiale** delle tecnologie adottate (*NestJS*, *Angular*, *PostgreSQL*, *Docker*, LaTeX);
- **Microsoft Teams**: per le sessioni formative con Vimar S.p.A.;
- **Verbali esterni**: per la tracciatura dei contenuti delle sessioni formative esterne.

5 Metriche di qualità di processo

In questa sezione vengono definite le metriche di qualità adottate per monitorare e valutare in modo quantitativo l'efficacia e l'efficienza dei processi. Ogni metrica è identificata da un codice univoco nella forma *MPCX*, dove *MPC* è l'acronimo di *Metrica di Processo* e *X* è un numero intero progressivo.

5.1 Processi primari

5.1.1 Fornitura

5.1.1.1 MPC1 - Planned Value (PV)

Codice	MPC1
Formula	$PV_k = BAC \cdot$ Percentuale completamento pianificato allo sprint k Dove: • BAC = Budget totale previsto (*Budget At Completion*); • k = Numero dello sprint di riferimento.
Descrizione	Valore pianificato del lavoro che, secondo il piano, dovrebbe essere raggiunto al termine dello *sprint* k. Viene calcolato al termine di ciascuno sprint.
Utilità	Fornisce un punto di riferimento temporale per valutare se il progetto procede secondo la schedulazione prevista, permettendo di identificare tempestivamente ritardi o anticipi.

Tabella 3: MPC1 - Planned Value (PV)

5.1.1.2 MPC2 - Earned Value (EV)

Codice	MPC2
Formula	$$EV_k = BAC \cdot \text{Percentuale completamento effettivo allo sprint k}$$ Dove: • BAC = Budget totale previsto (*Budget At Completion*); • k = Numero dello sprint di riferimento.
Descrizione	Valore effettivo del lavoro raggiunto al termine dello *sprint* k. Viene calcolato al termine di ciascuno sprint.
Utilità	Confronta il lavoro completato con i costi sostenuti (tramite CPI) e con la pianificazione temporale (tramite SPI), costituendo la base per tutte le analisi di performance.

Tabella 4: MPC2 - Earned Value (EV)

5.1.1.3 MPC3 - Actual Cost (AC)

Codice	MPC3
Formula	$$AC_k = \sum_{i=1}^{k} \text{Costo sostenuto nello sprint i}$$ Dove: k = Numero dello sprint di riferimento.
Descrizione	Costo effettivo sostenuto per il lavoro svolto fino a un determinato sprint. Viene calcolato alla fine di ogni sprint.
Utilità	Permette di confrontare le spese effettive con il budget pianificato, rilevando tempestivamente sforamenti o inefficienze.

Tabella 5: MPC3 - Actual Cost (AC)

5.1.1.4 MPC4 - Cost Performance Index (CPI)

Codice	MPC4
Formula	$$CPI_k = \frac{EV_k}{AC_k}$$ Dove: EV_k (§5.1.1.2), AC_k (§5.1.1.3), k = numero sprint.
Descrizione	Efficienza nell'utilizzo del budget: rapporto tra il valore del lavoro raggiunto e il costo reale sostenuto allo sprint k. • CPI > 1: progetto sotto budget; • CPI < 1: progetto sopra budget; • CPI $= 1$: progetto in linea con il budget.
Utilità	Quantifica l'efficienza nell'uso delle risorse e prevede se il progetto terminerà entro il budget.

Tabella 6: MPC4 - Cost Performance Index (CPI)

5.1.1.5 MPC5 - Schedule Performance Index (SPI)

Codice	MPC5
Formula	$$SPI_k = \frac{EV_k}{PV_k}$$ Dove: EV_k (§5.1.1.2), PV_k (§5.1.1.1), k = numero sprint.
Descrizione	Efficienza nel completamento del lavoro rispetto alla pianificazione: rapporto tra valore del lavoro completato e valore pianificato fino allo sprint k. • SPI > 1: progetto in anticipo; • SPI < 1: progetto in ritardo; • SPI $= 1$: progetto nei tempi previsti. Un SPI costantemente inferiore a 1 segnala la necessità di rivedere la pianificazione o allocare risorse aggiuntive.
Utilità	Permette di valutare il rispetto delle scadenze e stimare se il progetto terminerà nei tempi pianificati.

Tabella 7: MPC5 - Schedule Performance Index (SPI)

5.1.1.6 MPC6 - Estimate At Completion (EAC)

Codice	MPC6
Formula	$$EAC_k = AC_k + (BAC - EV_k)$$ Dove: AC_k (§5.1.1.3), BAC = Budget At Completion, EV_k (§5.1.1.2).
Descrizione	Stima del costo totale finale del progetto al completamento.
Utilità	Fornisce una previsione realistica del budget necessario e consente di identificare tempestivamente potenziali sforamenti.

Tabella 8: MPC6 - Estimate At Completion (EAC)

5.1.1.7 MPC7 - Estimate To Complete (ETC)

Codice	MPC7
Formula	$$ETC_k = EAC_k - AC_k$$ Dove: EAC_k (§5.1.1.6), AC_k (§5.1.1.3).
Descrizione	Stima del costo necessario per completare il lavoro rimanente del progetto allo sprint k.
Utilità	Permette di valutare se le risorse economiche residue sono sufficienti per completare il progetto.

Tabella 9: MPC7 - Estimate To Complete (ETC)

5.1.2 Sviluppo

5.1.2.1 MPC8 - Requirements Stability Index (RSI)

Codice	MPC8
Formula	$$RSI = \frac{R_{iniziali} - (R_{modificati} + R_{cancellati})}{R_{iniziali} + R_{aggiunti}} \cdot 100$$ Dove: $R_{iniziali}$, $R_{modificati}$, $R_{cancellati}$, $R_{aggiunti}$ come definiti.
Descrizione	Indice che misura la stabilità dei requisiti del progetto durante il ciclo di vita. • RSI = 100%: requisiti stabili; • 80% ≤ RSI < 100%: stabilità accettabile; • 50% ≤ RSI < 80%: elevata volatilità; • RSI < 50%: instabilità critica.
Utilità	Valuta la qualità dell'analisi iniziale dei requisiti e identifica problemi nella loro definizione.

Tabella 10: MPC8 - Requirements Stability Index (RSI)

5.2 Processi di supporto

5.2.1 Documentazione

5.2.1.1 MPC9 - Indice di Gulpease (IG)

Codice	MPC9
Formula	$$IG = 89 + \frac{300 \cdot N_{frasi} - 10 \cdot N_{lettere}}{N_{parole}}$$
Descrizione	Indice di leggibilità di un testo in lingua italiana. • IG ≥ 80: testo molto facile; • 60 ≤ IG < 80: testo facile; • 40 ≤ IG < 60: testo abbastanza difficile; • IG < 40: testo difficile.
Utilità	Garantisce che i documenti siano comprensibili, riducendo ambiguità e incomprensioni.

Tabella 11: MPC9 - Indice di Gulpease (IG)

5.2.1.2 MPC10 - Indice di Frammentazione (IF)

Codice	MPC10
Formula	$$IF = \frac{N_{sezioni} + N_{paragrafi}}{N_{linee}} \cdot 100$$
Descrizione	Grado di suddivisione del testo in unità logiche. • IF $< 5\%$: testo molto denso; • $5\% \leq$ IF $\leq 20\%$: bilanciamento ottimale; • IF $> 20\%$: testo eccessivamente frammentato.
Utilità	Assicura che la documentazione sia strutturata per facilitare la consultazione e il reperimento delle informazioni.

Tabella 12: MPC10 - Indice di Frammentazione (IF)

5.2.2 Verifica

5.2.2.1 MPC11 - Test Success Rate (TSR)

Codice	MPC11
Formula	$$TSR = \frac{T_{successo}}{T_{totali}} \cdot 100$$
Descrizione	Percentuale di test eseguiti con esito positivo. • TSR $= 100\%$: qualità eccellente; • $80\% \leq$ TSR $< 100\%$: qualità accettabile; • $70\% \leq$ TSR $< 80\%$: necessarie correzioni; • TSR $< 70\%$: qualità inaccettabile.
Utilità	Misura la correttezza del *software* e l'efficacia dell'attività di testing.

Tabella 13: MPC11 - Test Success Rate (TSR)

5.2.3 Gestione della qualità

5.2.3.1 MPC12 - Percentuale Metriche Soddisfatte (PMS)

Codice	MPC12
Formula	$$PMS = \frac{M_{soddisfatte}}{M_{totali}} \cdot 100$$
Descrizione	Percentuale di metriche di qualità soddisfatte sul totale definite. • PMS $\geq$ 90%: eccellente; • 80% $\leq$ PMS $<$ 90%: buono; • 70% $\leq$ PMS $<$ 80%: sufficiente; • PMS $<$ 70%: insoddisfacente.
Utilità	Offre una valutazione complessiva della qualità del progetto e facilita l'identificazione di criticità.

Tabella 14: MPC12 - Percentuale Metriche Soddisfatte (PMS)

5.3 Processi organizzativi

5.3.1 Gestione dei processi

5.3.1.1 MPC13 - Percentuale Rischi Inattesi (PRI)

Codice	MPC13
Formula	$$PRI_k = \frac{R_{inattesi,k}}{R_{totali,k}} \cdot 100$$
Descrizione	Percentuale di rischi non previsti manifestatisi nello sprint k. • PRI = 0: eccellente; • PRI $\leq$ 5%: buono; • 5% $<$ PRI $\leq$ 20%: accettabile ma migliorabile; • 20% $<$ PRI $\leq$ 40%: insufficiente; • PRI $>$ 40%: gestione inadeguata.
Utilità	Misura l'accuratezza dell'attività di analisi e gestione dei rischi.

Tabella 15: MPC13 - Percentuale Rischi Inattesi (PRI)

5.3.1.2 MPC14 - Labor Efficiency (LE)

Codice	MPC14
Formula	$$LE_k = \frac{H_{pianificate}}{H_{effettive}} \cdot 100$$
Descrizione	Rapporto tra ore pianificate e ore effettivamente impiegate nello sprint k. • LE $> 100\%$: il gruppo è stato più veloce del previsto; • LE $< 100\%$: il team ha impiegato più tempo del previsto; • LE $= 100\%$: stime perfettamente in linea.
Utilità	Monitora la precisione delle stime temporali, complementando lo SPI (§5.1.1.5) che valuta l'avanzamento in termini economici.

Tabella 16: MPC14 - Labor Efficiency (LE)

6 Metriche di qualità di prodotto

In questa sezione vengono definite le metriche di qualità adottate per garantire che i prodotti di progetto (documentazione e software) soddisfino i requisiti qualitativi stabiliti. Ogni metrica è identificata da un codice univoco nella forma *MPDX*, dove *MPD* è l'acronimo di *Metrica di Prodotto* e *X* è un numero intero progressivo.

6.1 Funzionalità

6.1.0.1 MPD1 – Percentuale Requisiti Obbligatori Soddisfatti (PROS)

Codice	MPD1
Formula	$$PROS = \frac{RO_{soddisfatti}}{RO_{totali}} \cdot 100$$
Descrizione	Percentuale di requisiti obbligatori soddisfatti sul totale. • PROS $= 100\%$: prodotto completo; • $90\% \leq$ PROS $< 100\%$: quasi completo; • $70\% \leq$ PROS $< 90\%$: parzialmente funzionante; • PROS $< 70\%$: non rilasciabile.
Utilità	Determina se il prodotto ha raggiunto il livello minimo di funzionalità necessario per il rilascio.

Tabella 17: MPD1 - Percentuale Requisiti Obbligatori Soddisfatti (PROS)

6.1.0.2 MPD2 – Percentuale Requisiti Opzionali Soddisfatti (PRPS)

Codice	MPD2
Formula	$$PRPS = \frac{RP_{soddisfatti}}{RP_{totali}} \cdot 100$$
Descrizione	Percentuale di requisiti opzionali soddisfatti sul totale. L'implementazione di requisiti opzionali aumenta il valore del prodotto ma non è essenziale per il funzionamento base.
Utilità	Misura il valore aggiunto del prodotto oltre le funzionalità minime richieste.

Tabella 18: MPD2 - Percentuale Requisiti Opzionali Soddisfatti (PRPS)

6.1.0.3 MPD3 – Percentuale Requisiti Desiderabili Soddisfatti (PRDS)

Codice	MPD3
Formula	$$PRDS = \frac{RD_{soddisfatti}}{RD_{totali}} \cdot 100$$
Descrizione	Percentuale di requisiti desiderabili soddisfatti sul totale. I requisiti desiderabili migliorano l'esperienza utente e la competitività del prodotto.
Utilità	Valuta il valore aggiunto fornito dal prodotto oltre le funzionalità minime e opzionali.

Tabella 19: MPD3 - Percentuale Requisiti Desiderabili Soddisfatti (PRDS)

6.2 Affidabilità

6.2.0.1 MPD4 - Line Coverage (LC)

Codice	MPD4
Formula	$$LC = \frac{L_{eseguite}}{L_{totali}} \cdot 100$$
Descrizione	Percentuale di linee di codice eseguite dai test automatici sul totale. • LC $\geq$ 80%: copertura eccellente; • 60% $\leq$ LC $<$ 80%: copertura buona; • 40% $\leq$ LC $<$ 60%: copertura insufficiente; • LC $<$ 40%: copertura critica.
Utilità	Quantifica l'efficacia dei test nell'individuare difetti e identifica porzioni di codice non testate.

Tabella 20: MPD4 - Line Coverage (LC)

6.2.0.2 MPD5 - Branch Coverage (BC)

Codice	MPD5
Formula	$$BC = \frac{B_{eseguite}}{B_{totali}} \cdot 100$$
Descrizione	Percentuale di rami decisionali eseguiti dai test sul totale. • BC $\geq$ 75%: copertura eccellente; • 60% $\leq$ BC $<$ 75%: copertura buona; • 40% $\leq$ BC $<$ 60%: copertura insufficiente; • BC $<$ 40%: rischio elevato.
Utilità	Garantisce che la logica condizionale sia correttamente testata.

Tabella 21: MPD5 - Branch Coverage (BC)

6.2.0.3 MPD6 - Statement Coverage (SC)

Codice	MPD6
Formula	$$SC = \frac{S_{eseguite}}{S_{totali}} \cdot 100$$
Descrizione	Percentuale di istruzioni nel codice eseguite dai test sul totale. • SC $\geq$ 80%: copertura eccellente; • 60% $\leq$ SC $<$ 80%: copertura buona; • 40% $\leq$ SC $<$ 60%: copertura insufficiente; • SC $<$ 40%: molte istruzioni non testate.
Utilità	Garantisce che ogni istruzione sia eseguita almeno una volta, identificando codice non raggiungibile.

Tabella 22: MPD6 - Statement Coverage (SC)

6.3 Usabilità

6.3.0.1 MPD7 - Profondità Massima di Navigazione (PMN)

Codice	MPD7
Formula	$$PMN = \max_{i=1,\dots,k}\{C_i\}$$
Descrizione	Massimo numero di click necessari per completare qualsiasi operazione tra le k analizzate. • PMN ≤ 3: operazioni immediate; • $3 <$ PMN ≤ 5: navigazione buona; • $5 <$ PMN ≤ 7: accettabile; • PMN > 7: inefficiente.
Utilità	Garantisce che nessuna funzionalità sia eccessivamente nascosta o difficile da raggiungere.

Tabella 23: MPD7 - Profondità Massima di Navigazione (PMN)

6.4 Efficienza

6.4.0.1 MPD8 - Tempo Medio di Risposta (TMR)

Codice	MPD8
Formula	$$TMR = \frac{\sum_{i=1}^{k} T_i}{k}$$
Descrizione	Tempo medio tra l'invio di una richiesta al sistema e la ricezione della risposta completa. • TMR ≤ 1 s: eccellente; • $1 <$ TMR ≤ 3 s: buono; • $3 <$ TMR ≤ 5 s: accettabile; • TMR > 5 s: inaccettabile.
Utilità	Garantisce che il sistema soddisfi le aspettative degli utenti in termini di reattività.

Tabella 24: MPD8 - Tempo Medio di Risposta (TMR)

6.5 Manutenibilità

6.5.0.1 MPD9 - Complessità Ciclomatica (CC)

Codice	MPD9
Formula	$$CC = E - N + 2P$$ Dove: E = archi, N = nodi, P = componenti connesse del grafo di controllo di flusso.
Descrizione	Numero di percorsi linearmente indipendenti attraverso il codice sorgente, calcolato per ciascuna funzione o metodo. • CC ≤ 10: complessità bassa, codice manutenibile; • $10 <$ CC ≤ 20: complessità moderata; • $20 <$ CC ≤ 30: complessità elevata; • CC > 30: complessità critica.
Utilità	Misura la complessità delle funzioni per prevenire difficoltà nel testing e nella comprensione del codice.

Tabella 25: MPD9 - Complessità Ciclomatica (CC)

6.6 Portabilità

6.6.0.1 MPD10 - Compatibilità Cross-Browser (CCB)

Codice	MPD10
Formula	$$CCB = \frac{B_{supportati}}{B_{target}} \cdot 100$$
Descrizione	Percentuale di browser target su cui il prodotto funziona correttamente. • CCB = 100%: piena compatibilità; • $80\% \leq$ CCB $< 100\%$: compatibilità buona; • $60\% \leq$ CCB $< 80\%$: compatibilità parziale; • CCB $< 60\%$: compatibilità insufficiente.
Utilità	Garantisce che il prodotto sia utilizzabile dalla maggior parte dei browser target.

Tabella 26: MPD10 - Compatibilità Cross-Browser (CCB)