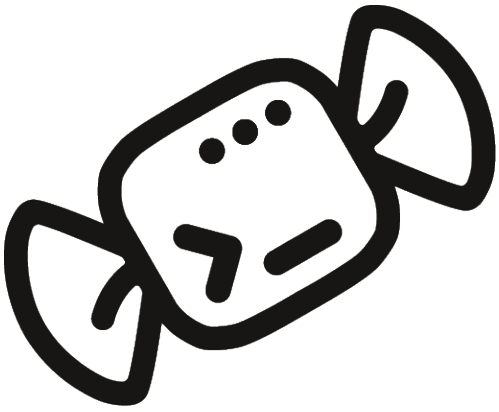

SWEetCode

Norme di progetto

Componenti del gruppo

Bresolin G.
Campese M.
Ciriolo I.
Dugo A.
Feltrin E.
Michelon R.
Orlandi G.

Università di Padova
Ingegneria del Software

Registro delle versioni

Versione	Data	Responsabile di stesura	Revisore	Dettaglio e motivazioni
v1.10.0(9)	$2023-12-29$	Michelon R.	Campese M.	Introduzione Progettazione.
v1.10.0(5)	$2023-12-26$	Bresolin G.	Michelon R.	Aggiornamento configuration management.
v1.10.0(4)	$2023-12-26$	Ciriolo I.	Bresolin G.	Aggiornamento documentazione.
v1.10.0(3)	$2023-12-21$	Bresolin G.	Ciriolo I.	Aggiornamento: sviluppo.
v1.9.0(9)	$2023-12-18$	Ciriolo I.	Bresolin G.	Processi primari: Acquisizione.
v1.9.0(8)	$2023-12-18$	Ciriolo I.	Bresolin G.	Processi primari: Fornitura.
v1.9.0(7)	$2023-12-18$	Bresolin G.	Ciriolo I.	Introduzione: applicazione Standard ISO/IEC 12207:1995.
v1.9.0(6)	$2023-12-17$	Bresolin G.	Ciriolo I.	Introduzione: riferimenti normativi e informativi.
v1.9.0(5)	$2023-12-14$	Ciriolo I.	Bresolin G.	Introduzione: glossario.
v1.0.0(15)	$2023-12-08$	Bresolin G.	Orlandi G.	Aggiornamento ITS: passaggio a Jira di Atlassian. Motivazione: GitHub non offre un diagramma di Gantt in grado di stabilire le dipendenze tra i vari tasks; maggior supporto al framework Scrum attraverso le funzionalità presentate.

Continua nella pagina successiva

Versione	Data	Responsabile di stesura	Revisore	Dettaglio e motivazioni
v1.0.0(11)	$2023-12-02$	Ciriolo I.	Orlandi G.	Processo primario: Sviluppo: attività Analisi dei requisiti.
v1.0.0(10)	$2023-11-27$	Campese M.	Orlandi G.	Ampliamento sezioni Comunicazioni e Strumenti.
v1.0.0(2)	$2023-11-22$	Bresolin G.	Campese M.	Aggiornamento branch protection rules.
v1.0.0(1)	$2023-11-22$	Michelon R.	Bresolin G.	Aggiunte sezioni automazione release e versionamento documenti.
v0.0.1(23)	$2023-11-20$	Michelon R.	Campese M.	Aggiunta parametrizzazione template documenti.
v0.0.1(22)	$2023-11-13$	Bresolin G.	Campese M.	Aggiornamento struttura di documento: modifica ordine registro delle versioni dalla più recente e aggiunta voce "motivazioni"; verbali: introduzione #issue nella tabella delle azioni.
v0.0.1(21)	$2023-11-13$	Bresolin G.	Campese M.	Aggiornamento documentazione: struttura di documento: presenza obbligatoria registro delle versioni e rimozione versione; norme tipografiche: aggiornamento versioni. Aggiunta configuration managment: version control.

Continua nella pagina successiva

Versione	Data	Responsabile di stesura	Revisore	Dettaglio e motivazioni
v0.0.1(20)	$2023-10-30$	Bresolin G.	Michelon R.	Processi di supporto: configuration management: *ITS* e pull request; Processi organizzativi: *GitHub Teams*.
v0.0.1(19)	$2023-10-30$	Campese M.	Michelon R.	Documentazione: strutture di documento: posta elettronica.
v0.0.1(18)	$2023-10-30$	Dugo A.	Bresolin G.	Processi organizzativi: comunicazione.
v0.0.1(13)	$2023-10-28$	Ciriolo I.	Campese M.	Strumenti.
v0.0.1(8)	$2023-10-27$	Bresolin G.	Ciriolo I.	Processo di supporto: documentazione e verifica.
v0.0.1(5)	$2023-10-26$	Campese M.	Orlandi G.	Introduzione.

Indice

1 Introduzione

1.1 Scopo del documento

Questo documento si prefigge lo scopo di normare e documentare un corretto *Way of Working* che verrà seguito lungo l'intero svolgimento del progetto.
Tutti i membri del team sono tenuti a visionare tale documento e a seguire le norme contenute in esso. Tali norme permettono di perseguire e migliorare la coerenza ed omogeneità del lavoro e dei file prodotti. Viene inoltre fornito uno storico delle modifiche e delle versioni del documento per seguire un approccio incrementale nella sua redazione, facilitandone i cambiamenti nel corso del tempo.

2 Riferimenti

2.1 Riferimenti normativi

- *ISO/IEC 12207:1997 Information technology — Software life cycle processes* `https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf`;
- *SEMAT* `https://semat.org/documents/20181/57862/formal-18-10-02.pdf/formal-18-10-02.pdf`;
- *Regolamento del progetto didattico* `https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/PD2.pdf`.

2.2 Riferimenti informativi

- *Glossario*;
- *Analisi dei requisiti*;
- *Piano di progetto*;
- *Piano di qualifica*;
- Capitolato C1: *Knowledge Management AI*
  - `https://www.math.unipd.it/~tullio/IS-1/2023/Progetto/C1.pdf`;
  - `https://www.math.unipd.it/~tullio/IS-1/2023/Progetto/C1p.pdf`.
- Slide lezioni *Ingegneria del software* 2023/2024:
  - `https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/T2.pdf`;
  - `https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/T3.pdf`;
  - `https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/T4.pdf`;
  - `https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/T5.pdf`;
  - `https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/T6.pdf`;
  - `https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/T7.pdf`;
  - `https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/T8.pdf`;
  - `https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/T9.pdf`;

- https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/T10.pdf;
- https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/T11.pdf.

.

3 Glossario

Per evitare ambiguità ed incomprensioni relative al linguaggio e ai termini utilizzati nella documentazione del progetto viene presentato un Glossario. I termini ambigui o tecnici-specifici presenti nello stesso, vengono identificati nei corrispondenti documenti con un pedice |g| e con una scrittura in corsivo. All'interno dei documenti viene identificata con tale scrittura solo e soltanto la prima occorrenza di un termine presente nel Glossario.
All'interno di quest'ultimo i termini vengono ordinati in ordine alfabetico e raggruppati per lettera d'inizio del termine.
I documenti che sono corredati dal Glossario sono:

- Analisi dei requisiti;
- Norme di progetto;
- Piano di progetto;
- Piano di qualifica.

4 Applicazione standard ISO/IEC 12207:1997

In questo documento il team si impegna ad essere conforme allo Standard textitISO/IEC 12207:1997 - Information technology — Software life cycle processes. Questa sezione del documento è dedicata all'applicazione pratica degli standard e dei processi definiti nell'ambito di questo Standard Internazionale.

4.1 Organizzazione dello Standard: processi del ciclo di vita del software

In questa sezione viene presentata l'organizzazione dei processi secondo lo Standard textitISO/IEC 12207:1997 - Information technology — Software life cycle processes del ciclo di vita del software. In questo documento utilizzato per normare il *way of working* del team, viene presentata una organizzazione gerarchica in cui ogni processo è costituito da un insieme di attività, che a loro volta possono presentare delle procedure e un elenco di strumenti utilizzati nel loro svolgimento.

4.1.1 Processi primari del ciclo di vita

Sebbene lo Standard adottato presenti cinque processi primari (*Acquisizione, Fornitura, Sviluppo, Operazione, Manutenzione*), all'interno del contesto del progetto universitario in atto, il team non ritiene il processo *Manutenzione* pertinente e, pertanto, si è deciso di escluderlo dalla presentazione nel documento.
I processi primari presentati nel presente documento sono:

- *Acquisizione*: definisce le attività dell'acquirente, l'organizzazione che acquisisce un prodotto software;

- *Fornitura*: definisce le attività del fornitore, l'organizzazione che fornisce il prodotto software all'acquirente;
- *Sviluppo*: definisce le attività dello sviluppatore, l'organizzazione che definisce e sviluppa il prodotto software.

4.1.2 Processi di supporto del ciclo di vita

I processi di supporto presentati nel presento documento sono:

- *Documentazione*: definisce le attività per la registrazione delle informazioni prodotte da un processo del ciclo di vita;
- *Configuration Management*: definisce le attività di gestione della configurazione;
- *Assicurazione della qualità*: definisce le attività per assicurare in modo oggettivo che i prodotti e i processi software siano conformi ai requisiti specificati e rispettino i piani stabiliti;
- *Verifica*: definisce le attività per verificare il prodotto software;
- *Validazione*: definisce le attività per validare il prodotto software;
- *Revisione congiunta*: definisce le attività per valutare lo stato e i prodotti di un'attività. Questo processo può essere utilizzato da qualsiasi coppia di parti, in cui una parte (la parte che esegue la revisione) valuta un'altra parte (la parte che viene revisionata) in un forum congiunto;
- *Revisione*: definisce le attività per determinare la conformità ai requisiti, piani e contratti;
- *Risoluzione dei problemi*: definisce un processo per analizzare e risolvere i problemi di qualsiasi natura o origine, scoperti durante l'esecuzione di processi.

4.1.3 Processi organizzativi del ciclo di vita

I processi organizzativi presentati nel presento documento sono:

- *Gestione organizzativa*: definisce le attività di management durante i processi del ciclo di vita;
- *Infrastruttura*: definisce le attività base per stabilire una struttura in un processo del ciclo di vita;
- *Miglioramento*: definisce le attività che il team esegue per stabilire, misurare, controllare e migliorare i processi del ciclo di vita;
- *Formazione*: definisce le attività atte a provvedere una adeguata formazione del team.

5 Processi primari

5.1 Acquisizione

Le modalità del processo di acquisizione per il progetto sono state fissate dal docente e sono reperibili all'interno della sezione (§Riferimenti), in (§Riferimenti normativi), in (§Regolamento del progetto didattico).
Le fasi del processo di acquisizione stabilite sono le seguenti:

1. Definizione della necessità di acquisizione di un prodotto software da parte dell'azienda proponente;
2. Formazione dei gruppi di lavoro da parte del docente del corso Vardanega T.(2023-10-14);
3. Presentazione dei capitolati d'appalto da parte dell'azienda proponente nei confronti dei gruppi di lavoro (2023-10-17);
4. Presentazione delle candidature di appalto da parte dei gruppi di lavoro (2023-10-31);
5. Aggiudicazione appalti con assegnazione svolta da parte del docente Vardanega T. in base alle candidature presentate e alle sue valutazioni interne e ad una consultazione con il proponente (2023-11-06);
6. Presentazione del *Proof of Concept (PoC)* da parte del gruppo di lavoro all'azienda proponente;
7. Accettazione da parte dell'azienda proponente del *PoC* proposto dal gruppo di lavoro;
8. Presentazione della candidatura per la revisione *Requirements and Technology Baseline* da parte del gruppo di lavoro;
9. Revisione *Requirements and Technology Baseline* sottoposta dai docenti del corso Vardanega T. e Cardin R. in due parti, con conseguente valutazione;
10. Presentazione del *Minimum Viable Product (MVP)* da parte del gruppo di lavoro all'azienda proponente;
11. Accettazione da parte dell'azienda proponente del *MVP* proposto dal gruppo di lavoro;
12. Presentazione della candidatura per la revisione *Product Baseline* da parte del gruppo di lavoro;
13. Revisione *Product Baseline* sottoposta dai docenti del corso Vardanega T. e Cardin R. in due parti, con conseguente valutazione.

I ruoli definiti all'interno di questo progetto didattico universitario sono:

- Docente del corso: committente;
- Azienda proponente: cliente e mentore;
- Gruppo di lavoro: fornitore.

Le stime dei tempi di consegna e del prezzo viene effettuata dal gruppo di lavoro durante la fase di presentazione della candidatura d'appalto.

La definizione del prezzo durante la fase di candidatura costituisce il limite superiore invalicabile durante lo svolgimento del progetto ed è derivante dai costi (stabiliti dal docente del corso) di ciascun ruolo ricopribile all'interno del gruppo di lavoro.
Le date di consegna sono invece libere e soggette a stringenti regole di ingresso e vengono stabilite dal gruppo di lavoro assieme al proponente, cercando di rispettare il più possibile le stime iniziali stabilite durante la fase di candidatura d'appalto.

5.2 Fornitura

Il processo di fornitura descrive le attività svolte dal fornitore e coinvolge pianificazione, acquisizione e gestione delle risorse necessarie. Il processo segue la determinazione delle procedure e delle risorse necessarie per gestire e garantire il progetto. L'obiettivo del processo è la garanzia dell'efficienza e della conformità ai requisiti del progetto per raggiungere gli obiettivi stabiliti con il proponente.
Viene fissata con il proponente la data di consegna del prodotto finale.
Il processo di fornitura è formato dalle seguenti 7 fasi:

- **Iniziazione**: Il fornitore effettua una revisione dei requisiti nella richiesta di proposta;
- **Preparazione della risposta**: Il fornitore definisce e prepara una proposta in risposta alla richiesta di proposta;
- **Negoziazione**: Il fornitore negozia e stipula un accordo con il proponente per fornire il prodotto finale;
- **Pianificazione**: Il fornitore effettua una revisione dei requisiti di acquisizione e valuta le opzioni per lo sviluppo del prodotto software in base ad un'analisi dei rischi associati a ciascuna opzione per definire la struttura del piano di gestione del progetto e per garantire la qualità del prodotto finale (il fornitore deve sviluppare e documentare tale piano);
- **Esecuzione e controllo**: Il fornitore esegue il piano di gestione del progetto e monitora e controlla il progresso e la qualità del prodotto durante l'intero ciclo del progetto;
- **Revisione e valutazione**: Il fornitore coordina le attività di comunicazione con il proponente e conduce o supporta riunioni informali, revisioni di accettazione, test di accettazione, revisioni congiunte. Il fornitore esegue la verifica e la convalida del processo per dimostrare che i prodotti o servizi software e i processi soddisfano appieno i rispettivi requisiti;
- **Consegna e completamento**: Il fornitore consegna il prodotto finale, e dovrà fornire assistenza al proponente a supporto del prodotto consegnato.

5.2.1 Gestione

Per garantire un processo di fornitura efficace ed efficiente viene mantenuta la comunicazione con l'azienda proponente durante tutto il corso del progetto, mediante riunioni settimanali su *Google Meet* pianificate con *Google Calendar* e scambio di messaggi qualora fosse necessario attraverso la piattaforma *Slack*. In questo modo al fornitore risulta possibile un'identificazione accurata dei bisogni del proponente con conseguente individuazione accurata dei requisiti e dei vincoli del progetto. Il dialogo

continuo con il proponente permette infine una valutazione continua e costante del lavoro svolto dal fornitore, in modo da permettere correzioni, integrazioni e miglioramenti in modo incrementale e costruttivo.

5.2.2 Piano di Progetto

Il Piano di progetto viene redatto dal responsabile con l'aiuto degli amministratori. Fornisce una guida dettagliata per la pianificazione, l'esecuzione e il controllo del progetto e serve come base per il monitoraggio del progresso, la gestione dei rischi e la comunicazione tra proponente e fornitore.
Il suo contenuto comprende:

- Calendario di progetto;
- Stima dei costi di realizzazione;
- Rischi e mitigazione;
- Pianificazione e modello di sviluppo;
- Preventivo e consuntivo;
- Retrospettiva.

5.2.3 Piano di Qualifica

Il Piano di qualifica definisce le strategie, gli obiettivi e le attività per garantire la qualità del prodotto finale. Vengono definite in tale documento le metriche di valutazione e validazione del progetto e la specifica degli obiettivi di qualità del prodotto finale. Tali parametri vengono stabiliti in accordo ai requisiti e alle aspettative del proponente e talvolta a discrezione del team sulla base delle valutazioni fatte nel corso di studi. Il suo scopo è assicurare che il prodotto soddisfi gli standard di qualità definiti e che eventuali deviazioni vengano gestite in modo appropriato.
Il suo contenuto comprende:

- Qualità di processo;
- Qualità di prodotto;
- Test e specifiche;
- Valutazioni per il miglioramento;
- Resoconto delle attività di verifica.

5.2.4 Strumenti

- **Gmail**: servizio di posta elettronica fornito da *Google*, utilizzato per le prime comunicazioni tra proponente e fornitore;
- **Google Calendar**: sistema di calendari usato per pianificare riunioni, scadenze e in generale eventi significativi tra proponente e gruppo;
- **Google Sheet**: parte del Google Workspace, fornisce dei fogli di calcolo utilizzati per la creazione di tabelle e grafici utili in particolare al Piano di progetto;

- **Slack**: piattaforma di messaggistica specializzata nella comunicazione e collaborazione all'interno di team aziendali. Viene utilizzata dal team per le comunicazioni asincrone con il proponente.

5.3 Sviluppo

Il processo di sviluppo si pone lo scopo di descrivere tutte le attività e i compiti che portano un prodotto dalla fase di ideazione a quella di realizzazione. Per tale motivo vengono dunque presentate le seguenti attività:

- Analisi dei requisiti;
- Progettazione logica;
- Progettazione di dettaglio;
- Codifica;
- Testing;
- Integrazione software;
- Installazione software.

5.3.1 Analisi dei requisiti

L'attività di analisi dei requisiti, la quale viene documentata all'interno del documento (§Analisi dei requisiti), viene svolta seguendo le seguenti fasi:

- Studio accurato del capitolato e delle esigenze del committente;
- Individuazione dei casi d'uso (e relativa produzione dei diagrammi dei casi d'uso) e dei requisiti;
- Discussione con il proponente del materiale prodotto;
- Divisione dei requisiti nelle tre categorie individuate e applicazione degli spunti emersi dalla discussione col proponente.

(L'analisi dei requisiti, essendo una attività incrementale, prevede che alcune fasi siano svolte più volte all'interno del progetto). Il risultato di tale processo è un documento che contiene tutti i requisiti richiesti dal proponente, con relativi casi d'uso dal punto di vista dell'utente finale.

5.3.1.1 Casi d'uso: Notazione

I casi d'uso sono indicati con la notazione seguente: **UC[Codice] - [Titolo]**
in cui:

- **UC** sta per Use Case;
- [**Codice**] è l'identificativo del caso d'uso. È composto da un unico numero progressivo univoco se il caso d'uso non ha padre, mentre se si tratta di un sottocaso d'uso, segue il formato **[Codice_padre].[Numero_figlio]**; questa struttura è ricorsiva, quindi non pone un limite alla profondità della gerarchia;
- [**Titolo**] è il titolo del caso d'uso.

5.3.1.2 Casi d'uso: Didascalie

Per ogni caso d'uso è presentata una breve didascalia che ne indica:

- Attore principale (attore che partecipa e dialoga attivamente con il sistema nello scenario principale per raggiungere uno scopo);
- Attore secondario (attore esterno al sistema che, se presente, aiuta il sistema a raggiungere l'obiettivo dell'attore principale);
- Precondizioni (condizioni che devono essere vere affinché il caso d'uso possa essere eseguito);
- Postcondizioni (condizioni del sistema dopo che il caso d'uso si è concluso);
- Scenario principale (sequenza di passaggi che definisce un'interazione tra attore e sistema);
- Scenario alternativo (scenario che si verifica se si diverge dallo scenario principale);
- Estensioni (lista di casi d'uso che deviano il flusso principale del caso d'uso in esame, introducendo flussi alternativi);
- Inclusioni (lista di casi d'uso che rappresentano una parte dello scenario principale, ma estratti come casi d'uso singoli per aumentare il riuso nel caso in cui appaiono in scenari principali di più casi d'uso);
- Generalizzazioni (lista di casi d'uso che ereditano le caratteristiche del caso d'uso in esame aggiungendo specificità);
- Trigger (evento scatenante del caso d'uso).

5.3.1.3 Requisiti: Notazione

Ogni requisito preso in analisi sarà trattato con la sigla: **R[Tipo].[Importanza].[Codice]** nella quale:

- [**R**] indica la parola "requisito";
- [**Tipo**] può essere:
  - F (Funzionale);
  - Q (Qualità);
  - V (Vincolo);
  - V (Prestazione).
- [**Importanza**] classifica i requisiti in:
  - O (Obbligatorio);
  - D (Desiderabile);
  - OP (Opzionale).
- [**Codice**] identifica i requisiti per ogni tipologia. È composto da un unico numero progressivo univoco se il requisito non ha padre, mentre se si tratta di un sottorequisito, segue il formato **[Codice_padre].[Numero_figlio]**; questa struttura è ricorsiva, quindi non pone un limite alla profondità della gerarchia;

5.3.1.4 Requisiti: Suddivisione

- I requisiti funzionali descrivono le funzionalità del sistema, le azioni che il sistema può compiere e le informazioni che il sistema può fornire.
  Seguendo la notazione riportata sopra, questi si possono partizionare in:
  - RF.O - Requisito Funzionale Obbligatorio;
  - RF.D - Requisito Funzionale Desiderabile;
  - RF.OP - Requisito Funzionale Opzionale.
- I requisiti di qualità descrivono come un sistema deve essere, o come il sistema deve esibirsi, per soddisfare le esigenze dell'utente.
  Seguendo la notazione riportata sopra, questi si possono partizionare in:
  - RQ.O - Requisito di Qualità Obbligatorio;
  - RQ.D - Requisito di Qualità Desiderabile;
  - RQ.OP - Requisito di Qualità Opzionale.
- I requisiti di vincolo descrivono i limiti e le restrizioni normative/legislative che un sistema deve rispettare per soddisfare le esigenze dell'utente.
  Seguendo la notazione riportata sopra, questi si possono partizionare in:
  - RV.O - Requisito di Vincolo Obbligatorio;
  - RV.D - Requisito di Vincolo Desiderabile;
  - RV.OP - Requisito di Vincolo Opzionale.
- I requisiti di prestazione descrivono vincoli di tempo e spazio che il prodotto deve rispettare.
  Seguendo la notazione riportata sopra, questi si possono partizionare in:
  - RP.O - Requisito di Prestazione Obbligatorio;
  - RP.D - Requisito di Prestazione Desiderabile;
  - RP.OP - Requisito di Prestazione Opzionale.

5.3.1.5 Diagrammi

I diagrammi dei casi d'uso e di attività riportati all'interno del documento (Analisi dei requisiti) devono seguire la notazione e le specifiche indicate da *UML v2.5*.

5.3.1.6 Applicazione milestone SEMAT

SWEetCode nell'esecuzione dell'attività di analisi dei requisiti ha deciso di adottare la pianificazione in milestone presentata da *SEMAT*: tale pianificazione viene spiegata nel dettaglio all'interno del (§Piano di progetto).

5.3.2 Progettazione

5.3.2.1 Scopo L'attività di progettazione ha lo scopo di fissare un'architettura del prodotto che soddisfi i requisiti determinati dall'attività di analisi, prima di passare alla codifica. La progettazione serve a dominare la complessità del prodotto, decomponendolo in parti componibili e organizzate.

I progettisti del team avranno l'obbligo di considerare le seguenti proprietà durante l'attività di progettazione:

- **Sufficienza**: in grado di soddisfare tutti i requisiti;
- **Modularità**: suddivisa in parti chiare e ben distinte, esponendo interfacce e nascondendo l'implementazione;
- **Robustezza**: resiliente a più tipi e quantità di input;
- **Flessibilità**: permette modifiche mantenendo limitati i costi per apportarle;
- **Riusabilità**: permette il riuso delle sue parti in applicazioni esterne;
- **Efficienza**: è efficiente nello spazio, nel tempo e nei consumi;
- **Affidabilità**: svolge bene il suo lavoro;
- **Semplicità**: ogni parte contiene lo stretto necessario;
- **Incapsulazione**: nasconde l'interno delle componenti architetturali all'esterno;
- **Coesione**: parti che agiscono sulle stesse unità di dati sono raccolte nello stesso componente;
- **Basso accoppiamento**: limita le dipendenze tra parti diverse;

Seguendo il modello a V, l'attività di progettazione si divide in due fasi: progettazione logica e progettazione di dettaglio.

5.3.2.2 Progettazione logica La progettazione logica consiste nel trasformare i requisiti software in un'architettura che descrive la struttura di alto livello del prodotto software e ne identifica i componenti. In questa fase i progettisti devono assicurarsi che tutti i requisiti siano assegnati ai componenti e che questi siano ulteriormente dettagliati, in modo da facilitare la successiva progettazione di dettaglio.

In particolare, i progettisti devono:

- Progettare le interfacce esterne al software e tra i componenti del software;
- Definire e progettare le strutture dati necessarie per soddisfare i requisiti;
- Documentare una versione preliminare del (Manuale utente);
- Pianificare e definire, collaborando con i verificatori, i test di integrazione tra componenti;
- Valutare e revisionare l'architettura definita nei passi precedenti, coinvolgendo tutti i membri del team e altre parti interessate. In particolare, il team collaborerà con dei membri dell'azienda proponente in modo da ottenere consigli e correzioni da fonti autorevoli e con esperienza.

5.3.2.3 Progettazione di dettaglio La progettazione di dettaglio consiste nel suddividere il sistema affinché ogni singola parte abbia una complessità tale da poter essere codificata da un unico individuo, in modo rapido, fattibile e verificabile.

I progettisti hanno il dovere di fermarsi quando la decomposizione diventa controproduttiva, cioè nel momento in cui il costo di coordinamento tra le parti decomposte è maggiore del beneficio prodotto decomponendo.

La progettazione di dettaglio agisce sulle unità architetturali, cioè unità funzionali ben definite, la cui codifica può essere assegnata ad un unico programmatore. Ogni unità architetturale è formata da uno o più moduli, e più unità architetturali formano un componente, che è l'oggetto dell'architettura logica. L'architettura di dettaglio guida la codifica e l'integrazione e semplifica il tracciamento dei requisiti.

Durante la progettazione di dettaglio, i progettisti hanno il compito di:

- Definire e progettare in modo dettagliato i componenti software, lavorando a livello di unità e moduli;
- Aggiornare e dettagliare il (Manuale utente);
- Definire la specifica dei test di unità, collaborando con i verificatori;
- Definire la specifica dei test di integrazione, collaborando con i verificatori;

5.3.2.4 Specifica architetturale La progettazione deve essere ben documentata, affinché la codifica e la verifica del software possano essere svolte in modo autonomo e disciplinato.

Ogni parte individuata deve essere dodata di una specifica chiara e coesa, realizzabile entro i costi e le risorse disponibili.

Queste specifiche sono raccolte nel documento (Specifica architetturale), che verrà prodotto dopo il superamento della fase di revisione RTB. Le modalità di stesura e l'organizzazione di questo documento verranno indicate in parallelo alla sua creazione.

5.3.3 Codifica

Tale sezione sarà realizzata a seguito della versione v2.0.0 delle (§Norme di progetto).

5.3.4 Testing

Tale sezione sarà realizzata a seguito della versione v2.0.0 delle (§Norme di progetto).

5.3.5 Integrazione software

Tale sezione sarà realizzata a seguito della versione v2.0.0 delle (§Norme di progetto).

5.3.6 Installazione software

Tale sezione sarà realizzata a seguito della versione v2.0.0 delle (§Norme di progetto).

6 Processi di supporto

Nella seguente sezione vengono presentati i processi di supporto allo sviluppo del progetto.

6.1 Configuration management

6.1.1 Scopo

Nella seguente sezione vengono presentate le attività svolte dal gruppo SWEetCode che rientrano all'interno del processo "configuration management".

6.1.2 Descrizione

Il processo di configuration management è un processo di applicazione di procedure amministrative e tecniche lungo l'intero ciclo di vita del software al fine di:

- Identificare, definire e stabilire una base per gli elementi software in un sistema;
- Controllare le modifiche e le release degli elementi;
- Registrare e riportare lo stato degli elementi e delle richieste di modifica;
- Garantire la completezza, la coerenza e la correttezza degli elementi.

6.1.3 Configuration control

6.1.3.1 Scopo L'attività di configuration control si pone i seguenti obiettivi:

- Identificare e registrare le richieste di modifica;
- Analizzare e valutare le modifiche;
- Approvare o rifiutare le richieste di modifica;
- Tracciamento di audit.

6.1.3.2 Issue tracking system (ITS) Per raggiungere l'obiettivo di tracciamento di audit, ovvero la registrazione sistematica e dettagliata dei task e delle modifiche, SWEetCode utilizza l'*Issue Tracking System Jira* fornito da *Atlassian*: *Jira*.

6.1.3.2.1 Ticket In questo sistema di tracciamento si permette la creazione di *ticket* che possono essere di tre tipi:

- *task*: tale tipo di ticket viene utilizzato dal team per assegnate compiti atomici ai vari membri del gruppo;
- *bug*: tale tipo di ticket viene utilizzato dal team per assegnare un compito di fix di un bug all'interno del progetto ad un membro del gruppo;
- *story*: tale tipo di ticket viene utilizzato per rappresentare un requisito.

Ciascun *ticket* presenta (di seguito sono riportate le sole funzionalità utilizzate dal team):

- Titolo: un titolo breve ma esplicativo del compito associato al *ticket*;
- Id numerico univoco: tale id viene generato in automatico dall'ITS nella forma *SWE-ID*;

- *epic*: ogni *ticket* può essere infatti associato ad una *epic*;
- Collegamento: *Jira* mette a disposizione una funzionalità di collegamento per stabilire le dipendenze tra i vari *ticket*;
- Assegnatario: il membro del team che a cui è stata affidata la responsabilità di svolgere il *ticket*;
- Descrizione: una breve descrizione inerente al *ticket* associata;
- Data: la data di creazione del *ticket*;
- Reporter: il membro del team che ha creato il *ticket*;
- Sviluppo: voce contenente il link ai commit, alle pull request e alle build che sono stati effettuati nel repo su *GitHub* in correlazione al *ticket* in questione;
- Stato: gli stati possibili di un *ticket* sono *Da fare*, *In Progress*, *Pronto per commit* (introdotto dagli amministratori di SWEetCode nel flusso di lavoro) ed infine *Completato*;
- *Sprint*: ogni *ticket* può essere associato ad uno *sprint*;
- *Versione* di correlazione: ogni *ticket* può essere associata ad una determinata *versione*.

6.1.3.2.2 Epic Un insieme di *ticket* può essere raggruppato all'interno di una *epic*: una *epic* costituisce un'unità logica e strategica di lavoro più ampia, rappresentando un obiettivo o un risultato di alto livello. Questo strumento di lavoro offre al team una panoramica di completamento generale di un particolare insieme di azioni correlate tra loro, attraverso una scoreboard che indica la percentuale di *ticket* completati, in progresso o ancora da iniziare. Oltre ai *ticket* associati, ogni *versione* presenta i medesimi field di un *ticket* precedentemente descritti.

6.1.3.2.3 Versioni Per rappresentare le milestone su *Jira* vengono utilizzate le *Versioni*, alle quali possono essere associate *epic* e i relativi *ticket*, riportando come nelle *epic* una scoreboard che indica la percentuale di *ticket* completati, in progresso o ancora da iniziare. Una volta che tutti i *ticket* associati alla *versione* sono stati completati, tale *versione* può essere rilasciata. Oltre ai *ticket* associati, ogni *versione* presenta una breve descrizione, una data di inizio ed una di fine.

6.1.3.2.4 Backlog e Sprint *Jira* offre all'interno dell'*ITS* differenti funzionalità di supporto al metodo agile, seguendo il framework Scrum, in particolare quando i *ticket* vengono creati sono inseriti di default all'interno del *Backlog* per essere in seguito assegnati ai vari *Sprint*. Ciascun *Sprint* è caratterizzato da una data di inizio ed una di fine e da uno stato (*Attivato*/*Chiuso*).

6.1.3.2.5 Timeline *Jira* mette a disposizione come strumento una *timeline* realizzata attraverso un diagramma di Gantt all'interno del quale poter inserire i *ticket* creati associati ad una *epic*.
Ciascun *ticket* può dunque essere disposto in uno spazio temporale, facendo riferimento ad una data di inizio ed una di fine, costituendo il periodo all'interno del quale portare a termine il compito associato. Tra i vari *ticket* è inoltre possibile creare collegamenti che rappresentano graficamente le dipendenze presenti tra essi.

Ogni *versione* viene rappresentata all'interno della *timeline* da una linea verticale posta in corrispondenza della data di scadenza della versione stessa.
Nella parte superiore della *timeline* sono inoltre visualizzati i vari *Sprint* presenti all'interno dell'ITS, visualizzando in opaco quelli chiusi.
Internamente alla *timeline* è inoltre resa disponibile una ulteriore funzionalità di filtraggio dei *ticket* visualizzati in base alle *versioni* e/o alle *epic*, permettendo di effettuare una istantanea.

6.1.3.2.6 Automazione chiusura ticket *Jira* mette a disposizione una funzionalità di creazione di automazioni (chiamate *Trigger*) configurabili dell'utente, permettendo di adattare al meglio ciascuna esigenza. Il team ha deciso di introdurre un'automazione di chiusura automatica dei ticket presenti all'interno dell'*ITS*: ogni volta che un membro del team effettua un commit nel repository finalizzato alla chiusura di un *ticket*, il contenuto di tale commit, grazie all'esecuzione di uno script Python creato dal team (Norme di progetto, §3.2.2 Automazione versionamento documenti) presenterà al suo interno l'operazione "*SWE-#id_ticket*".
A seguito dell'approvazione della pull request da parte del verificatore, il *ticket* associato a quel particolare *ID* passerà in automatico allo stato *Completato*.

6.1.3.3 Pull requests A seguito dell'adozione dei *GitHub Teams*, SWEetCode propone l'utilizzo delle pull requests per permettere un rilascio controllato che permetta di analizzare e valutare le richieste di modifica effettuate: l'unico *Team* dotato dei permessi necessari all'approvazione di una pull request sono i *Verificatori*.
Una volta proposta la pull request il verificatore attuale del progetto, membro del *Team Verificatori*, dovrà effettuare una verifica sulle modifiche proposte e, a seguito delle sue valutazioni, dovrà decidere se declinare o accettare la pull request, con conseguente merge nel branch all'interno del quale la pull request è stata sollevata in caso di esito positivo. Tali responsabilità vengono definite tramite l'utilizzo di *Branch Protection Rules* offerte da *GitHub*. L'identificazione attraverso un titolo fornito da colui che solleva la pull request e la registrazione delle richieste di modifica e del loro esito è una conseguenza dell'applicazione della pratica delle pull request offerta da *GitHub*.

6.1.3.4 GitHub Teams SWEetCode, attraverso un profilo *Organizations*, ha deciso di utilizzare una struttura interna ai membri del gruppo che usufruisce dei *Teams* di *GitHub*: ogni ruolo presente all'interno del progetto porta alla creazione di un *Team* avente il medesimo nome.
I *Teams* presentati sono dunque:

- Amministratori;
- Analisti;
- Progettisti;
- Programmatori;
- Responsabili;
- Verificatori.

Lo strumento *Teams* consente di gestire i vari ruoli nel migliore dei modi, permettendo di assegnare responsabilità e permessi specifici a determinate posizioni piuttosto

che ai membri del gruppo. Questa mansione altrimenti risulterebbe dispendiosa e disagevole a causa della rotazione dinamica interna dei ruoli.

6.1.4 Configuration status accounting

6.1.4.1 Scopo L'obiettivo che si pone il configuration status accounting è quello di mantenere una registrazione accurata e aggiornata del prodotto software e dei suoi elementi in relazione allo stato del prodotto stesso nel corso del tempo.

6.1.4.2 Version control Per tenere traccia dello stato e della storia degli elementi controllati, il team ha deciso di utilizzare un sistema di *Version control* basato su una versione del prodotto che si attiene al seguente formato: vX.Y.Z(build).

- v: sta per versione, rimane immutato lungo tutte le versioni;
- X: numero che indica la versione principale di riferimento. Viene incrementato quando viene superata una fase di revisione di avanzamento;
- Y: numero che viene incrementato con l'introduzione di nuove features o di miglioramenti significativi;
- Z: numero che indica piccoli cambiamenti rispetto alla versione vX.Y, come il fix di bug o il caricamento di nuovi documenti all'interno del repository (ad esclusione di verbali, sia interni che esterni);
- (build): numero che indica le build di progetto eseguite nella versione vX.Y.Z. Tale valore viene incrementato a seguito di modifiche/aggiunte nella documentazione.

Il sistema di numerazione delle versioni ha come prima versione il valore "v0.0.1(0)".
Ogni volta che viene incrementato il valore X, i valori Y, Z e (build) ripartono dai loro valori iniziali, ovvero "0".
Ogni volta che viene incrementato il valore Y, il valore Z e (build) ripartono dal valore "0".
Ogni volta che viene incrementato il valore Z, il valore (build) riparte dal valore "0".

6.1.5 Configuration evaluation

6.1.5.1 Scopo L'attività di configuration evaluation si pone l'obiettivo di garantire la completezza funzionale degli elementi software realizzati rispetto ai loro requisiti.

6.1.5.2 Tracciamento dei requisiti Per raggiungere l'obiettivo precedentemente citato, il team SWEetCode ha deciso di effettuare un tracciamento dei requisiti all'interno del prodotto software, in modo da portare nel concreto l'evidenza della correlazione tra requisiti e codice.
Per effettuare il tracciamento il team apporta un commento, contenente l'identificativo del requisito, antecedente alla parte di codice che porta al soddisfacimento del requisito menzionato.

6.1.6 Release management

6.1.6.1 Scopo L'attività di release management si occupa di effettuare un rilascio controllato.

6.1.6.2 Automazione release L'automazione del versionamento consente di rilasciare nuove release in modo automatico, seguendo le regole di cambio versione indicate in (Norme di progetto, §Version control). Attraverso l'uso delle *GitHub Actions*, l'accettazione di una pull request, eseguita seguendo le norme indicate nel paragrafo (Norme di progetto, §Pull requests), è immediatamente seguita dalla pubblicazione della nuova versione. Il tipo di cambio versione viene indicato dal prefisso del titolo della Pull Request.

Questa automazione si integra in modo efficace con quella citata nel paragrafo (Norme di progetto, §Automazione versionamento documenti). Ciò che separa l'esecuzione consecutiva delle due automazioni è la creazione della pull request e la sua accettazione.

La richiesta dell'intervento umano, in questo caso, non è un fattore limitante, ma rappresenta due ulteriori fasi di verifica, che garantiscono la pulizia del repository remota e la correttezza del suo contenuto.

6.1.6.3 Automazione versionamento documenti L'aggiornamento del registro delle versioni, la creazione dei pdf e la pubblicazione delle modifiche nella repository remota avvengono in modo consecutivo e automatico grazie all'esecuzione di uno script Python creato dal team. Questo script richiede inizialmente l'inserimento di alcuni input riguardanti le modifiche, compresa l'eventuale chiusura del *ticket* soddisfatto, e dopo una fase di elaborazione in cui aggiorna il registro delle versioni e crea il pdf usando il programma *Latexmk*, esegue il push in remoto dei cambiamenti introdotti in locale.

Il componente del gruppo che ha eseguito lo script dovrà semplicemente recarsi nel sito web della repository su GitHub e creare la Pull Request. Questa verrà successivamente analizzata e accettata o rifiutata in base ai criteri descritti nel (Piano di qualifica) e secondo le modalità citate nel paragrafo (§Produzione).

L'utilizzo di questa automazione permette di ridurre gli errori legati all'aggiornamento e alla pubblicazione di nuove versioni dei documenti, e riduce notevolmente il carico di lavoro assegnato ai componenti del gruppo, limitando volutamente le loro azioni alla sola modifica del contenuto della documentazione.

6.2 Documentazione

Tutte le attività di sviluppo sono accompagnate da una loro documentazione, con l'obiettivo di permettere una consultazione semplice e rapida alle informazioni inerenti al prodotto e alle attività stesse, svolgendo inoltre un ruolo di storicizzazione e di supporto alla manutenzione.

6.2.1 Implementazione del processo

In questa sezione viene documentato e implementato un piano che identifica i documenti da produrre durante il ciclo di vita del prodotto software. Tutti i documenti da redigere vengono presentati nella tabella che segue. Vengono inclusi solamente i documenti relativi al vero e proprio ciclo di vita del prodotto software e sono di conseguenza esclusi i documenti presentati per la candidatura per il progetto didattico, quali "Lettera di presentazione", "Preventivo costi e impegni orari" e "Valutazione dei capitolati d'appalto".

Nome del documento	Scopo	Redattore	Destinatari	Consegne previste
Analisi dei requisiti	Definizione requisiti utente	Analista	Azienda proponente, Docenti	RTB, PB
Norme di progetto	Definizione regolamento normativo per il team	Responsabile, Amministratore	Docenti	RTB, PB
Piano di progetto	Definizione temporale scadenze e progressi	Responsabile	Docenti	RTB, PB
Piano di qualifica	Definizione qualità e testing del prodotto finale	Amministratore	Docenti	RTB, PB
Verbali esterni	Tracciamento riunioni esterne	Responsabile	Azienda proponente, Docenti	Candidatura, RTB, PB
Verbali interni	Tracciamento riunioni interne	Responsabile	Docenti	Candidatura, RTB, PB

6.2.2 Progettazione e sviluppo

In questa sezione vengono presentati gli standard e le regole (nello specifico di stile) a cui i membri di SWEetCode si attengono relativamente alla stesura di documenti relativi al progetto.

6.2.2.1 Template Il team ha deciso di produrre un template grafico ricorrente all'interno della documentazione.
Ogni documento presenta la medesima pagina di copertina, con una grafica realizzata tramite *Adobe Illustrator* ed in seguito disposta nella pagina tramite *Overleaf*, dove viene esposto il nostro logo, il nostro nome del team e i membri del team.
Le seguenti pagine di ciascun documento presentano un:

- ***Header***: intestazione contenente il nostro logo, il nome del documento ed il nostro nome;
- ***Footer***: piè di pagina contenente i link alla nostra email e al nostro account *GitHub*; ed infine la numerazione di pagina.

6.2.2.2 Parametrizzazione template Per facilitare e velocizzare la procedura di inserimento dei dati all'interno dei documenti prodotti con il linguaggio di marcatura

Latex, il gruppo ha deciso di creare dei template parametrizzati, sfruttando l'uso di comandi già presenti nel linguaggio, come *\def* e *\newcommand*, e l'uso di librerie esterne, come *ifthen* e *forloop*. Considerando i valori dei parametri, durante la fase di compilazione la struttura dei documenti si adatterà in modo dinamico, assumendo l'aspetto del tipo di documento indicato, ad esempio verbale interno, esterno o altri tipi di documenti più discorsivi. L'adozione di questa pratica permette al team di mantenere coerenti struttura, formato del contenuto e presentazione, aumentando il throughput e consentendo ai componenti del gruppo di concentrarsi più sul contenuto che sulla sua visualizzazione.

Di seguito vengono elencati i parametri utilizzati a seconda del tipo di documento:

- Titolo;
- Data;
- Orario inizio / fine;
- Luogo;
- Tipo di verbale (esterno o interno);
- Nome di responsabile, verificatore, segretario ed eventuale azienda;
- Firma di responsabile ed eventuale azienda;
- Lista dei partecipanti (interni e esterni);
- Lista revisione delle azioni;
- Lista ordine del giorno;
- Lista discussione (interna o esterna);
- Lista decisioni.

6.2.2.3 Struttura di documento Tutti i documenti prodotti da SWEetCode presentano una struttura comune:

- **Pagina di copertina**: descritta nella sezione Template precedente;
- **Registro delle versioni**: presente in tutti i documenti ad eccezione dei verbali. Questo registro utilizzato per tenere traccia delle varie versioni permette di comprendere in modo rapido chi ha realizzato determinate sezioni della documentazione, quando sono state realizzate, cosa è stato aggiunto e perché (sotto la voce "Dettaglio e motivazioni"). Il registro presenta le versioni ordinate a partire dalla versione più recente;
- **Indice**: presente nel caso in cui si abbia un documento di notevole dimensione, dotato di sezioni. Il suo obiettivo è quello di facilitare e agevolare l'accesso ad un determinato contenuto trattato nel documento;
- **Contenuto**: il contenuto vero e proprio del documento.

6.2.2.4 Verbali Fanno eccezione alla struttura precedentemente esposta i verbali, sia esterni che interni. Essi infatti oltre a presentare nella prima pagina di copertina i ruoli inerenti alla loro produzione (al posto dell'elenco dei membri del team) ed una sezione apposita per la firma del responsabile di riunione (e in caso di verbale esterno

anche la firma da parte di un rappresentate dell'azienda), offrono una struttura particolare.
Il loro contenuto presenta le seguenti sezioni:

- **Intestazione**: sezione contenente le informazioni inerenti a data, ora e luogo della riunione;
- **Partecipanti**: sezione contenente l'elenco in ordine alfabetico dei partecipanti interni ed eventualmente esterni alla riunione e una voce apposita per gli assenti;
- **Revisione delle azioni**: sezione contenente un elenco cronologico delle azioni assegnate nelle riunioni precedenti, con informazioni in merito al progresso, problematiche riscontrate ed eventuale conclusione di tale azione;
- **Ordine del giorno**: elenco cronologico e approssimativo delle tematiche trattate all'interno della riunione;
- **Discussione**: sezione contente l'elenco cronologico delle tematiche affrontate, trattate con una descrizione accurata delle informazioni emerse;
- **Decisioni**: decisioni prossime e/o future da intraprendere o intraprese a seguito della riunione affrontata;
- **Azioni da intraprendere**: tabella contente le azioni assegnate durante l'incontro e da svolgere nel futuro immediato. Ogni voce è accompagnata dal numero del *ticket* associato (se presente) nell'ITS su *Jira*, un incaricato, da un revisore che dovrà verificare il suo operato e da una data di scadenza entro la quale l'azione da svolgere dovrà essere ritenuta completata;
- **Altro**: sezione contente informazioni aggiuntive quali riferimenti esterni e la data della prossima riunione (nel caso in cui essa sia stata stabilita).

6.2.2.5 Posta elettronica La posta elettronica rappresenta uno strumento fondamentale di comunicazione asincrona con proponenti e committente.
Per uniformare lo stile delle mail, vengono descritti di seguito delle norme a cui attenersi nella loro stesura:

- **Oggetto**: campo da riempire obbligatoriamente in maniera breve e concisa;
- **Firma**: ogni mail viene conclusa con la firma digitale del gruppo collocata automaticamente;
- **Forma**: le informazioni e le richieste più importanti vanno posizionate all'inizio del messaggio. La lunghezza del testo della mail non deve essere eccessivo, onde evitare la perdita di visibilità di informazioni preziose; qualora fosse necessario un testo complesso e denso di informazioni, ci si deve servire dell'uso di paragrafi ed elenchi per strutturarlo;
- **Stile**: è buona norma seguire lo stile del mittente. La mail sarà dotata di formule di saluto iniziale e finale quali ad esempio: *"Gentile.."*, *"Spettabile.."*, *"Cordiali saluti"* e *"Distinti saluti"*.

6.2.2.6 Norme tipografiche

- **Citazioni tecnologiche e strumentali**: Ogni tecnologia e/o strumento menzionato all'interno della documentazione deve essere scritta in *corsivo*.

- **Date**: Tutte le date presenti nella documentazione prevedono il seguente formato: AAAA-MM-GG.
- **Elenchi**: Tutti gli elenchi presenti nella documentazione prevedono un ordine:
  - **Alfabetico**: sono puntati e non presentano alcuna correlazione con l'aspetto temporale;
  - **Cronologico**: sono numerati e descrivono una serie di eventi o argomenti ordinati nel tempo.

  In tutti gli elenchi ogni punto termina con il ";", fatta eccezione per l'ultimo elemento che termina con il ".".
- **Menzioni**: Ogni menzione di una persona, interna od esterna al team, avviene nel seguente formato: Cognome N. (dove la lettera N sta per l'iniziale del nome).
- **Nome del documento**: I nomi dei documenti prevedono la lettera iniziale maiuscola seguiti dal resto dei caratteri in case minuscolo e dalla versione del documento stesso.
  I verbali, sia interni che esterni, fanno eccezione e presentano un titolo composta dalla data in cui sono stati svolti nel seguente formato: AAAA-MM-GG.
- **Riferimenti interni**: Menzioni e riferimenti a sezioni interne allo stesso documento a cui appartengono vengono riportate seguendo la notazione (§Nome sezione). Questi riferimenti saranno opportunamente collegati tramite link al paragrafo indicato.
- **Riferimenti esterni**: Menzioni e riferimenti a sezioni di documenti esterni vengono riportate seguendo la notazione (Nome documento, §Nome sezione)
- **Nome del gruppo**: Il nome del gruppo presenta il seguente formato: "SWEetCode".
- **Stile tipografico**: La formattazione del testo dei documenti segue rigorosamente un unico stile tipografico: *Poppins*.
- **Versioni**: Ogni documento viene associato nel nostro sito GitHub alla versione del progetto per cui risulta essere ancora in vigore.
  All'interno del documento, nella prima riga del registro delle versioni, viene invece presentata la versione del progetto in cui è avvenuta l'ultima modifica al documento.
  L'incremento della versione del progetto a seguito di elaborazioni nella documentazione viene gestito attraverso la seguente norma: l'aggiunta di documentazione non precedentemente presente all'interno del repository o la modifica della documentazione precedentemente presente all'interno del repository porta all'incremento del solo valore "(build)".

6.2.3 Produzione

La produzione di ogni documento redatto da SWEetCode presenta le seguenti fasi:

1. **Assegnazione**: Il documento viene assegnato ad uno o più responsabili di stesura affiancati a loro volta da uno o più revisori. Una volta presa la decisione per i precedenti ruoli viene aperto su *Jira* un task a loro associato, relativo alla realizzazione del documento, il quale viene immediatamente inserita all'interno di una epic e ad una versione.

2. **Stesura**: La stesura viene realizzata dal segretario di riunione (nel caso in cui il documento in questione sia un verbale interno o esterno) o dai responsabili di stesura. La produzione della documentazione in formato *Latex* viene gestita tramite la condivisione dei sorgenti attraverso *GitHub*. Una volta che la stesura viene ritenuta stabile, tramite l'automazione descritta in (Norme di progetto, automazione versionamento documenti), viene creata una *pull request* nel repository *Knowledge_Management_AI* su *GitHub*;
3. **Verifica e Validazione**: La fase di verifica viene realizzata dal revisore del documento: in questo stadio il documento viene sottoposto ad un controllo di contenuto e sintassi, tenendo conto delle linee guida definite dal team nelle (Norme di progetto, §Documenti). Se il documento è di tipo esterno, una volta passata la verifica da parte del revisore, l'atto viene condiviso con l'ente terzo in causa, per essere sottoposto ad una sua ulteriore verifica e validazione. In caso di esito positivo la *pull request* viene risolta mentre, in caso di esito negativo, essa viene rifiutata e si ritorna alla fase precedente, in modo da produrre una documentazione che tiene conto delle correzioni e precisazioni presentate dal revisore e/o dall'eventuale parte esterna;
4. **Pubblicazione**: La pubblicazione costituisce l'ultima fase del ciclo di vita del documento e avviene solamente nel caso in cui la fase di verifica abbia avuto un riscontro positivo: una volta risolta la *pull request* il documento troverà infatti collocazione all'interno del repository *Knowledge_Management_AI* del team su *GitHub*. In questo modo il repository contiene solamente documenti verificati ed eventualmente validati.

6.2.4 Manutenzione

Il processo di manutenzione della documentazione è fondamentale per garantire che la stessa rimanga accurata, aggiornata ed utile nel corso del ciclo di vita del progetto. Tale processo è integrato in modo continuo in base ai progressi svolti. Il team segue le seguenti fasi per il processo di manutenzione:

- **Identificazione della necessità della modifica**: Monitoraggio continuo delle esigenze di modifica della documentazione e comunicazione al team della necessità di una possibile modifica;
- **Valutazione dell'impatto**: Discussione tra i membri che ricoprono i ruoli interessati alla modifica per valutarne l'impatto sulla documentazione già esistente. Tale discussione coinvolge tutto il team se necssario;
- **Aggiornamento della documentazione**: Modifica e aggiornamento del documento interessato secondo le esigenze identificate, e verifica dell'accuratezza delle informazioni e dell'allineamento con le modifiche del progetto relative;
- **Push della modifica**: Una volta che la stesura della modifica viene ritenuta stabile, tramite l'automazione descritta in (Norme di progetto, §automazione versionamento documenti), viene creata una pull request nel repository *Knowledge_Management_AI* su *GitHub* per la nuova versione del documento da parte di chi ha operato la modifica;
- **Verifica e validazione**: A seguito della creazione della pull request, le modifiche apportate ai documenti sono sottoposte a verifica eseguita da parte del veri-

ficatore. Se tale operazione ha esito positivo, se necessario, il documento viene posto ad ulteriore validazione con l'ente esterno in questione. Se sia la verifica che la eventuale validazione hanno avuto esito positivo, le modifiche vengono pubblicate attraverso l'accettazione della pull request da parte del revisore, in caso invece di esito negativo in anche una delle due operazioni, la pull request viene rifiutata e colui che aveva realizzato le modifiche è responsabile di applicare le correzioni emerse, ripetendo i passi precedentemente descritti.

6.3 Verifica

Ogni produzione realizzata da SWEetCode viene sottoposta ad una fase di verifica che permette la sua validazione. Onde evitare conflitti di interesse, tale compito viene assegnato ad uno o più membri differenti da coloro che hanno conseguito alla realizzazione del prodotto in questione.

6.3.1 Verifica dei documenti

Ogni documento prodotto dal team passa per una o più fasi di verifica prima di essere pubblicato ufficialmente nel repository *GitHub*.
Tale fasi vengono realizzate solamente in seguito alla consegna del documento da parte dei responsabili di stesura, i quali ritengono che in quel momento il prodotto da loro realizzato sia completato.
La verifica dei documenti prevede che venga assicurato:

- Il rispetto delle norme tipografiche stabilite nelle *Norme di progetto* del team;
- Il rispetto della struttura definita nelle *Norme di progetto* del team;
- Il rispetto del contenuto, il quale deve essere chiaro, coerente, preciso ed esaustivo;
- Il rispetto della sintassi italiana e della correttezza ortografica dei termini utilizzati.

7 Processi organizzativi

7.1 Comunicazione

SWEetCode ha deciso di mantenere una comunicazione frequente tra i membri del gruppo. In particolare, di seguito si trova la modalità di svolgimento per le comunicazioni interne e le applicazioni utilizzate.

7.1.1 Comunicazioni interne

7.1.1.1 Comunicazioni sincrone Il gruppo ha deciso che gli incontri si svolgeranno in presenza oppure attraverso la piattaforma *Discord*. Per gli incontri in presenza, essi verranno decisi alcuni giorni prima in modo tale da poter consentire la presenza di tutti i membri del gruppo. Per quanto riguarda *Discord*, invece, si opta per una comunicazione asincrona anche pochi minuti prima del collegamento. In entrambe le situazioni, il gruppo utilizza un approccio libero alla discussione e improntato alla crescita.

7.1.1.2 Comunicazioni asincrone Il gruppo utilizzerà *Whatsapp* e *Notion* per le comunicazioni asincrone. In particolare è stata creata una community su *Whatsapp*, composta da un gruppo principale per comunicazioni informali, una bacheca nella quale verranno inviati i messaggi più importanti dei quali è necessario prendere visione, ed altri canali in cui i componenti del gruppo si organizzano per task collaborativi.

7.1.2 Comunicazioni esterne

7.1.2.1 Comunicazioni sincrone Assieme ai referenti della azienda AzzurroDigitale il gruppo ha concordato ed accettato un calendario di incontri settimanali su piattaforma *Google Meet*. Tali incontri saranno cruciali nella possibilità di discutere con maggiore facilità e immediatezza argomenti complessi ed estesi. Il gruppo si impegna a presenziare in maniera più assidua possibile agli incontri con il proponente e a redarre un verbale alla fine degli stessi.

7.1.2.2 Comunicazioni asincrone Il canale scelto per la comunicazione asincrona concordato con l'azienda è *Slack*. Questo strumento viene usato per brevi comunicazioni e per accordarsi sugli orari degli incontri.

8 Strumenti

Nella seguente sezione vengono presentati tutti gli strumenti utili al team per lo svolgimento di ogni attività.

8.1 Adobe Illustrator

https://www.adobe.com

Software utilizzato per la creazione dei template ufficiali del team, per la generazione del logo e per la scelta della palette di colori da usare.

8.2 Canva

https://www.canva.com

Sito utilizzato per la produzione delle slide per i Diari di Bordo. É stato creato un account condiviso utilizzabile da tutti i membri.

8.3 Discord

https://discord.com

Piattaforma utilizzata per lo svolgimento di riunioni formali ed informali del team, attraverso la creazione di un server apposito con un canale dedicato ai messaggi di testo e diverse sale dedicate alle conversazioni vocali.

8.4 GitHub

https://github.com/ Piattaforma di hosting e collaborazione per lo sviluppo di software che offre strumenti per il controllo di versione, consentendo agli sviluppatori di lavorare insieme, monitorare le modifiche al codice, gestire problemi e pubblicare il proprio lavoro in modo collaborativo.

8.5 Jira (di *Atlassian*)

https://www.atlassian.com/it/software/jira

Software utilizzato per l'ITS e gli strumenti di organizzazione del lavoro tra i membri del team.

8.6 Diagrams.net

https://app.diagrams.net/

Applicazione utilizzata per la creazione dei diagrammi UML.

8.7 Notion

https://www.notion.so/product

Spazio di lavoro utilizzato per l'organizzazione delle prime task divise per membro del gruppo e per tenere traccia delle attività e dei progressi portati avanti da ogni componente. Usato anche per l'assegnazione chiara dei ruoli, come calendario condiviso del gruppo e per il salvataggio link utili in modo da favorire l'accesso in maniera veloce a tutti i documenti ed i siti necessari.

8.8 Overleaf

https://www.overleaf.com

Sito utilizzato per la scrittura di tutta la documentazione ufficiale in linguaggio LaTeX, attraverso la creazione di un account condiviso per tutti i membri del team.

8.9 Slack

https://slack.com

Piattaforma di messaggistica specializzato nella comunicazione e collaborazione all'interno di team aziendali. Viene utilizzata dal team per le comunicazioni asincrone con il proponente.