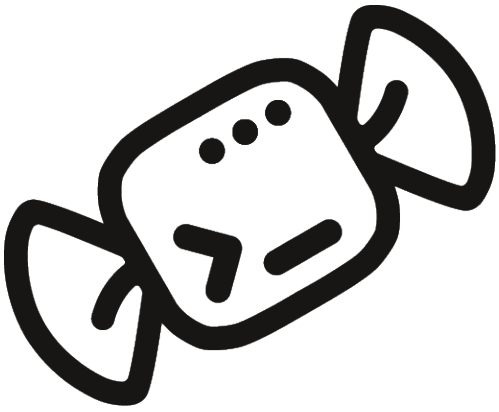

SWEetCode

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
v1.9.0(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.
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.

Continua nella pagina successiva

Versione	Data	Responsabile di stesura	Revisore	Dettaglio e motivazioni
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.
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.

1.2 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.

2 Processi primari

2.1 Sviluppo

Il processo di sviluppo si pone lo scopo di descrivere tutte le azioni che portano un prodotto dalla fase di ideazione a quella di realizzazione. Per tale motivo include le seguenti azioni:

- Determinazione di ogni genere di requisito del prodotto, con eventuali vincoli;
- Progettazione del prodotto seguendo i vincoli ed i requisiti analizzati;
- Testing del prodotto secondo le direttive del committente;
- Realizzazione del prodotto.

2.1.1 Analisi dei requisiti

Il documento 'Analisi dei requisiti' è stilato secondo le seguenti fasi:

- Studio accurato del Capitolato e delle esigenze del committente;
- Ideazione dei Casi d'uso principali ed individuazione dei requisiti obbligatori per il proponente;
- Discussione con il proponente per eventuali chiarimenti;
- Divisione dei requisiti nelle tre categorie individuate con conseguente ampliamento dei Casi d'uso;

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.

Casi d'uso: Notazione
I Casi d'Uso sono indicati con la notazione seguente:
UC[X].[Y] in cui:

- UC sta per *Use Case*;
- .[X] indica il numero del Caso d'Uso, presentati secondo successione gerarchica;
- .[Y] indica il possibile numero del sotto-Caso d'uso, che aggiunge informazioni al Caso d'Uso [X]. Anche i sotto-Casi d'Uso sono presentati secondo successione gerarchica.

Requisiti: Notazione
Ogni requisito preso in analisi è trattato con la sigla:
R[Tipo].[Importanza].[Ordine];
nella quale:

- R indica la parola "requisito";
- Tipo può essere:
  - F (Funzionale);
  - Q (Qualità);

    - V (Vincolo).
- Importanza può essere:
    - O (Obbligatorio);
    - D (Desiderabile);
    - OP (Opzionale).
- Ordine: indica l'ordine in cui il requisito è stato preso in esame. Per i requisiti con stessa importanza e tipologia indica l'ordine cronologico di precedenza che esso ha avuto rispetto agli altri, ovvero il suo ordine inserimento nell'Analisi dei requisiti.

N.B: per requisiti che dovessero generare requisiti figli, si andrà ad aggiungere .[N.figlio] alla notazione successivamente all'ordine, di modo da avere i figli ordinati per importanza.

Requisiti: Suddivisione (Funzionali, di Qualità, di Vincolo)

- I requisiti funzionali descrivono le funzionalità del sistema, le azioni che il sistema può compiere e le informazioni che il sistema può fornire. Si intende che la numerologia di ciascuno rispecchi ove previsto gli UC presenti, mentre le sigle sotto riportate indicano:
    - ROF - Requisito Obbligatorio Funzionale;
    - RDF - Requisito Desiderabile Funzionale;
    - ROPF - Requisito Opzionale Funzionale.
- I requisiti di qualità descrivono come un sistema deve essere, o come il sistema deve esibirsi, per soddisfare le esigenze dell'utente. Le sigle sotto riportate possono essere così classificate:
    - ROQ - Requisito Obbligatorio di Qualità;
    - RDQ - Requisito Desiderabile di Qualità;
    - ROPQ - Requisito Opzionale di Qualità.
- I requisiti di vincolo descrivono i limiti e le restrizioni normative/legislative che un sistema deve rispettare per soddisfare le esigenze dell'utente. Le sigle sotto riportate possono essere così classificate:
    - ROV - Requisito Obbligatorio di Vincolo;
    - RDV - Requisito Desiderabile di Vincolo;
    - ROPV - Requisito Opzionale di Vincolo.

3 Processi di supporto

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

3.1 Configuration management

In questa sezione vengono presentate tutte le attività che coinvolgono la gestione delle configurazioni del software realizzate da SWEetCode.

3.1.1 Issue tracking system (ITS)

SWEetCode utilizza l'*Issue Tracking System Jira* fornito da *Atlassian*.

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 crezione 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 collerazione: ogni *ticket* può essere associata ad una determinata *versione*.

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.

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.

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 inzio ed una di fine e da uno stato (*Attivato*/*Chiuso*).

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.

Automazione chiusura ticket *Jira* mette a disposizione una funzionalità di creazione di automazioni custumizzabili dall'utente. 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*.

3.1.2 Pull requests

A seguito dell'adozione dei *GitHub Teams*, SWEetCode propone l'utilizzo delle pull requests per permettere un rilascio controllato: 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*.

3.1.3 Version control

Il sistema di *Version control* adottato dal team prevede una versione nel 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".

3.1.4 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 pr 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 repo remota e la correttezza del suo contenuto.

3.2 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.

3.2.1 Ciclo di vita

Ogni documento prodotto da SWEetCode presenta le seguenti fasi:

1. Assegnazione:
   Ogni 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 aperta su *GitHub* una issue a loro associata relativa alla realizzazione del documento, la quale viene immediatamente inserita all'interno della Board del progetto. Ogni issue viene inoltre etichettata da una label relativa al tipo di documento da produrre;
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 piattaforma *Overleaf*, la quale rende disponibile gli strumenti adatti ad una eventuale stesura sincronizzata e collaborativa tra i membri del team. Una volta che la stesura viene ritenuta stabile, tramite l'automazione descritta in (Norme di progetto, automazione documenti), viene poste le basi per 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). Se il documento è di tipo esterno, una volta eseguita 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 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.

3.2.2 Automazione versionamento documenti

L'aggiornamento del registro delle versioni, la creazione dei pdf e la pubblicazione delle modifiche nella repo 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 (Norme di progetto, §Ciclo di vita).

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.

3.2.3 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.

3.2.4 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.

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.

Verbali Fanno eccezione a questa struttura i verbali, sia esterni che interni, che oltre a presentare nella prima pagina di copertina i ruoli inerenti alla sua produzione al posto dell'elenco dei membri del team e 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), offre una struttura particolare.
Il contenuto presenta infatti 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).

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"*.

3.2.5 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 che non presentano alcuna correlazione con l'aspetto temporale;
- cronologico: sono elenchi numerati e descrivono il seguirsi di avvenimenti o argomenti 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 vengono riportate seguendo la notazione (Nome documento, §Nome sezione). Questi riferimenti saranno opportunamente collegati tramite link al paragrafo indicato.

Riferimenti esterni Menzioni e riferimenti a sezioni di documenti esterni vengono riportate seguento la notazione (Nome documento, §Numero 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)".

3.3 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.

3.3.1 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.

4 Processi organizzativi

4.1 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.

4.1.1 Comunicazioni interne

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.

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.

4.1.2 Comunicazioni esterne

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.

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.

4.2 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.

5 Strumenti

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

5.1 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.

5.2 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.

5.3 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.

5.4 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.

5.5 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.

5.6 Lucidchart

https://www.lucidchart.com

Applicazione utilizzata per la creazione dei diagrammi dei Casi d'Uso presenti nell'Analisi dei Requisiti.

5.7 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.

5.8 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.

5.9 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.