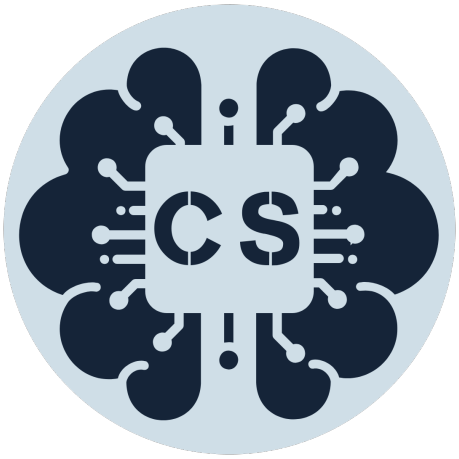
Norme di progetto

CyberSorcerers Team



Informazioni sul documento		
Destinatari:	Prof Tullio Vardanega	Prof Riccardo Cardin
G al pedice:	Consultare il Glossario	

Membri del team:
Sabrina Caniato
Giulia Dentone
Nicola Lazzarin
Giovanni Moretti
Andrea Rezzi
Samuele Vignotto

Registro dei Cambiamenti - Changelog$_G$

Versione	Data	Autore	Verificatore	Dettaglio
2.0.0	28/05/2024	Giulia Dentone	Sabrina Caniato	Verifica e correzione finale del documento.
1.1.1	26/05/2024	Sabrina Caniato	Giulia Dentone	Update della sezione di Gestione Organizzativa-Sprint.
1.1.0	06/05/2024	Giulia Dentone	Samuele Vignotto	Update della sezione di Gestione Organizzativa.
1.0.0	30/12/2023	Sabrina Caniato	Andrea Rezzi	Update della sezione di codifica.
0.7.5	30/12/2023	Giovanni Moretti	Giulia Dentone	Update della sezione di codifica.
0.7.4	30/12/2023	Samuele Vignotto	Sabrina Caniato	Update della sezione di validazione.
0.7.3	29/12/2023	Nicola Lazzarin	Andrea Rezzi	Update della sezione di verifica.
0.7.2	29/12/2023	Andrea Rezzi	Sabrina Caniato	Update della sezione documenti.
0.7.1	28/12/2023	Giulia Dentone	Giovanni Moretti	Update della sezione dei documenti con i verbali.
0.7.0	27/12/2023	Samuele Vignotto	Sabrina Caniato	Compilazione della sezione documenti.
0.6.2	27/12/2023	Nicola Lazzarin	Sabrina Caniato	Update dei processi di supporto, gestione della configurazione e della repository.
0.6.1	26/12/2023	Andrea Rezzi	Nicola Lazzarin	Update dei processi di supporto, validazione.
0.6.0	20/12/2023	Sabrina Caniato	Nicola Lazzarin	Aggiunta dei processi di supporto, verifica.
0.5.1	12/12/2023	Giovanni Moretti	Andrea Rezzi	Update dei processi organizzativi.

0.5.0	31/11/2023	Giovanni Moretti	Andrea Rezzi	Aggiunta dei processi organizzativi.
0.4.2	25/11/2023	Samuele Vignotto	Nicola Lazzarin	Aggiunta delle sezione delle norme.
0.4.1	17/11/2023	Giulia Dentone	Sabrina Caniato	Udate della sezione delle attività.
0.4.0	17/11/2023	Giulia Dentone	Sabrina Caniato	Aggiunta sezione delle attività.
0.3.1	17/11/2023	Giovanni Moretti	Andrea Rezzi	Update sezione dei processi di fornitura.
0.3.0	16/11/2023	Giovanni Moretti	Andrea Rezzi	Aggiunta sezione delle attivtà.
0.2.0	14/11/2023	Nicola Lazzarin	Giovanni Moretti	Aggiunta dei processi di sviluppo.
0.1.0	14/11/2023	Giovanni Moretti	Andrea Rezzi	Aggiunta del processo di fornitura.
0.0.1	13/11/2023	Andrea Rezzi	Giovanni Moretti	Definizione struttura del documento e scheletro delle sezioni. Scrittura introduzione ed obiettivi delle diverse sezioni.

Indice

1 Introduzione

1.1 Scopo del documento

Lo scopo di questo documento, in continuo aggiornamento, è quello di raggruppare in un unico luogo tutte le decisioni prese dal gruppo, per quanto riguarda il proprio *Way of Working*$_G$.

1.2 Scopo del capitolato

Lo scopo è creare un middleware$_G$ che riceva in input dei requisiti di business$_G$ e produca epic e user stories$_G$ associate ai requisiti di business$_G$ tramite ChatGPT$_G$ e AWS BedRock$_G$, inoltre è richiesto che venga creato un plug-in$_G$ per VisualStudio Code. Sarà necessario comparare la capacità di ChatGPT$_G$ e quella di AWS BedRock$_G$ nell'interpretare del codice sorgente ed associare le user stories$_G$ generate. Provare dall'interpretazione dei criteri di accettazione delle user stories$_G$ ed il codice analizzato se il risultano dei test$_G$ non gestiti.

1.3 Glossario

I termini impiegati in questo testo potrebbero suscitare incertezze circa il loro significato, rendendo quindi necessaria una definizione per evitare ambiguità. Tali termini sono identificati da una lettera "G" maiuscola posta in pedice alla parola, e la loro spiegazione è fornita nel Glossario v1.0.0.

1.4 Riferimenti

1.4.1 Riferimenti normativi

- **Regolamento del progetto didattico:**

PD2.pdf

- **Capitolato d'appalto C7 - ChatGPT$_G$ vs BedRock$_G$ developer Analysis**

C7.pdf

1.4.2 Riferimenti informativi

- **Documentazione Amazon BedRock$_G$**

Documentazione BedRock$_G$

- **Documentazione OpenAI ChatGPT$_G$**

Documentazione ChatGPT$_G$

- **Standard ISO/IEC 12207**

ISO/IEC 12207-1995

- **Standard ISO/IEC 9126**

ISO/IEC 9126

- Lezioni del corso di Ingegneria del Software "I processi di ciclo di vita del SW":

  `https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/T2.pdf`

- Lezioni del corso di Ingegneria del Software "Progettazione Software":

  `https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/T6.pdf`

- Lezioni del corso di Ingegneria del Software "Verifica e Validazione: introduzione":

  `https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/T9.pdf`

- Lezioni del corso di Ingegneria del Software "Amministrazione di progetto":

  `https://www.math.unipd.it/~tullio/IS-1/2023/Dispense/PD5.pdf`

- Lezioni del corso di Ingegneria del Software "Verifica e Validazione: analisi statica":

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

- Lezioni del corso di Ingegneria del Software "Verifica e Validazione: analisi dinamica aka testing":

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

2 Processi primari

2.1 Fornitura

2.1.1 Scopo

In questa sezione sono indicati tutti i parametri, gli strumenti e i documenti impiegati per portare a termine il processo di fornitura.

2.1.2 Aspettative

Le aspettative relative all'implementazione del processo di fornitura comprendono:

- Ottenere una struttura documentale chiara;
- Definire i tempi di lavoro;
- Risolvere eventuali dubbi con il proponente;
- Stabilire vincoli con il proponente;

2.1.3 Valutazione dei capitolati

Per poter determinare la corretta valutazione dei capitolati proposti, il gruppo guidato dal Responsabile ha organizzato incontri interni. In questa prima fase, i vari membri hanno alternato il ruolo di Analisti, prendendo in esame per ogni capitolato:

- informazioni generali e dominio applicativo/tecnologico;
- valutazione oggettiva (pro e contro), completa disamina dei punti di forza e fattori critici;
- interessi comuni ai membri del gruppo per quanto riguarda la tipologia di progetto da svolgere;
- relative conclusioni in merito alle scelte operate.

Tale analisi può trovare riscontro nel relativo documento Valutazione dei capitolati a cui si è giunti per la scelta definitiva.

2.1.4 Descrizione

Il processo di fornitura identifica ogni compito, attività e risorsa necessaria per l'implementazione del progetto. Questo processo sarà avviato solo dopo la comprensione delle richieste del proponente, seguito da uno studio di fattibilità delle richieste e concluso con la definizione di un accordo contrattuale. Le fasi del processo di fornitura includono:

- Avvio;
- Contrattazione;
- Pianificazione;
- Esecuzione;
- Controllo;
- Revisione;
- Valutazione;
- Consegna;
- Completamento;

2.1.5 Proponente

Il team di sviluppo ha concordato con il proponente di mantenere un contratto regolare per monitorare l'andamento positivo del progetto. Questo viene realizzato attraverso incontri periodici organizzati e un costante scambio asincrono tramite un canale dedicato sulla piattaforma Slack$_G$. Il team di sviluppo mira a mantenere un contatto costante con il proponente per discutere i seguenti argomenti:

- Vincoli e requisiti obbligatori.
- Feedback riguardante le tecnologie utilizzate;
- Valutazione delle soluzioni innovative proposte dal team di sviluppo;
- Chiarimenti su eventuali dubbi;
- Feedback riguardante la documentazione redatta;
- Stima dei costi;

2.1.6 Collaudo e rilascio

La fase di collaudo permette al team di verificare che i requisiti stabiliti durante l'analisi siano rispettati e che non ci siano errori. Se il collaudatore rileva anomalie, queste devono essere risolte. Se invece il prodotto risulta privo di errori, il Responsabile di Progetto presenterà al cliente il consuntivo finale e consegnerà il codice sorgente e la documentazione relativa al progetto. Infine, una volta che il prodotto è stato rilasciato, non è prevista alcuna fase di manutenzione.

Strumenti

Di seguito vengono indicati gli strumenti utilizzati dal team per la realizzazione del processo di fornitura:

- *Microsoft Excel*: utilizzato per creare grafici, eseguire calcoli e per realizzare tabelle;
- *Project di Github*: utilizzato per gestire le task$_G$.
- *draw.io*: utilizzato per la realizzazione dei diagrammi UML$_G$.
- *GanttProject*: utilizzato per la realizzazione dei diagrammi di Gantt$_G$.
- *LucidChart*: utilizzato per disegnare diagrammi descrittivi del funzionamento e delle architetture del progetto.

2.2 Sviluppo

2.3 Scopo

L'obiettivo del processo di sviluppo è delineare i compiti e le attività necessarie per lo sviluppo del prodotto software. In questa sezione, vengono dettagliate le attività, le norme e le convenzioni adottate per la composizione di questo processo.

2.4 Aspettative

Le aspettative nell'applicazione del processo di sviluppo includono:

- Determinare vincoli tecnologici.
- Stabilire gli obiettivi di sviluppo.
- Definire vincoli di design$_G$.
- Produrre un prodotto finale che superi i test$_G$ e soddisfi i requisiti e le richieste del proponente.

2.5 Descrizione

Il processo di sviluppo comprende le attività e i compiti dello sviluppatore, tra cui l'analisi dei requisiti, la progettazione, la codifica, l'integrazione, il test, l'installazione e l'accettazione relativi ai prodotti software.

2.6 Attività

Qui di seguito sono elencate e successivamente approfondite le attività caratterizzanti tale processo:

- Analisi dei requisiti.
- Progettazione.
- Codifica.

2.6.1 Analisi dei requisiti

Scopo

L'obiettivo dell'analisi dei requisiti è individuare, attraverso uno studio approfondito del capitolato, i requisiti diretti e indiretti, impliciti ed espliciti richiesti dal proponente per la realizzazione del prodotto, nonché i vari casi d'uso$_G$ del prodotto stesso. In questa attività, è fondamentale suddividere il problema iniziale in requisiti il più elementari possibile, al fine di ridurre la complessità del problema originale e facilitare il lavoro durante la fase di sviluppo.

Aspettative

L'obiettivo dell'attività di analisi dei requisiti è la creazione di documentazione formale che contenga tutti i requisiti richiesti dal proponente, sia che siano stati definiti all'inizio del processo, sia che siano stati concordati in fase di sviluppo.

Requisiti

I requisiti sono raccolti da diverse fonti, tra cui:

- Lettura investigativa del capitolato.
- Confronto interno tra i membri del gruppo.
- Dialogo con il proponente.
- Analisi dei casi d'uso$_G$.

Classificazione dei requisiti

Ogni requisito è classificato attraverso l'utilizzo di un codice identificativo. Il gruppo ha definito uno standard per questo codice, che sarà spiegato di seguito.
Codice identificativo:

R[Rilevanza][Tipologia][Codice]

dove:
Rilevanza: indica la rilevanza associata ad un requisito e può assumere una tra tre possibili lettere con il seguente significato:

Lettera	Descrizione
O	Requisito obbligatorio
D	Requisito desiderabile ma non obbligatorio
P	Requisito opzionale

Tabella 1: Tabella della classificazione della rilevanza dei requisiti

Tipologia: indica una tipologia di requisito e può assumere una delle seguenti lettere con il seguente significato:

Lettera	Descrizione
V	Vincolo
P	Prestazionale
Q	Qualitativo
F	Funzionale

Tabella 2: Tabella della classificazione della tipologia dei requisiti

Codice: L'identificativo è unico e espresso in forma gerarchica padre/figlio. Una volta associato a un requisito, l'identificativo è immutabile. Ogni requisito deve essere accompagnato da una serie di informazioni aggiuntive.

- **Descrizione:** Descrizione sintetica ed esplicativa del requisito;
- **Classificazione:** Per migliorare la leggibilità della tabella, viene enfatizzata nuovamente l'importanza del requisito, anche se già presente nell'identificativo del codice.
- **Indirizzato a:** Indica gli attori$_G$ a cui è indirizzato il requisito.
- **Casi d'uso collegati:** Indica i codici di eventuali casi d'uso$_G$ collegati al requisito.
- **Fonte:** Indica l'origine del requisito.

Casi d'uso

I casi d'uso$_G$ rappresentano un comportamento o un modo di utilizzare il prodotto e vengono descritti graficamente attraverso l'uso di diagrammi UML$_G$. Ciascun caso d'uso$_G$ è costituito da:

- Codice identificativo;
- Attore primario;
- Precondizioni;
- Postcondizioni;
- Scenario principale;
- Generalizzazioni;
- Estensioni;

Classificazione dei casi d'uso

Ogni caso d'uso$_G$ è classificato attraverso l'utilizzo di un codice identificativo. Il gruppo ha definito uno standard per questo codice, che sarà spiegato di seguito.
Codice Identificativo

UC[Numero caso d'uso](.[Numero caso d'uso figlio])*-[Titolo caso d'uso]

dove:

- **UC:** Acronimo di "Use Case";
- **Numero caso d'uso:** Numero associato al caso d'uso$_G$ principale;
- **Caso d'uso figlio:** Numero associato al sottocaso del caso d'uso principale;
- **Titolo caso d'uso:** Titolo assegnato al caso d'uso.

Una volta associato a un caso d'uso$_G$, l'identificativo è immutabile.

UML

I diagrammi UML$_G$ vengono realizzati utilizzando la versione 2.0 del linguaggio.

2.6.2 Progettazione

Progettazione

Scopo

L'obiettivo dell'attività di progettazione è integrare le parti ottenute durante l'*analisi dei requisiti*, specificando le funzionalità dei sottosistemi e convergendo verso una soluzione unificata.

Aspettative

L'obiettivo dell'attività di progettazione è realizzare l'architettura del sistema. Inizialmente, questa architettura è creata attraverso un Proof of Concept$_G$, sviluppato come una demo prototipale del sistema durante la fase di Requirements & Technology Baseline$_G$. Successivamente, questa architettura viene approfondita e descritta nel documento tecnico allegato alla Product Baseline$_G$. L'attività di progettazione può essere sufdivisa in tre livelli:

- Design dell'interfaccia: Questa fase si concentra su un livello elevato di astrazione rispetto al funzionamento interno del sistema. Durante la progettazione dell'interfaccia, l'attenzione è rivolta alle tecnologie che verranno impiegate nella fase di sviluppo del software. Questo livello specifico porta alla creazione di un Proof of Concept;
- Progettazione architettonica: In questa fase viene determinata la struttura generale del sistema, con una definizione ad alto livello che ignora i dettagli interni dei principali componenti del prodotto;
- Progettazione dettagliata: Questa fase specifica gli elementi interni di tutti i principali componenti e le specifiche architetturali del prodotto. Include anche la definizione dei diagrammi delle classi e dei test di unità per ciascun componente. Questa fase della progettazione rappresenta la Product Baseline.

Requirements & Technology Baseline$_G$

Questa fase stabilisce i requisiti che il fornitore si impegna a soddisfare, in accordo con il proponente, e motiva le tecnologie, i framework$_G$ e le librerie selezionate per la realizzazione del prodotto. Dimostra l'adeguatezza e la fattibilità coerente con gli obiettivi. I documenti utili a tali fini sono:

- *Piano di progetto*;
- *Piano di qualifica*;
- *Norme way of working*$_G$;
- *Verbali* interni ed esterni;

Ad eccezione dei *verbali*, i documenti sopracitati devono essere conservati e aggiornati anche nelle fasi successive.

Product Baseline$_G$

Questa fase presenta le linee guida architetturali del prodotto. Oltre all'aggiornamento dei documenti menzionati in precedenza, in questa fase saranno necessari:

- *Manuale utente*;
- *Specifica Tecnica*;
- *Verbali di periodo*.

Analogamente alla fase precedente, anche i documenti di questa fase devono essere conservati e aggiornati fino alla consegna e accettazione del prodotto finale.

2.7 Codifica

2.7.1 Descrizione, scopo e aspettative

Dopo la fase di progettazione del prodotto, i membri del team con il ruolo di programmatori passeranno alla fase di codifica, implementando le specifiche dei requisiti e i documenti di progettazione. L'obiettivo di questa fase è concretizzare il progetto, realizzando il software desiderato attraverso la programmazione. Le aspettative per la fase di codifica sono:

- completare lo sviluppo del prodotto finale garantendone la qualità e rispettando le richieste del proponente;
- assicurare che il codice prodotto sia facilmente leggibile.

Strumenti per la codifica

- **TypeScript**$_G$: è un linguaggio di programmazione che estende JavaScript aggiungendo il supporto per tipi statici. Questo permette di rilevare errori e migliorare la qualità del codice durante lo sviluppo. TypeScript è compatibile con tutti i browser e i framework JavaScript, facilitando la scrittura di codice più robusto e manutenibile.

  https://www.typescriptlang.org/

- **Node.js**$_G$: è un runtime system$_G$ open source$_G$ multipiattaforma orientato agli eventi per l'esecuzione di codice JavaScript$_G$;

  nodejs.org/it/

- **React**: è una libreria open source$_G$, frontend$_G$, JavaScript$_G$ per la creazione di interfacce utente;

  it.reactjs.org/

- **MongoDB Atlas**: è il servizio di database cloud completamente gestito offerto da MongoDB, Inc. È progettato per semplificare la gestione, l'implementazione e la scalabilità dei database MongoDB su AWS$_G$, Azure e Google Cloud Platform;

  www.mongodb.com/atlas/database

- **Amazon AWS**$_G$: è una piattaforma di servizi cloud che offre una vasta gamma di servizi informatici affidabili, scalabili ed economici. Questi servizi consentono alle imprese di eseguire applicazioni e gestire risorse IT senza dover investire in infrastrutture hardware costose e complesse;

  aws.amazon.com/it/

  - **AWS API Gateway**$_G$: è un servizio gestito che semplifica la creazione, la pubblicazione, la manutenzione, il monitoraggio e la sicurezza di API$_G$ per le applicazioni su AWS$_G$. Consente agli sviluppatori di creare API RESTful o WebSocket che consentono alle applicazioni di comunicare tra loro o con servizi back-end;

    aws.amazon.com/it/api-gateway/

- **AWS Lambda$_G$**: è un servizio di calcolo serverless che consente agli sviluppatori di eseguire codice senza dover gestire l'infrastruttura sottostante. Con Lambda$_G$, è possibile eseguire codice in risposta a eventi generati da altri servizi AWS$_G$ o da applicazioni personalizzate;

  aws.amazon.com/it/lambda/

- **AWS Cognito$_G$**: è un servizio di gestione dell'identità e dell'accesso (IAM) fornito da Amazon Web Services (AWS)$_G$. È progettato per semplificare l'aggiunta di funzionalità di autenticazione, autorizzazione e gestione degli utenti alle applicazioni web e mobile.

  aws.amazon.com/it/cognito/

3 Norme

3.1 Gestione dell'ITS e delle task

Per il progetto verrà utilizzato l'*issue tracking system$_G$* di Github$_G$. Le task$_G$ vengono decise e assegnate durante le riunioni interne. Oltre al responsabile della task$_G$, al quale viene assegnata tramite l'apposita *feature$_G$*, viene segnato nella prima riga della descrizione il verificatore.
Quando la task$_G$ è completata, viene creata una *pull-request*, che andrà approvata dal verificatore. Lo stato degli *issues$_G$* è tracciato in un *project* di Github$_G$.

3.2 File di documentazione

Tutti i file di documentazione vengono posti nella *repository$_G$* in formato PDF. I *template$_G$* dei file vengono invece posti nell'apposita cartella in formato tex. Inoltre in ogni documento sarà presente un Changelog$_G$ in modo tale da tracciare la stesura della documentazione.

3.3 Numero di versione

Ogni file ha un numero di versione$_G$ posto in coda al proprio nome. Il numero di versione$_G$ è di tipo x.y.z. Il cambiamento di ogni cifra è determinato dai seguenti criteri:

- **Cambiamento di x** → indica una modifica sostanziale
- **Cambiamento di y** → indica l'aggiunta di una nuova feature$_G$
- **Cambiamento di z** → indica una modifica minore

3.4 Politica di decisioni

Le decisioni ufficiali devono essere prese quando tutti i membri del gruppo sono presenti in modo tale da mantenere coerenza. In caso di impossibilità di presenza di tutti i membri del gruppo, deve essere comunque raggiunta una quota di maggioranza.

3.5 Incontri successivi

Ad ogni incontro viene fissata la data del successivo.

4 Processi organizzativi

4.1 Formazione

Il processo di formazione è di fondamentale importanza per garantire che tutti i membri del team acquisiscano le competenze necessarie per svolgere con successo le attività previste dal progetto. La formazione include sessioni di apprendimento relative agli strumenti utilizzati, alle tecnologie adottate e ai processi implementati. L'obiettivo è assicurare una conoscenza approfondita e uniforme all'interno del team. Di seguito, alcune documentazioni utilizzate per la formazione:

- GitHub$_G$ https://docs.github.com/
- Jira https://confluence.atlassian.com/jira
- Git https://docs.github.com/en/get-started/using-git/about-git
- Amazon AWS$_G$ https://aws.amazon.com/it/getting-started/hands-on/build-web-app-s3-lambda-api-gateway-dynamodb/

4.2 Impiego delle infrastrutture interne

L'utilizzo delle infrastrutture interne,è cruciale per mantenere un ambiente di lavoro organizzato ed efficiente. Il team sfrutta a pieno queste risorse per tracciare lo stato delle attività, gestire le modifiche al codice e facilitare la collaborazione tra i membri. Gli strumenti utilizzati sono i seguenti:

- Github$_G$: issue tracking system$_G$ largamente utilizzato, affidabile e intuitivo
- Telegram: comunicare velocemente tra membri tramite chat
- Google Meet: per videochiamarci e fare riunioni interne ed esterne da remoto, evitando l'ostacolo della distanza tra i membri
- Slack$_G$: comunicare rapidamente col proponente
- Google Calendar: segnare le milestone e gli obiettivi principali, oltre alle riunioni
- Gmail: creare account sulle varie piattaforme, comunicare con i professori del corso a nome del gruppo e in una prima fase anche comunicare con il proponente

4.3 Gestione organizzativa

La gestione organizzativa del progetto coinvolge la pianificazione, la distribuzione delle responsabilità e la gestione degli incontri. Il team adotta un approccio collaborativo, dove ciascun membro ha un ruolo definito e contribuisce al raggiungimento degli obiettivi. Gli incontri periodici sono fondamentali per monitorare l'avanzamento, risolvere eventuali problemi e pianificare le prossime attività. Il team, con l'obiettivo di ottimizzare i tempi e massimizzare l'efficienza delle proprie operazioni, ha scelto di adottare una metodologia Agile$_G$, implementando le pratiche di miglioramento continuo attraverso il framework Scrum$_G$. Questo approccio consente di organizzare il lavoro in intervalli temporali definiti, noti come sprint, della durata di una settimana. Nello specifico, il processo si articola in diverse fasi:

- Pianificazione dello Sprint: Il team si riunisce il primo giorno dello sprint per pianificare le attività da svolgere durante il periodo corrente. Attraverso sessioni di brainstorming, vengono identificate le attività da completare (backlog), stabilendo preventivamente gli impegni e le risorse necessarie.

- Revisione dello Sprint: Al termine dello sprint, si tiene una riunione di revisione alla quale partecipano tutti i membri del team. L'obiettivo è definire gli obiettivi raggiunti e produrre almeno un incremento, ovvero un prodotto software utilizzabile. Durante questa fase, vengono analizzate in modo approfondito le risorse impiegate rispetto agli obiettivi, distinguendo tra quelli raggiunti e non raggiunti. Ciò consente di identificare aree di miglioramento e di stabilire nuovi obiettivi per gli sprint successivi.
- Retrospective dello Sprint: Concluso lo sprint, il team si riunisce per una valutazione generale del suo andamento. Si analizza ciò che è stato realizzato con successo e ciò che può essere migliorato. Questa fase è cruciale per definire le strategie di ripianificazione delle attività, decidendo come procedere con quelle attuali o future.

La durata media di uno sprint nel contesto del framework Agile può variare significativamente in base alle preferenze del team e alle esigenze specifiche del progetto. Tuttavia, di solito uno sprint ha una durata compresa tra una e quattro settimane. Il nostro team ha deciso di adottare sprint settimanali, in modo tale da evitare il più possibile momenti di stallo e favorire il confronto tra i membri.

4.3.1 Ruoli

4.3.2 Metodo organizzativo

Il team, con l'obiettivo di ottimizzare i tempi e massimizzare l'efficienza delle proprie operazioni, ha scelto di adottare una metodologia Agile$_G$, implementando le pratiche di miglioramento continuo attraverso il framework Scrum$_G$. Questo approccio consente di organizzare il lavoro in brevi intervalli temporali, noti come sprint, della durata di circa due settimane. Nello specifico, il processo si articola in diverse fasi:

- *Pianificazione dello Sprint:* Il team si riunisce il primo giorno dello sprint per pianificare le attività da svolgere durante il periodo corrente. Attraverso sessioni di brainstorming, vengono identificate le attività da completare (backlog), stabilendo preventivamente gli impegni e le risorse necessarie.
- *Revisione dello Sprint:* Al termine dello sprint, si tiene una riunione di revisione alla quale partecipano tutti i membri del team. L'obiettivo è definire gli obiettivi raggiunti e produrre almeno un incremento, ovvero un prodotto software utilizzabile. Durante questa fase, vengono analizzate in modo approfondito le risorse impiegate rispetto agli obiettivi, distinguendo tra quelli raggiunti e non raggiunti. Ciò consente di identificare aree di miglioramento e di stabilire nuovi obiettivi per gli sprint successivi.
- *Retrospective dello Sprint:* Concluso lo sprint, il team si riunisce per una valutazione generale del suo andamento. Si analizza ciò che è stato realizzato con successo e ciò che può essere migliorato. Questa fase è cruciale per definire le strategie di ripianificazione delle attività, decidendo come procedere con quelle attuali o future.

4.3.3 Gestione degli incontri

Gli incontri vengono organizzati almeno una volta alla settimana e vengono fatti da remoto grazie a google meet. Ci servono per capire come stiamo procedendo e per evitare la modalità sottomarino, ovvero il rischio che ogniuno si prenda le cose da fare e non avvisi degli avanzamenti delle task. Sempre con questo scopo, ogni due giorni facciamo un daily meeting di cui non rediamo il verbale. Alla fine di ogni incontro settimanale, di cui si è steso il verbale, decidiamo il giorno e l'ora del prossimo meeting. Per farlo abbiamo stabilito due metodi:
Il primo metodo: il responsabile propone un giorno e se tutti sono liberi abbiamo già deciso altrimenti si va avanti a proporre.
Il secondo metodo: utilizziamo un sondaggio fatto al momento, sempre dal responsabile, tramite telegram ed il giorno più votato ci sarà l'incontro.

Appena deciso il giorno il responsabile scrive nel calendario google il girno e l'ora dell'incontro, così facendo ogni membro del gruppo avrà nel calendario un promemoria della riunione.
Se ci si trova internamente al gruppo di non essere in grado a rispondere ai quesiti che sorgono durante una riunione interna, decidiamo di contattare l'azienda tramite slack$_G$. Solitamente noi proponiamo un incontro e loro ci dicono i giorni in cui sono disponibili, tramite un sondaggio decidiamo il giorno migliore per tutti. Se sono incontri formativi, solitamente li facciamo in presenza, se sono incontri informativi vengono fatti da remoto, grazie a google meet.

4.3.4 Gestione della comunicazione interna ed esterna

La comunicazione interna avviene tramite telegram. Viene usato principalmente per chiarire dubbi e per tenerci aggiornati sulle cose che sono state fatte.
Per la comunicazione esterna abbiamo un canale slack$_G$, di cui fanno parte anche i responsabili dell'azienda proponente. Il canale viene utilizzato solo se abbiamo domande veloci e non complicate, ad esempio per chiedere degli incontri. Se ci sono domande più complesse, che hanno bisogno di più tempo per le risposte, solitamente utilizziamo le mail, come suggerito dall'azienda. Il responsabile è l'unico che scrive i messaggi per la comunicazione esterna.

4.3.5 Gestione delle task

Le task$_G$ vengono gestite tramite GitHub$_G$. Il responsabile, alla fine di ogni pianificazione dello Sprint aggiunge nella colonna "to-do" di GitHub$_G$ le attività che si erano concordate. Da la ogni membro del gruppo quando inizia a svolgere un'attività la sposta nella colonna "in progress", e quando viene terminata crea il commit in cui chiude la task$_G$ e automaticamente la sposta in "done".

5 Processi di supporto

5.1 Gestione della qualità

La gestione della qualità è un approccio sistemico che coinvolge la definizione di standard di qualità, la misurazione delle performance$_G$ e l'implementazione di azioni correttive. Le metriche di qualità vengono regolarmente monitorate, e i processi vengono migliorati costantemente per garantire l'aderenza agli standard stabiliti e il raggiungimento degli obiettivi di qualità.

5.1.1 Verifica

Scopo
Il processo di supporto di verifica nel contesto dello sviluppo di un progetto software ha lo scopo di garantire che il software prodotto soddisfi i requisiti specificati e che sia conforme agli standard e alle linee guida stabilite. Questo processo è fondamentale per identificare e correggere eventuali difetti, errori o problemi nel software prima che venga rilasciato agli utenti finali.
Aspettative
Le aspettative del processo di verifica sono principalmente:

- **Identificare difetti**: L'obiettivo principale è individuare difetti, errori o problemi nel software. Ci si aspetta che il processo di verifica identifichi in modo efficace i difetti presenti nel software, che possono includere errori di programmazione, discrepanze rispetto ai requisiti o problemi di prestazioni.
- **Garantire la conformità**: Il processo di verifica mira a garantire che il software sia conforme ai requisiti specificati, agli standard e alle linee guida stabilite per il progetto. Ci si aspetta che il software sia sviluppato in modo da soddisfare tali criteri di conformità.

- **Migliorare la qualità**: Il processo di verifica contribuisce a migliorare la qualità complessiva del software, identificando e correggendo difetti prima del rilascio. Ci si aspetta che il software verificato sia più affidabile, sicuro e performante.
- **Rispettare i vincoli di tempo e budget**: Un'altra aspettativa è che il processo di verifica venga condotto in modo efficiente e in linea con i vincoli di tempo e budget del progetto. Ci si aspetta che le attività di verifica siano pianificate e eseguite in modo da non compromettere il programma di sviluppo complessivo o il bilancio del progetto.
- **Documentazione accurata**: Si richiede che il processo di verifica generi documentazione accurata e completa, risultati dei test$_G$ e altre informazioni pertinenti. Questa documentazione è essenziale per tracciare il progresso del progetto e fornire una chiara visione dello stato del software.

Descrizione
Il processo di verifica è una pratica svolta in maniera continua che coinvolge l'analisi approfondita di ogni componente del progetto. La revisione dei documenti, l'analisi statica e dinamica del codice, insieme a test$_G$ specifici, contribuiscono a garantire la qualità e l'affidabilità del prodotto. Le correzioni sono apportate tempestivamente in risposta ai risultati della verifica.

- **Pianificazione della verifica**: In questa fase, si stabiliscono gli obiettivi e le strategie per la verifica del software. Si definiscono le attività di verifica necessarie, i criteri di accettazione e le risorse richieste per condurre con successo il processo di verifica.
- **Preparazione degli ambienti e degli strumenti di verifica**: Vengono allestiti gli ambienti di test$_G$ e vengono identificati gli strumenti necessari per eseguire le attività di verifica. Questo potrebbe includere l'installazione di software di test$_G$, la configurazione dei sistemi di prova e la preparazione dei dati di test$_G$.
- **Revisione dei requisiti e del design$_G$**: Prima di procedere con lo sviluppo effettivo del software, vengono esaminati e validati i requisiti e il design$_G$ del sistema. Si verifica che i requisiti siano completi, corretti e comprensibili, e che il design$_G$ sia in linea con gli obiettivi e le specifiche del progetto.
- **Analisi statica del codice**: Questa fase coinvolge l'analisi del codice sorgente senza eseguirlo. Obiettivo principale è individuare errori di programmazione, violazioni delle convenzioni di codifica e possibili problemi di design$_G$. Questa analisi viene eseguita manualmente e tramite strumenti automatici di analisi del codice.
- **Test$_G$ del software**: In questa fase il software viene eseguito e testato$_G$ per verificare il suo comportamento rispetto ai requisiti specificati. Questo include test$_G$ funzionali, di integrazione, di sistema e di accettazione, che mirano a identificare difetti e verificare la conformità del software.
- **Analisii dei risultati dei test$_G$**: Una volta completati i test, vengono analizzati i risultati per identificare difetti, errori o problemi nel software. Questi difetti vengono documentati e tracciati per garantire che vengano risolti in modo tempestivo.
- **Gestione della configurazione**: Durante tutto il processo di verifica, è essenziale gestire la configurazione del software e dei suoi componenti per assicurare la tracciabilità e la riproducibilità degli ambienti di test$_G$ e dei risultati ottenuti.

Il processo di verifica è un ciclo continuo e iterativo, che può ripetersi più volte durante lo sviluppo del software per garantire la qualità del prodotto finale. La documentazione accurata di tutte le attività di verifica e dei risultati ottenuti è essenziale per garantire la trasparenza e la tracciabilità del processo.

5.1.2 Validazione

Scopo
Nel contesto dello sviluppo di un progetto software, il processo di supporto di validazione ha lo scopo di confermare che il software soddisfi le esigenze e le aspettative degli utenti finali e che sia in grado di operare efficacemente nel contesto previsto. Mentre il processo di verifica si concentra sulla conformità del software ai requisiti specificati, il processo di validazione si concentra sulla verifica che il software sia appropriato per l'uso previsto e che fornisca il valore desiderato agli utenti.
Aspettative
Le aspettative del processo di validazione sono principalmente:

- **Confermare l'adeguatezza del software per l'uso previsto**: Si prevede che il software validato sia appropriato per soddisfare le esigenze e le aspettative degli utenti finali. Questo significa che il software dovrebbe essere in grado di fornire valore aggiunto agli utenti e di supportare efficacemente i loro processi o attività.
- **Accertarsi che il software soddisfi i requisiti utente**: Le aspettative includono la conferma che il software sia conforme ai requisiti utente specificati. Gli utenti finali dovrebbero essere soddisfatti delle funzionalità, delle prestazioni e dell'usabilità del software, confermando che risponde alle loro esigenze.
- **Identificare e risolvere eventuali problemi**: Durante il processo di validazione, ci si aspetta di individuare eventuali difetti, errori o problemi di usabilità nel software. L'obiettivo è risolvere questi problemi in modo tempestivo e efficace per garantire che il software sia pronto per il rilascio.
- **Confermare la stabilità e l'affidabilità del software**: Le aspettative includono la verifica che il software sia stabile, affidabile e operi correttamente in scenari reali. Questo è importante per garantire che il software possa essere utilizzato in modo sicuro e efficace dagli utenti finali senza rischi di malfunzionamenti o perdite di dati.
- **Accertare il rispetto dei requisiti non funzionali**: Si prevede che il software validato soddisfi anche i requisiti non funzionali specificati, come le prestazioni, la scalabilità e la sicurezza. Questi aspetti sono cruciali per garantire che il software sia in grado di gestire carichi di lavoro reali e di proteggere i dati sensibili.

Descrizione
Il processo di validazione è il momento culminante in cui il prodotto sviluppato viene testato e confrontato con i requisiti del proponente. Questa fase include una serie di test, sia funzionali che non funzionali, per garantire che il prodotto soddisfi le aspettative. I risultati vengono documentati nel piano di qualifica, evidenziando il grado di conformità e identificando eventuali aree di miglioramento.

- **Definizione degli obiettivi di validazione**: La prima fase consiste nel definire chiaramente gli obiettivi di validazione del software. Questi obiettivi dovrebbero includere la conferma che il software soddisfi i requisiti utente, che sia conforme alle normative e che sia adatto all'uso previsto.
- **Pianificazione della validazione**: Vengono identificate le risorse necessarie per eseguire il processo di validazione, inclusi tempo, persone e strumenti. Si pianificano anche le attività di validazione, stabilendo i criteri di accettazione e i metodi di valutazione.
- **Convalida dei requisiti utente**: Questa fase coinvolge la verifica che il software soddisfi i requisiti utente specificati. Gli utenti finali possono essere coinvolti per revisionare i requisiti e confermare che il software risponda alle loro esigenze e aspettative.
- **Test$_G$ di accettazione utente (UAT)**: Durante il UAT, gli utenti finali eseguono test$_G$ specifici per confermare che il software sia conforme alle loro esigenze e che sia utilizzabile in modo efficace nel contesto operativo previsto. I risultati del UAT vengono documentati e utilizzati per valutare il successo della validazione.

- **Validazione delle funzionalità e dei requisiti non funzionali**: Vengono eseguiti test$_G$ specifici per confermare che il software fornisca tutte le funzionalità previste e che soddisfi i requisiti non funzionali, come le prestazioni, la sicurezza e l'affidabilità.
- **Risoluzione dei problemi e revisione**: Durante il processo di validazione, potrebbero emergere difetti o problemi che devono essere risolti. Si risolvono questi problemi e si esegue una revisione finale per assicurarsi che il software sia pronto per il rilascio.
- **Approvazione della validazione**: Una volta completato con successo il processo di validazione e risolti tutti i problemi identificati, il software viene approvato per il rilascio. L'approvazione viene documentata e archiviata come parte del processo di gestione della qualità del progetto.

Il processo di validazione è fondamentale per garantire che il software sia adatto all'uso previsto dagli utenti finali e che soddisfi le loro esigenze e aspettative.

5.2 Gestione della configurazione

La gestione della configurazione è essenziale per tracciare e controllare le modifiche apportate ai componenti del progetto. Utilizzando strumenti di controllo di versione$_G$, il team assicura la coerenza e la tracciabilità di tutte le versioni$_G$ del codice sorgente, della documentazione e di altri artefatti. Questo approccio contribuisce a evitare conflitti e garantisce la riproducibilità del progetto in qualsiasi punto temporale. Il nostro gruppo ha deciso di utilizzare per la gestione della configurazione il servizio GitHub$_G$, basato sul sistema di controllo di versione distribuito Git$_G$.
Il link alla Repository$_G$ pubblica del progetto è il seguente:
https://github.com/CyberSorceres/CyberSorceresRepository

Gestione della repository

- All'interno della Repository$_G$ è presente una cartella "documenti" contenente i documenti versionati del progetto
- All'interno della cartella "documenti" è presente una cartella "verbali" con al suo interno i verbali, interni caricati nel formato tex ed esterni in formato pdf
- per il Proof of Concept$_G$ (PoC), è stato creato una repository$_G$ separata, contenente tutti i file sorgente relativi al PoC$_G$ che sarà consegnato in occasione della prima revisione RTB.

5.3 Norme di utilizzo per la repository

Di seguito descriviamo le norme poste per il mantenimento dell'ordine della Repository$_G$ per tutta la durata del progetto:

- segnalare problemi (issue$_G$) e proporre modifiche (pull request$_G$). Il team si impegna a risolvere gli issue$_G$in modo tempestivo e valutare attentamente le pull request$_G$ per attuare i miglioramenti necessari
- flusso di lavoro con Branches$_G$: La repository$_G$ segue un flusso di lavoro basato su branches$_G$, con un branch principale stabile e branch di sviluppo separati. Ciò consente di isolare nuove funzionalità o correzioni di bug prima di integrarle nella versione$_G$ principale
- nominare i tutti i file seguendo le convenzioni decise dal gruppo e caricarli al suo interno nel formato deciso

5.4 Documentazione

La documentazione è un pilastro fondamentale del processo. Tutti i documenti sono gestiti in modo rigoroso, versionati e archiviati. La struttura documentale, definita nel piano di progetto, è regolarmente aggiornata per riflettere le modifiche apportate durante lo sviluppo del progetto. La documentazione serve come riferimento chiave per tutti i membri del team e per gli stakeholder$_G$ interessati. I documenti prodotti dal gruppo possono essere divisi in due categorie: formali e informali.

5.4.1 Documenti informali

I documenti informali hanno lo scopo di essere di facile utilizzo e manutenzione ma non sono documenti ufficiali. Si parla quindi di documenti che non ancora stati approvati dal responsabile del progetto, bozze, appunti e documenti che non necessitano di versionamento. Come team abbiamo deciso di usufruire di Google Drive, piattaforma che ci permette di comunicare sincronicamente. Il team lo ha utilizzato ad esempio per la creazione di tabelle dove ogni membro avrebbe potuto aggiornare il monte ore svolto per quel ruolo, o comunicare agli altri membri i propri impegni settimanali e facilitare la struttura organizzativa interna. Gli strumenti utilizzati per questo tipo di documenti sono:

- Google Docs
- Google Drive
- Draw.io
- Overleaf

5.4.2 Documenti formali

Di seguito viene indicata la documentazione formale elaborata.

Piano di qualifica

Lo scopo di un piano di qualifica è definire un quadro dettagliato per assicurare che il software sviluppato soddisfi gli standard di qualità prestabiliti e le specifiche dei requisiti. Questo piano, parte integrante della gestione della qualità del progetto, delineerà le strategie, le procedure e le risorse necessarie per garantire che il prodotto software sia affidabile, efficace e conforme alle aspettative del cliente. Il **Piano di qualifica** viene redatto dal **verificatore** e comprende le azioni indispensabili per assicurare l'eccellenza del prodotto e dei processi. Esso è composto dalle seguenti sezioni:

- Qualità di processo;
- Qualità di prodotto;
- Specifica dei test;
- Resoconto attività di verifica.

Piano di progetto

Lo scopo di un piano di progetto è definire in modo chiaro e dettagliato come il processo di sviluppo del software sarà pianificato, gestito e controllato durante tutto il ciclo di vita$_G$ del progetto. Il piano di progetto fornisce una guida strategica e operativa per il team di sviluppo e gli altri stakeholder$_G$ coinvolti, stabilendo obiettivi, tempi, risorse e responsabilità. Di seguito vengono indicate le sezioni del documento **Piano di progetto**:

- Analisi dei rischi;

- Modello di sviluppo;
- Pianificazione;
- Preventivo;
- Consultivo di periodo;
- Attualizzazione dei rischi.

Norme di way of working$_G$

Lo scopo di un documento è fornire linee guida dettagliate e standardizzate sulle pratiche e i processi che devono essere seguiti durante lo sviluppo del software. Queste norme sono progettate per stabilire un quadro coerente e uniforme per l'intero team di sviluppo, garantendo coerenza, qualità e efficienza nelle attività quotidiane. Di seguito vengono indicate le sezioni del documento **Way of working**:

- Processi primari;
- Norme;
- Processi organizzativi;
- Processi di supporto;

Analisi dei requisiti

Lo scopo è definire in modo completo e dettagliato le esigenze e le specifiche del sistema software che deve essere sviluppato. Questo documento svolge un ruolo fondamentale nella fase iniziale del ciclo di vita$_G$ del progetto, servendo come base per la progettazione, lo sviluppo, il test$_G$ e la valutazione del prodotto software. Di seguito vengono indicate le sezioni del documento **Analisi dei requisiti**:

- User Cases (dall'UC1 all' UC18);
- Requisiti

Glossario

Lo scopo di questo documento è quello di contenere tutte le definizioni necessarie alla totale comprensione della documentazione redatta. I termini che presentano la lettera G come pedice saranno presenti, in ordine alfabetico, all'interno del glossario. I termini saranno letteralmente riportati o saranno riportate delle declinazioni molto simili del termine.

5.4.3 Verbali interni

Gli scopi di un verbale interno sono principalmente i seguenti:

- Registro delle Decisioni: Registra le decisioni prese durante incontri interni del team di sviluppo. Queste decisioni possono riguardare questioni tecniche, piani di progetto, assegnazioni di compiti, ecc.
- Comunicazione Interna: Fornisce un mezzo formale per comunicare informazioni rilevanti tra i membri del team. Può includere aggiornamenti sullo stato delle attività, problemi risolti, ostacoli incontrati e soluzioni proposte.à
- Tracciamento delle Attività: Documenta le attività in corso, le discussioni e gli accordi raggiunti durante il processo di sviluppo. Aiuta nel monitoraggio del progresso e nell'identificazione di eventuali ritardi o sfide.

- Archivio Storico: Costituisce un archivio storico delle decisioni e delle attività, facilitando il riferimento futuro e la comprensione del contesto per i membri del team.
- Risposta a Rischi e Problemi: Registra le discussioni relative a rischi o problemi emersi durante lo sviluppo e le strategie di mitigazione o risoluzione proposte.

I verbali interni vengono redatti da un membro del gruppo presente alla riunione, e vengono compilati tramite i template $_G$ creati dal gruppo, e vengono resi disponibili il prima possibile all'interno della Repository$_G$ in modo tale da aggiornare nella maniera più celere i membri assenti alla riunione. I verbali interni non necessitano di firma, e vengono versionati$_G$, rispettando le norme sopra citate.

5.4.4 Verbali esterni

I verbali esterni nascono per i seguenti scopi:

- Comunicazione con gli Stakeholder$_G$: Fornisce una documentazione formale delle discussioni e delle decisioni prese durante incontri con gli stakeholder esterni, come clienti, partner o altri team coinvolti nel progetto.
- Conferma di Accordi: Serve come conferma scritta degli accordi raggiunti tra le parti, riducendo il rischio di malintesi o interpretazioni divergenti.
- Documentazione di Riunioni con Clienti: Registra le discussioni avute con il cliente, comprese le richieste specifiche, le risposte alle domande e le spiegazioni dettagliate fornite durante le interazioni.
- Monitoraggio dello Stato del Progetto: Comunica agli stakeholder$_G$ esterni lo stato del progetto, i progressi compiuti e le sfide incontrate. Contribuisce a mantenere un elevato livello di trasparenza e fiducia.
- Base per l'Approvazione: Può essere utilizzato come documento di riferimento per ottenere l'approvazione formale da parte degli stakeholder$_G$ su decisioni, modifiche o altri aspetti critici del progetto. I verbali esterni infatti necessitano di firma da parte del proponente.

I verbali esterni vengono redatti da un membro del gruppo presente alla riunione, e vengono compilati tramite i template $_G$ creati dal gruppo, e vengono resi disponibili il prima possibile all'interno della Repository$_G$ dopo essere stati mandati ai proponenti. Necessario è infatti che, vista la tipologia di informazioni archiviate all'interno di questi documenti, necessitano di firma da parte del proponente e del Responsabile del gruppo. Infine vengono versionati$_G$, rispettando le norme sopra citate.

5.4.5 Struttura dei documenti e vincoli formali

Nome dei documenti

Ogni documento viene nominato tramite una convezione decisa dal gruppo. I nomi dei documenti è in carattere interamente minuscolo, ogni parola suddivisa da un underscore e successivamente il versionamento, fatta eccezione dei verbali in cui al posto del versionamento è presente la data dell'incontro in forma "giorno"_"mese"_"anno".

Prima pagina

- Presenterà sempre il nome del documento, il nome del team e il logo scelto
- Tabella con i nomi dei membri del gruppo e una tabella con informazioni riguardanti il documento (come i suoi destinatari).

- Sarà poi presente un Changelog$_G$ con il verionamento del file relativo alla modifica messa in atto, oltre al suo autore, verificatore, una breve spiegazione della modifica e la data in cui questa è stata messa in atto.

5.5 Header

Sarà presente in ogni documento un header contenente il nome del documento, il nome del team e il versionamento.

Versionamento

Ogni documento che non sia un verbale viene versionato secondo la norma sopra citata (vedi sezione 3.3). Vengono versionati tutti i documenti fatta eccezione, per loro natura, dei verbali interni ed esterni.

Indici

Successivamente agli elementi citati nella sezione 5.3.5 sarà presente un indice contenente tutte le sezioni del documento. Al termine del documento sarà presente inoltre un indice contenente il riferimento a tutti gli elementi grafici (tabelle ed immagini) presenti.

5.5.1 Elementi grafici

5.5.2 Tabelle

Le tabelle, all'interno dei documenti devono essere centrate orizzontalmente rispetto alla pagina. Ogni tabella all'interno dei documenti è contrassegnata da una didascalia descrittiva di ciò che l'immagine rappresenta. La didascalia è posta al di sotto della tabella, è centrata orizzontalmente rispetto alla pagina e ha il seguente sintassi:

Immagine [n]: Descrizione

Dove:

n rappresenta il numero assoluto della tabella all'interno del documento;

- Descrizione, contiene una breve descrizione del contenuto della tabella.

5.5.3 Immagini

Le immagini presenti sono state create usando due diversi strumenti grafici:

- Canva: utilizzato per la creazione del logo del team.
- Draw.io: per le immagini rappresentative degli Use Case.

Le immagini, all'interno dei documenti devono essere centrate orizzontalmente rispetto alla pagina. Ogni immagine all'interno dei documenti è contrassegnata da una didascalia descrittiva di ciò che l'immagine rappresenta. La didascalia è posta al di sotto dell'immagine, è centrata orizzontalmente rispetto alla pagina e ha il seguente sintassi:

Immagine [n]: Descrizione

Dove:

n rappresenta il numero assoluto dell'immagine all'interno del documento;

- Descrizione, contiene una breve descrizione del contenuto dell'immagine.

Eventuali aggiunte di diagrammi uml all'interno dei documenti verranno inseriti nei documenti come immagini.

5.5.4 Grafici

I grafici, all'interno dei documenti devono essere centrati orizzontalmente rispetto alla pagina. Ogni grafico all'interno dei documenti è contrassegnato da una didascalia descrittiva di ciò che rappresenta. La didascalia è posta al di sotto del grafico, è centrato orizzontalmente rispetto alla pagina e ha il seguente sintassi:

Immagine [n]: Descrizione

Dove:

n rappresenta il numero assoluto del grafico all'interno del documento;

- Descrizione, contiene una breve descrizione del contenuto del grafico.

Glossario

Scopo del documento

Il Glossario è un documento dove sono presenti tutte le definizioni delle parole che il gruppo ha ritenuto necessitassero di una definizione per facilitare la comprensione dei documenti.

Norme di compilazione

- I termini sono ordinati in ordine alfabetico e sono facilmente individuabili all'interno del Glossario tramite una legenda presente all'inizio del documento.
- I termini sono ordinati in ordine alfabetico e sono facilmente individuabili all'interno del Glossario tramite una legenda presente all'inizio del documento.
- Le parole contenute nel Glossario sono presenti nei documenti e sono, per ogni loro occorrenza, indicate con una "G" al pedice.
- Il team ha deciso di escludere dal glossario le parole contenute nei titoli delle sezioni e nelle didascalie delle immagini, poichè già certamente presenti all'interno del testo dei documenti.
- Il Glossario viene aggiornato sulla piattaforma Overleaf da ogni membro ogni qualvolta viene compilata una sezione testuale, in modo tale che l'aggiunta dei termini sia costante e completa.
- Prima di aggiungere una parola nel Glossario ogni membro più velocemente verificare se la parola è già presente al suo interno tramite la legenda.

5.6 Verbali

Esterni

Ogni verbale esterno avrà la seguente struttura:

- Titolo del documento
- Logo del team
- Data della riunione, orario della riunione, metodologia di svolgimento (in presenza o da remoto) e redattore del verbale
- Una tabella contenente i presenti e gli assenti alla riunione
- La firma del responsabile interno e quella del responsabile esterno del progetto

- Un header per ogni pagina con il nome del team, il tipo di documento e la data della riunione
- Obiettivo della riunione: sezione nella quale viene identificato il motivo per il quale si è tenuta la riunione e ciò che si vuole discutere.
- Dubbi: sezione contenente i dubbi emersi durante la presentazione del lavoro
- Decisioni: lista di decisioni discusse e messe in atto dal team a seguito dell'esito della riunione
- Azioni: in cui vengono stilate eventuali azioni svolte durante la riunione
- Attività future: vengono elencate le attività che il team ha intenzione di svolgere nel successivo Sprint.
- Eventuali note

Interni

Ogni verbale esterno avrà la seguente struttura:

- Titolo del documento
- Logo del team
- Data della riunione, orario della riunione, metodologia di svolgimento (in presenza o da remoto) e redattore del verbale
- Una tabella contenente i presenti e gli assenti alla riunione
- Un header per ogni pagina con il nome del team, il tipo di documento e la data della riunione
- Obiettivo della riunione: sezione nella quale viene identificato il motivo per il quale si è tenuta la riunione e ciò che si vuole discutere.
- Dubbi: sezione contenente i dubbi emersi durante la presentazione del lavoro
- Decisioni: lista di decisioni discusse e messe in atto dal team a seguito dell'esito della riunione
- Azioni: in cui vengono stilate eventuali azioni da svolgere nel successivo sprint, associando il compito ad un membro o più del gruppo ed indicando il termine massimo per il suo completamento.
- Attività future: vengono elencate le attività che il team ha intenzione di svolgere nel successivo Sprint.
- Eventuali note