NORME DI PROGETTO

BOT4ME - IMOLA INFORMATICA

seven.solutions.unipd@gmail.com

INFORMAZIONI DOCUMENTO

Versione	1.0.0
Uso	interno
Stato	approvato
Destinatari	Prof. Tullio Vardanega
	Prof. Riccardo Cardin
	Seven Solutions
Redattori	Gambirasio Leonardo
	Cavaliere Alessandro
	Galtarossa Marco
	Bonato Manuele
	Marangon Marco
	Filippi Gabriele
	Ruffin Filippo
Verificatori	Cavaliere Alessandro
	Marangon Marco
Approvazione	Galtarossa Marco

Descrizione

Insieme delle norme che il gruppo è tenuto a rispettare per allinearsi correttamente al Way of Working scelto per questo progetto.

Registro delle modifiche

Versione	Data	Autore	Ruolo	Descrizione
1.0.0	Galtarossa Marco	11-02-2022	Responsabile	Approvazione
0.2.0	Marangon Marco	10-02-2022	Verificatore	Verifica
0.1.3	Gambirasio Leonardo	07-02-2022	Amministratore	Aggiornamento workflow §3.2
0.1.2	Filippi Gabriele	18-01-2022	Progettista	Aggiornamento tecnologie §4.2
0.1.1	Bonato Manuele	15-12-2021	Amministratore	Aggiornamento metodi comunicazione §4.1
0.1.0	Cavaliere Alessandro	08-12-2021	Verificatore	Verifica sezioni
0.0.8	Bonato Manuele	03-12-2021	Amministratore	Stesura parziale §4.1 e stesura §4.2
0.0.7	Filippi Gabriele	03-12-2021	Responsabile	Stesura parziale §4.1
0.0.6	Marangon Marco	01-12-2021	Amministratore	Stesura §3.4, §3.5 e §3.6
0.0.5	Ruffin Filippo	30-12-2021	Amministratore	Stesura §3.2 e §3.3
0.0.4	Gambirasio Leonardo	29-12-2021	Amministratore	Stesura §2.2
0.0.3	Cavaliere Alessandro	25-11-2021	Amministratore	Stesura §2.1
0.0.2	Galtarossa Marco	23-11-2021	Responsabile	Stesura §3.1
0.0.1	Cavaliere Alessandro	22-11-2021	Amministratore	Stesura §1

Indice

1 Introduzione

1.1 Scopo del documento

Il seguente documento definisce le linee guida che tutti i membri del gruppo devono conoscere e rispettare durante tutta la durata del progetto.
L'approccio per la stesura del documento è incrementale, in modo da inserire man mano al suo interno solamente le norme pattuite a seguito dei vari incontri.

1.2 Scopo del capitolato

Il capitolato C1 prevede la creazione di un chatbot segretariale, con la quale un qualsiasi dipendente dell'azienda Imola Informatica possa interagire.
Le finalità del bot sono quelle di facilitare le operazioni di: ingresso nell'azienda (presenze), consuntivazione (attività giornaliere), organizzazione di meeting e generazione di ticket.

1.3 Organizzazione sezioni

Il documento si articola in 3 sezioni principali:

- **Processi primari**
- **Processi di supporto**
- **Processi organizzativi**

Ogni sezione si articola nei vari processi che il gruppo ha deciso di istanziare. Per ogni processo è presente la stessa struttura, che prevede sempre le seguenti parti:

- **Descrizione**: cosa tratta il processo.
- **Scopo**: quali sono gli obiettivi del processo.
- **Attività**: da quali attività è composto. Possono essere presenti i seguenti campi:
  - **Descrizione**: in cosa consiste l'attività.
  - **Procedure**: come si svolge l'attività.
  - **Vincoli**: regole imposte da altre attività.
  - **Documentazione**: documenti prodotti durante lo svolgimento.
  - **Varie**: altri campi utili a definire meglio alcuni aspetti.
  - **Metriche di qualità**: misurazioni per garantirne qualità.
  - **Strumenti**: tecnologie utilizzate per la sua realizzazione.
- **Metriche di qualità**: misurazioni per garantirne qualità.
- **Strumenti**: tecnologie utilizzate per la gestione del processo.

1.4 Annotazioni

Qualora ci fossero ambiguità riguardandi i termini utilizzati, verranno inserite a piè di pagina delle annotazioni per delucidare eventuali dubbi.

1.5 Riferimenti

1.5.1 Normativi

- Capitolato C1, Bot4Me - Imola Informatica

1.5.2 Informativi

- Standard ISO/IEC 12207-1995
- Standard ISO/IEC 25010-2011
- Slide del corso - Progettazione software
- Slide del corso - Regolamento del progetto didattico

2 Processi primari

2.1 Fornitura

2.1.1 Descrizione

Determinare le principali attività organizzative che il gruppo dovrà attuare per la scelta del capitolato a cui candidarsi e per redigere i documenti necessari a garantirne qualità (PdQ) ed efficienza (PdP).

2.1.2 Scopo

- Individuare competenze e risorse richieste per la realizzazione di ogni capitolato.
- Fornire un documento su come il gruppo organizza il lavoro e le scadenze.
- Fornire metriche per misurare la qualità del lavoro in ogni fase del progetto.

2.1.3 Attività

2.1.3.1 Analisi capitolati

Descrizione

Ottenere una visione d'insieme sulle offerte proposte al fine di scegliere il capitolato al quale il gruppo decide di candidarsi per la sua realizzazione.

Procedure

- **Studio autonomo**: ogni membro del gruppo analizza in autonomia i vari capitolati proposti, evidenziando:
  1. Tecnologie e linguaggi di programmazione necessari per la realizzazione.
  2. Pregi e difetti.
- **Scelta del capitolato**: durante un incontro con tutti i membri del gruppo, ognuno voterà 2 preferenze. Il capitolato con maggiori voti sarà quello scelto dal gruppo per la candidatura.

2.1.3.2 Redazione Piano di Progetto

Descrizione

Attività successiva alla conferma di candidatura per il capitolato scelto.
Redigere un documento che contenga il piano di lavoro del gruppo.

Procedure

Redazione del documento, suddiviso nelle seguenti sezioni:

- **Analisi dei rischi**: rischi potenziali e rischi incontrati nelle varie fasi.
- **Modello di sviluppo**: sistema di lavoro scelto dal gruppo.
- **Pianificazione**: successive fasi di lavoro.
- **Costi preventivati**: preventivo ore e conseguenti costi per ogni periodo.
- **Analisi a consuntivo**: consuntivo ore e costi effettivamente sostenuti per ogni periodo.

2.1.3.3 Redazione Piano di Qualifica

Descrizione

Attività successiva alla conferma di candidatura per il capitolato scelto.
Redigere un documento contenente le misure di qualità che il gruppo deve rispettare per garantire la buona riuscita del progetto e del prodotto.

Procedure

Redazione del documento, suddiviso nelle seguenti sezioni:

- **Qualità di processo**: metriche per garantire la qualità dei processi.
- **Qualità di prodotto**: metriche per garantire la qualità delle componenti prodotte.
- **Testing**: specifica (a vari livelli) dei test da effettuare sulle componenti software.
- **Attività di verifica**: resoconto delle verifiche effettuate nei vari periodi.
- **Miglioramenti**: valutazioni per migliorare il way of working.

2.1.3.4 Revisione

Descrizione

Coordinare le attività di revisione e interazione con il proponente e il committente.

Procedure

- **Definizione revisioni**: quando e come effettuarle.
- **Comunicazioni esterne**: definire i mezzi di comunicazione con proponente e committente.

2.1.3.5 Verifiche e consegna finale

Descrizione

Determinare le attività di verifica e validazione del prodotto.

Procedure

- **Verifica e validazione**: controlli sugli oggetti prodotti dal gruppo (documenti e prodotto).
- **Rilascio**: concordare con proponente e committente le modalità di rilascio del prodotto.

2.1.4 Metriche di qualità

Nessuna in particolare.

2.1.5 Strumenti

Nessuno in particolare.

2.2 Sviluppo

2.2.1 Descrizione

Definire tutte le operazioni necessarie alla realizzazione del prodotto finale.

2.2.2 Scopo

Definire e normare le attività di:

- **Analisi dei requisiti**: comprensione del problema.
- **Progettazione**: realizzazione di una soluzione al problema.
- **Codifica**: implementazione della soluzione trovata.

2.2.3 Attività

2.2.3.1 Analisi dei requisiti

Descrizione
Analisi capitolato e ambiente d'uso del prodotto, per estrapolarne i requisiti.

Procedure
Redazione del documento contenente i requisiti trovati, con le seguenti sezioni:

- **Descrizione**: analisi del prodotto.
- **Casi d'uso**: scenari designabili nel prodotto.
- **Requisiti**: classificazione requisiti e funzionalità del prodotto.
- **Tracciamento**: mappatura requisiti individuati.

Definizione requisiti
Ogni requisito viene definitio nel seguente modo:

- Codice identificativo
- Descrizione
- Fonte/i

Il codice indentificativo prevede la seguente sintassi:

REQ[Priorità] - [Tipologia] - [Identificativo]

dove:

- **REQ**: identifica che si tratta di un requisito.
- **Priorità**: grado di priorità, può assumere i valori:
  - **A**: obbligatorio, necessario.
  - **B**: desiderabile, non strettamente necessario.
  - **C**: opzionale, non necessario.
- **Tipologia**: tipo di requisito, può assumere i valori:

  - **F**: funzionale
  - **P**: prestazionale
  - **Q**: qualitativo
  - **V**: vincolo

- **ID**: numero progressivo, rappresentato in forma gerarchica:

[codicePadre].[codiceFiglio]

Definizione casi d'uso

Ogni caso d'uso viene definito nel seguente modo:

UC[codiceCaso].[codiceSottoCaso].[codiceSottoSottoCaso]

È descritto da:

- **Diagramma UML**
- **Attori primari**
- **Attori secondari**
- **Descrizione**
- **Estensioni**: scenario alternativo.
- **Inclusioni**
- **Precondizione**: stato del sistema affinchè si abbia a disposizione la funzionalità.
- **Post-condizione**: stato del sistema dopo gli eventi del caso d'uso.
- **Scenario principale**: flusso degli eventi, in forma di elenco numerato.

Metriche di qualità

Nessuna in particolare.

Strumenti

- **Editor UML**: draw.io.

2.2.3.2 Progettazione

Descrizione

Attività successiva all'analisi dei requisiti.
Definire una soluzione al problema secondo le aspettative concordate con il proponente.

Procedure

- **Progettazione architetturale**

  - **Input**: Analisi dei Requisiti.
  - **Output**: Technology Baseline.

  - Definizione di specifiche, architettura e parti del prodotto ad alto livello.
  - Individuare interazioni tra le parti e definizione dei test d'integrazione (TI).

- **Progettazione di dettaglio**: successiva a progettazione di dettaglio.
  - **Input**: Tecnology Baseline.
  - **Output**: Product Baseline.
  - Definizione specifiche e componenti del prodotto in modo dettagliato, dal sistema nel suo complesso alle singole unità.
  - Argomentazione della soluzione proposta attraverso diagrammi UML di vario tipo (classi, attività, sequenza).

Vincoli

- **Architetturali**
  - **Efficacia**: il prodotto deve soddisfare tutti i requisiti individuati durante l'analisi dei requisiti.
  - **Efficienza**: il prodotto deve contenere parti quando più possibili riusabili, manutenibili e ben documentate.
  - **Gestione complessità**: il sistema deve essere suddiviso in unità sufficientemente piccole, in modo da agevolarne la codifica e la verifica.
- **Architettura logica**: da perseguire:
  - **Sufficienza**: soddisfa tutti i requisiti.
  - **Comprensibilità**: comprensibile al proponente.
  - **Modularità**: parti chiare e ben distinte, minimizzando le dipendenze.
  - **Robustezza**: buona risposta ad input diversi da parte di utente e ambiente d'uso.
  - **Flessibilità**: modifiche e manutenzione migliorativa agili al variare dei requisiti.
  - **Riusabilità**: possibile riutilizzo delle parti del prodotto.
  - **Efficienza**: efficiente in termini di tempo impiegato e spazio di memoria occupato.
  - **Affidabilità**: svolge bene i task quando utilizzata.
  - **Disponibilità**: tempo di timeout per manutenzione ridotto.
  - **Sicurezza**: robusta a malfunzionamenti e intrusioni.
  - **Semplicità**: ogni parte non ha componenti superflue.
  - **Incapsulazione**: struttura interna nascosta dall'esterno, ridurre dipendenze.
  - **Coesione**: ogni componente è composta da parti che perseguono lo stesso fine
  - **Basso accoppiamento**: la dipendenza tra le parti è irrisoria o nulla.
  - **Principi SOLID**.
  - **Buone prassi codifica**: rispetto di regole e convenzioni nella fase di codifica.

Technology Baseline

Insieme di oggetti che costituiscono la base architetturale del prodotto. Contiene:

- **Tecnologie utilizzate**: descrizione delle tecnologie, framework e librerie scelte per la realizzazione del prodotto.
- **Proof Of Concept**: dimostratore eseguibile coerente con obiettivi, baseline per il prosieguo del progetto.

- **Tracciamento componenti**: corrispondenza tra componenti e requisiti che esse soddisfano.
- **Test integrazione**: definizione dei TI da eseguire per verificare che le componenti interagiscano in modo corretto tra di loro e producendo i risultati attesi (da requisiti).

Product Baseline

Insieme di documenti che argomentano le scelte architetturali di dettaglio. Contiene:

- **Design pattern**: baseline architetturale del prodotto, coerente e migliorativa rispetto a RTB.
- **Specifica architetturale**: insieme di diagrammi UML.
  - **Classi**: rappresentano parti del sistema con attributi, metodi e relazioni tra di esse.
  - **Sequenza**: descrivono le procedure e le interazioni tra un insieme di oggetti che devono svolgere un determinato task.
  - **Attività**: a supporto dei casi d'uso, descrivono il flusso di operazioni di un processo.
- **Test**
  - **Unità**: verificano il corretto funzionamento delle unità.
  - **Integrazione**: verificano la corretta interazione tra le unità.
  - **Sistema**: verificano il corretto funzionamento del sistema nel suo complesso.
- **Tracciamento delle classi**: corrispondenza tra classi e requisiti che esse soddisfano.

Diagrammi delle classi

Descrivono i tipi di oggetto presenti nel sistema e le relazioni tra essi. Contengono:

- **Nome**: nome della classe.
- **Attributi** (opzionali): rappresentano lo stato interno della classe. Sono identificati dalle seguenti proprietà:
  - **Visibilità**: visibilità dell'attributo rispetto ad altre classi (pubblica, protetta o privata).
  - **Nome**: nome dell'attributo.
  - **Tipo**: tipo di dato dell'attributo.
  - **Molteplicità** (opzionale): occorrenze dell'attributo nella classe.
  - **Default** (opzionale): valore predefinito dell'attributo.
  - **Proprietà aggiuntive** (opzionali).
- **Operazioni** (opzionali): rappresentano le azioni eseguibili dalla classe. Sono identificate dalle seguenti proprietà:
  - **Visibilità**: visibilità dell'operazione rispetto ad altre classi (pubblica, protetta o privata).
  - **Nome**: nome dell'operazione.
  - **Parametri**: lista dei parametri dell'operazione. Ogni parametro è identificato delle seguenti proprietà:

    - **Direzione** (opzionale): modalità di accesso al parametro (lettura, scrittura, entrambe).
    - **Nome**: nome del parametro.
    - **Tipo**: tipo del parametro.
    - **Default** (opzionale): valore predefinito del parametro.
  - **Ritorno**: tipo di ritorno dell'operazione.
  - **proprietà aggiuntive** (opzionali).

- **Relazioni** (opzionali): rappresentano il grado di dipendenza tra due classi:
  - **Dipendenza**: la classe utilizza un oggetto della classe con cui si relaziona.
  - **Associazione**: la classe crea ed utilizza un oggetto della classe con cui si relaziona.
  - **Aggregazione**: la classe possiede come attributo un riferimento ad un oggetto condiviso della classe con cui si relazione.
  - **Composizione**: la classe possiede come attributo un oggetto della classe con cui si relaziona.
  - **Generalizzazione**: la classe è un istanza che specializza la classe con cui si relazione, ereditandone attributi ed operazioni.

Oltre alle classi, in questi diagrammi possono essere utilizzati i seguenti costrutti:

- **Classe astratte**: rappresenta una classe di cui non istanziabile, in quanto possiede almeno un'operazione astratta.
- **Interfaccia**: rappresenta un costrutto simile ad una classe astratta, ma a differenza di quest'ultima non può possedere attributi e tutte le sue operazioni sono obbligatoriamente astratte.

Diagrammi di sequenza

I diagrammi di sequenza descrivono la realizzazione di un determinato comportamento tramite la collaborazione di più oggetti. Si utilizzano i seguenti costrutti:

- **Partecipante**: rappresenta un oggetto che detiene il flusso di esecuzione e collabora alla realizzazione di un comportamento. È composto da due parti:
  - **Nome**: nome dell'oggetto partecipante.
  - **Barra di attivazione**: indicazione della durata del periodo di tempo durante il quale il partecipante è attivo.
- **Messaggio**: operazione di un partecipante che viene chiamata da parte di un altro partecipante con relativi dati scambiati tra i due. Un messaggio può essere:
  - **Sincrono**: il partecipante chiamante attende la risposta del partecipante chiamato prima di proseguire la sua esecuzione.
  - **Asincrono**: il partecipante chiamante non attende la risposta del partecipante chiamato, ma prosegue la sua esecuzione subito dopo la chiamata.
  - **Ritorno**: messaggio di ritorno riferito ad un precedente messaggio di chiamata.
  - **Creazione**: messaggio di creazione di un nuovo partecipante da parte del partecipante chiamante.

  - **Distruzione**: messaggio di distruzione di un partecipante da parte del partecipante chiamante.

- **Frame interazione**: un ciclo o una condizione che coinvolge più messaggi e parti delle barre di attivazione di più partecipanti. È caratterizzato da:
  - **Guardia**: condizione di attivazione del frame, posta in corrispondenza del partecipante coinvolto.
  - **Etichetta**: tipologia del frame. I frame possono essere etichettati come:
    * **alt**: alternativa, è eseguito solo il frame per cui la guardia è verificata.
    * **opt**: opzionale, il frame è eseguito solo se la guardia è verificata.
    * **par**: parallelo, ogni frame è eseguito in parallelo.
    * **loop**: ciclo, il frame può essere eseguito più volte, in base al verificarsi della guardia.
    * **region**: regione critica, il frame può essere eseguito da un solo flusso di esecuzione alla volta.
    * **neg**: negativo, il frame rappresenta un'interazione non valida.
    * **ref**: riferimento, il frame si riferisce ad un'interazione definita in un altro diagramma.
    * **sd**: diagramma di sequenza, il frame comprende un intero diagramma di sequenza.

È possibile definire due tipi di controllo dell'elaborazione durante la collaborazione tra i partecipanti:

- **Centralizzato**: un unico partecipante concentra su di sè la responsabilità della chiamata dei messaggi verso tutti gli altri partecipanti, governando l'intera elaborazione.
- **Distribuito**: ogni partecipante ha dei compiti ben definiti e la responsabilità della chiamata dei messaggi verso gli altri partecipanti coinvolti.

Diagrammi delle attività

I diagrammi delle attività descrivono la logica dei processi che compongono l'applicazione software sviluppata, rappresentando la parte dinamica dei casi d'uso, mostrando le attività che si possono presentare e le sequenze in cui le loro azioni vengono svolte. I costrutti utilizzati in questi diagrammi sono i seguenti:

- **Nodo iniziale**: punto di inizio dell'esecuzione dell'attività.
- **Nodo di fine flusso**: punto di terminazione di un percorso di esecuzione. Può non causare il termine dell'attività, in quanto essa può continuare su altri percorsi.
- **Nodo finale**: punto di terminazione dell'esecuzione dell'attività.
- **Attività**: azione all'interno dell'attività, identificata da una breve descrizione.
- **Sotto-attività**: sotto-attività utilizzata per descrivere un'azione che ne comprende altre al suo interno. Per non rendere diagramma complesso può essere realizzato un diagramma della sotto-attività che ne illustri l'input, l'output e le azioni contenute.
- **Pin**: parametro prodotto o consumato da un'azione, con l'indicazione del suo tipo.
- **Fork**: punto d'inizio di un elaborazione parallela all'interno dell'attività.

- **Join**: punto di sincronizzazione tra più flussi di esecuzione parallela generati in seguito ad una fork.
- **Branch**: punto di scelta di uno tra i possibili percorsi di esecuzione disponibili, in base alla guardia associata a ciascuno di essi.
- **Merge**: punto di unione dei diversi percorsi di esecuzione (non paralleli) generati in seguito ad un branch.
- **Segnale**: evento esterno, generato in modo non bloccante e catturato in modo bloccante, all'interno dell'attività.
- **Timeout**: attesa bloccante all'interno dell'attività, di cui deve essere specificata la durata e l'unità di misura.
- **Evento ripetuto**: evento generato ad intervalli regolari all'interno dell'attività (da specificare durata e unità di misura).
- **Swimlane**: partizione che fornisce una singola responsabilità all'esecuzione delle azioni all'interno di un'attività.
- **Regione di espansione**: rappresenta una sequenza di azioni che vengono ripetute sugli elementi di una collezione.

2.2.3.3 Codifica

Descrizione

Il codice dovrà rispettare le norme stabilite, con il fine di perseguire u buon livello di qualità, conforme alle metriche stabilite. L'attività mira a:

- Ottenere codice leggibile ed uniforme.
- Agevolare le fasi di manutenzione, verifica e validazione.
- Fornire un prodotto conforme alle richieste prefissate dal proponente.
- Realizzare un prodotto di qualità, mantenibile e facilmente estendibile.

Procedure

Per garantire l'uniformità del codice, in base al linguaggio utilizzato, ciascuno sviluppatore dovrà utilizzare i seguenti stili di formattazione:

- **Kotlin**: linee guida fornite dal sito web del linguaggio. Esse corrispondo all'impostazione di stile di base dell'ambiente di sviluppo *Android Studio*.
- **Python**: linee guida fornite dal sito web del linguaggio. Esse corrispondo all'impostazione di stile di base dell'ambiente di sviluppo *PyCharm*.
- **HTML5**: linee guida fornite dal sito web del linguaggio.
- **CSS**: linee guida fornite dal sito web del linguaggio.
- **JavaScript**: linee guida fornite da Google.

Per il rispetto delle linee guida si affidamento ai vari strumenti:

- ***Android Studio***: verifica dello stile di formattazione del linguaggio Kotlin.
- ***PyCharm***: verifica dello stile di formattazione del linguaggio Python.

- **Prettier**: verifica dello stile di formattazione linguaggio JavaScript, con le regole riportate da Google JavaScript Style Guide.
- **W3C**: verifica validità codice HTML5 e CSS3
  - HTML Validator.
  - CSS Validator.

2.2.4 Metriche di qualità

2.2.4.1 QC-1 Sviluppo

Misurare la capacità del prodotto a soddisfare requisiti richiesti dal proponente

MQ-PD-1 Requisiti obbligatori soddisfatti (ROS)

- **Descrizione**: indica quanti requisiti obbligatori sono stati soddisfatti (progettati, implementati e testati). Determina la percentuale di completamento del prodotto.
- **Unità di misura**: %.
- **Formula**:
$$ROS = \frac{\#requisiti\_obbligatori\_soddisfatti}{\#requisiti\_obbligatori\_totali} * 100$$
- **Interpretazione risultato**: diverse situazioni in base a ROS:
  - 0%: nessun requisito obbligatorio è stato soddisfatto.
  - 0% < x < 100%: alcuni requisiti obbligatori sono stati soddisfatti.
  - 100%: tutti i requisiti obbligatori sono stati soddisfatti.

MQ-PD-2 Requisiti desiderabili soddisfatti (RDS)

- **Descrizione**: indica quanti requisiti desiderabili sono stati soddisfatti (progettati, implementati e testati).
- **Unità di misura**: %.
- **Formula**:
$$RDS = \frac{\#requisiti\_desiderabili\_soddisfatti}{\#requisiti\_desiderabili\_totali} * 100$$
- **Interpretazione risultato**: diverse situazioni in base a RDS:
  - 0%: nessun requisito desiderabile è stato soddisfatto.
  - 0% < x < 100%: alcuni requisiti desiderabili sono stati soddisfatti.
  - 100%: tutti i requisiti desiderabili sono stati soddisfatti.

MQ-PD-3 Requisiti opzionali soddisfatti (RPS)

- **Descrizione**: indica quanti requisiti opzionali sono stati soddisfatti (progettati, implementati e testati).
- **Unità di misura**: %.
- **Formula**:
$$RPS = \frac{\#requisiti\_opzionali\_soddisfatti}{\#requisiti\_opzionali\_totali} * 100$$

- **Interpretazione risultato**: diverse situazioni in base a RPS:
  - 0%: nessun requisito opzionale è stato soddisfatto.
  - 0% < x < 100%: alcuni requisiti opzionali sono stati soddisfatti.
  - 100%: tutti i requisiti opzionali sono stati soddisfatti.

MQ-PD-4 Requisiti totali soddisfatti (RTS)

- **Descrizione**: indica quanti requisiti sono stati soddisfatti (progettati, implementati e testati). Determina la percentuale di completamento del prodotto globale.
- **Unità di misura**: %.
- **Formula**:

$$RTS = \frac{\#requisiti\_soddisfatti}{\#requisiti\_totali} * 100$$

- **Interpretazione risultato**: diverse situazioni in base a RTS:
  - 0%: nessun requisito è stato soddisfatto.
  - 0% < x < 100%: alcuni requisiti sono stati soddisfatti.
  - 100%: tutti i requisiti sono stati soddisfatti.

2.2.4.2 QC-2 Affidabilità

MQ-TS-5 Test superati (TS)

- **Descrizione**: indica quanti test sono stati superati rispetto al totale dei test previsti.
- **Unità di misura**: %.
- **Formula**:

$$TS = \frac{\#test\_superati}{\#test\_totali} * 100$$

- **Interpretazione risultato**: diverse situazioni in base a TS:
  - 0%: nessun test è stato superato,prodotto scarsamente affidabile.
  - 0% < x < 100%: alcuni test sono stati superati, prodotto un po' più affidabile
  - 100%: tutti i test sono stati superati, prodotto affidabile.

2.2.4.3 QP-1 Sviluppo

MQ-PS-6 Numero di commit effettuati sviluppo (NCES)

- **Descrizione**: indica quanti commit sono stati effettuati nel repository dedicato al mantenimento del codice.
- **Unità di misura**: numero intero.
- **Fonte**: *GitHub* (*#commit_effettuati* al termine di ogni periodo previsto dal PdP).
- **Interpretazione risultato**: il risultato diventa interessante quando guardato in più fasi successive, in quanto evidenzia le differenze con le fasi precedenti.

2.2.5 Strumenti

- **Android Studio**: codifica e testing componenti scritte in Kotlin.
- **PyCharm**: codifica e testing componenti scritte in Python.
- **Visual Studio Code**: codifica e testing di parti scritte in HTML, CSS3 e JavaScript.

3 Processi di supporto

3.1 Documentazione

3.1.1 Descrizione

Fornire dettagli su come deve essere redatta e verificata la documentazione. Tutte le regole descritte devono essere rispettate in tutti i documenti, interni ed esterni.

3.1.2 Scopo

Definire uno standard unico per la stesura dei documenti, in modo da avere omogeneità sulla documentazione.

3.1.3 Attività

3.1.3.1 Stesura documenti

Descrizione

Normare il processo di stesura dei documenti, definendo diverse fasi della stessa.

Ciclo di Vita

Ogni documento attraverso le seguenti fasi, che sono eventualmente ripetibili:

- **Stesura**
  1. Scrittura del documento, assegnata ad uno o più redattori.
  2. Al termine, il responsabile autorizzerà l'avanzamento alla fase successiva.
- **Verifica**
  1. Controllo che stesura sia avvenuta in modo corretto, sintatticamente e semanticamente, eseguita da uno o più verificatori.
  2. In caso di errori superficiali il verificatore correggerà direttamente. In caso contrario verrà notificato il redattore, riportando il documento in fase di stesura.
  3. Ottenuta la conformità, il responsabile autorizzerà l'avanzamento alla fase successiva.
- **Approvazione**: conferma e rilascio del documento da parte del responsabile.

3.1.3.2 Strutturazione documenti

Descrizione

Definire una struttura generica per il layout dei documenti.

Template LaTeX

Si è deciso di utilizzare un template LaTeX per facilitare:

- Versionamento.
- Stesura con uniformità di layout dei documenti.
- Lavoro asincrono, evitando conflitti da scrittura contemporanea sullo stesso file.

Il file "main.tex" raccoglie sezioni, pacchetti e comandi per la compilazione.

Frontespizio

Fornisce i dati principali del documento, è composto da:

- Logo del gruppo.
- Titolo del documento.
- Nome del progetto a cui fa riferimento.
- Indirizzo e-mail del gruppo.

Inoltre possiede una sezione "*INFORMAZIONI SUL DOCUMENTO*", contenente:

- **Versione**: versione attuale del documento.
- **Uso**: destinazione d'uso, interna o esterna.
- **Stato**: attuale stato, "in redazione" o "approvato".
- **Destinatari**: destinatari del documento.
- **Redattori**: membri del gruppo che si sono occupati della redazione.
- **Verificatori**: membri del gruppo che si sono occupati della verifica.
- **Approvazione**: membro del gruppo che ha approvato il rilascio.

Tutti gli elementi di questa pagina sono centrati e disposti in colonna.

Registro modifiche

Consiste in una tabella contenente le informazioni riguardanti il ciclo di vita del documento. Più precisamente, la tabella riporta per ogni modifica:

- **Versione**: versione del documento relativa alla modifica effettuata.
- **Data**: data in cui la modifica è stata effettuata.
- **Autore**: nominativo della persona che ha effettuato la modifica.
- **Ruolo**: ruolo della persona che ha effettuato la modifica.
- **Descrizione**: breve descrizione della modifica effettuata.

Layout pagine contenuto

Il design del contenuto principale di una pagina prevede:

- **Logo**: in alto a sinstra.
- **Sezione trattata**: in alto a destra.
- **Body**: posto tra header e footer.
- **Linea orizzonatale**: divide body e footer.
- **Versione documento**: in basso a sinistra.
- **Numero pagina corrente**: in basso a destra.

3.1.3.3 Classificazione dei documenti

Descrizione

Catalogare i documenti per tipo e utilità.

Documenti formali

Si dicono formali i documenti:

- Revisionati, verificati ed approvati dal responsabile del progetto.
- Soggetti a versionamento.
- Rilasciabili all' esterno del gruppo.

Questi possono essere ad uso interno o esterno.

- **Interni**: di interesse esclusivamente del gruppo.
- **Esterni**: di interesse del gruppo, del committente e del proponente.

Documenti informali

Si dicono informali i documenti:

- Non ancora approvati dal responsabile del progetto.
- Non necessariamente soggetti a versionamento.

3.1.3.4 Redazione verbali

Descrizione

Redazione di un documento informale che riassume gli argomenti trattati durante una riunione interna o esterna.

Sezioni

- **Informazioni generali**:
  - **Data**: data dell'incontro.
  - **Orario**: orario di inizio e di fine dell'incontro.
  - **Luogo**: luogo o piattaforma in cui si è svolto l'incontro.
  - **Partecipanti**: lista di persone presenti all'incontro.
  - **Motivo**: ragione dell'incontro.
- **Ordine del giorno**: punti trattati durante l'incontro.
  - Dubbi e/o discussioni sull'argomento.
  - Decisioni a cui si è giunti.
- **Considerazioni finali**: breve resoconto dell'incontro e task per il futuro.

3.1.3.5 Redazione altri documenti

Descrizione

Definire quali altri documenti dovranno essere redatti nel corso del progetto.

Sezioni

Per ogni documento in questa sezione non viene definita la struttura, in quanto viene descritta nei processi che sono direttamente legati con la redazione del documento.

Documenti

- **Piano di progetto**
- **Piano di qualifica**
- **Analisi dei requisiti**

3.1.3.6 Definizione norme tipografiche

Descrizione

Determinare norme tipografiche da adottare durante tutta la sua stesura, per attribuire uniformità e coerenza alla documentazione.

Convenzioni sui nomi dei file

I nomi delle directory e dei documenti prodotti rispettano la convenzione *snake case*:

- Utilizzo esclusivo del minuscolo.
- Separatore *underscore* "_" nel caso il nome sia composto da piu parole.
- Non omissione delle preposizioni.

Le estensioni dei file sono escluse da questa convenzione.

Stile del testo

I vari stili del testo hanno una specifica funzione semantica:

- **Corsivo**: utilizzato per denotare termini tecnici appartenenti ad una particolare tecnologia.
- **Grassetto**: utilizzato per evidenziare le parole con la definizione della stessa in seguito, oppure per denotare le sezioni e sottosezioni dei documenti.

Elenchi puntati

- Ogni elemento dell'elenco deve essere seguito da un ".".
- La prima lettera di ogni voce dell'elenco deve essere maiuscola.
- Gli elementi chiave vanno preferibilmente messi, in grassetto, come prima parte del punto.

Gli elenchi avranno punto elenco differente a seconda della loro tipologia:

- **Non ordinati**: pallino pieno ed eventuale trattino o asterisco, per elenchi annidati.
- **Non ordinati**: non è stato stabilito un punto definito. Possono essere usati numeri e letterali (numeri romani, alfabeto, ...).

3.1.3.7 Definizione elementi dei documenti

Descrizione

Definire quali oggetti grafici possono entrare nella documentazione, a scopo integrativo ed esplicativo.

Oggetti grafici

- **Tabelle**: se non esplicative, da accompagnare con didascalia.
- **Immagini**: se non esplicative, da accompagnare con didascalia.
- **Grafici**
  - **Diagrammi UML**: per modellare *Use Case* ed oggetti della progettazione.
  - **Torte**: per dare una visione intuitiva ed immediata di altri dati.
  - **Istogrammi**: per dare una visione intuitiva ed immediata di altri dati.

Formati comuni

- **Data**: formato **DD-MM-YYYY**.
- **Ora**: formato **HH:MM**.
- **Versione**: formato **X.Y.Z**.

Sigle

Per ragioni di scorrevolezza e brevità, nei vari documenti verranno utilizzate abbreviazioni di parole ricorrenti, successivamente elencate per categorie:

- **Revisioni di avanzamento**:
  - **RTB**: Requirement and Technology Baseline.
  - **PB**: Product Baseline.
  - **CA**: Customer Acceptance.
- **Documentazione interna ed esterna**:
  - **AdR**: Analisi dei Requisiti.
  - **NdP**: Norme di Progetto.
  - **PdQ**: Piano di Qualifica.
  - **PdP**: Piano di Progetto.
  - **MU**: Manuale Utente.
  - **V**: Verbale.
- **Ruoli di progetto**:
  - **Re**: Responsabile.
  - **Am**: Amministratore.
  - **An**: Analista.
  - **Pgt**: Progettista.
  - **Pgr**: Programmatore.
  - **Ve**: Verificatore.
- **Altro**:
  - **ML**: Machine Learning.
  - **JS**: Javascript.
  - **PY**: Python.
  - **DB**: Database.

3.1.4 Metriche di qualità

3.1.4.1 QC1 - Comprensione

MQ-PD-7 Indice Gulpease (GULP)

- **Descrizione**: indice che riporta il grado di leggibilità di un testo in lingua italiana.
- **Unità di misura**: numero intero.
- **Formula**:

$$GULP = 89 + \frac{300 * (\#frasi) - 10 * (\#parole)}{\#lettere}$$

- **Interpretazione risultato**: indice con valori compresi tra 0 e 100:
  - $<$ 80: indica una leggibilità difficile per un utente con licenza elementare.
  - $<$ 60: indica una leggibilità difficile per un utente con licenza media.
  - $<$ 40: indica una leggibilità difficile per un utente con licenza superiore.

MQ-PD-8 Correttezza Ortografica (CO)

- **Descrizione**: correttezza ortografica della lingua italiana del documento.
- **Unità di misura**: numero intero.
- **Formula**:

$$CO = \#errori_ortografici$$

- **Interpretazione risultato**:
  - 0: documento non esiste o non ha errori ortografici.
  - $>$ 0: documento esiste e contiene errori ortografici.

3.1.5 Strumenti

- **Stesura**:
  - **LaTeX**: linguaggio di markup basato sul programma di tipografia digitale TEX. Permette di scrivere documenti in maniera modulare e collaborativa.
  - **Editor di testo**: nessun editor predefinito, in quanto ogni componente del gruppo ha familiarità con editor ed ambienti differenti.
- **Oggetti grafici**:
  - **GanttProject**: realizzazione grafici di Gantt.
  - **Editor UML**: draw.io.
  - **Microsoft Excel** - **LibreOffice Calc**: torte e istogrammi.

3.2 Gestione della configurazione

3.2.1 Descrizione

Insieme di attività volte a predisporre lo spazio di lavoro del gruppo, cercando di semplificare e automatizzare i processi di mantenimento dei documenti degli oggetti software.

3.2.2 Scopo

Rendere usabile l'ambiente di lavoro, con controlli automatizzati su attività e metodi di lavoro. Normare come avviene il versionamento degli oggetti prodotti dal gruppo.

3.2.3 Attività

3.2.3.1 Versionamento e rilascio

Descrizione

Per ogni oggetto prodotto dal gruppo, sia questo un documento o una componente software, si definiscono le seguenti norme di versionamento.

Documentazione

Tutti i file che riguardano la documentazione vengono conservati in una repository diversa da quella dedicata al mantenimento del codice. Ogni documento viene versionato attraverso un nominativo univoco.

Ci si attiene alla seguente sintassi di versionamento:

$$\text{v}[x].[y].[z]$$

- $[x]$: numero identificativo del rilascio:
  - Parte da 0 e non si resetta mai.
  - Incrementabile dal responsabile, quando il documento viene approvato al rilascio.
- $[y]$: numero identificativo che rappresenta un avanzamento sostanziale delle revisioni:
  - Parte da 0 e si resetta quando $[x]$ viene incrementato.
  - Incrementabile solo dai revisori.
- $[z]$: numero identificativo che rappresenta una aggiunta, modifica o eliminazione:
  - Parte da 0 e si resetta quando $[x]$ o $[y]$ vengono incrementati.
  - Incrementato da un redattore o un revisore in base alle situazioni.

Il rilascio può avvenire nel momento in cui il documento viene approvato dal responsabile.

Le modifiche ai documenti vengono gestite seguendo il **Registro delle modifiche** presente in tutti i documenti successivamente al frontespizio.
In tale registro si tiene traccia di ogni aggiunta, modifica, eliminazione, revisione o approvazione. Maggiori dettagli sul formalismo al paragrafo 3.1.3.2.

Componenti software

I sorgenti del software che riguardano la codifica del prodotto sono mantenuti in una repository differente da quella dedicata alla documentazione.
Ogni file viene versionato tramite uno storico delle modifiche.

Ci si attiene alla seguente sintassi di versionamento:

$$\text{v}[x].[y].[z] - [s]$$

- $[x]$: numero identificativo del rilascio del software per una *major release*:
  - Parte da 0 e non si resetta mai.
  - Incrementato solo dopo aver implementato tutti i requisiti obbligatori.
- $[y]$: numero identificativo per una *minor release*:
  - Parte da 0 e si resetta quando $[x]$ viene incrementato.
  - Incrementabile solo dopo l'implementazione di uno o più requisiti.
- $[z]$: numero rappresentativo di una *patch*:
  - Parte da 0 e si resetta quando $[x]$ o $[y]$ vengono incrementati.
  - Incrementabile ad ogni modifica di uno o più requisiti e ad ogni cambio di configurazione del software.
- $[s]$: sigla che identifica uno stato del software:
  - `dev`: versione ancora in fase di sviluppo.
    - Componente non completa, soggetta a aggiunte, modifiche o eliminazioni.
    - Contiene potenzialmente errori che fanno fallire i test.
    - Non destinabile all'uso da parte dell'utente finale.
  - `rc`: versione candidata al rilascio:
    - Software con tutti o parte dei requisiti obbligatori richiesti soddisfatti.
    - Passa tutte le tipologie di test.
    - Utilizzabile dall'utente finale.
  - `stable`: pronta al rilascio pubblico:
    - Software completo che implementa tutti i requisiti obbligatori.
    - Passa tutte le tipologie di test ed è validato.
    - Collaudabile con il proponente e pubblicabile.

Vi possono essere varie tipologie di rilascio:

- ***Major release***: solo con approvazione di amministratore e responsabile.
- ***Minor release***: solo con superamento di tutti i test previsti e con approvazione di amministratore e verificatori.
- ***Patch***: rilasciabile autonomamente dal programmatore che la implementa.

3.2.3.2 Configurazione del workflow

Descrizione

Istanziazione di una repository per salvare, mantenere e versionare i vari file (e oggetti) prodotti dal gruppo. Normazione del suo utilizzo.

Repository

Si utilizza un *Version Control System*, fornito dal software *Git* attraverso la piattaforma *GitHub*, che permette la collaborazione asincrona attraverso diverse interfacce (web, *command line*, applicativo).

Workflow

Ogni membro lavorerà su una copia locale della repository remota, effettuando un *commit* al termine di ogni fase di lavoro.
Sono previste più *repository* per gestire meglio il flusso di lavoro.

- **Repository documentale**:
  - ***Docs***: contiene i documenti approvati, resi disponibili all'esterno.
  - ***docs-private***: spazio di lavoro del gruppo, contiene i seguenti branch:
    * `main`: branch principale su cui vengono fatti i rilasci.
    * `dev`: branch per lo sviluppo di nuovi documenti e funzionalità.
    * `feature-*`: branch indipendente per lo sviluppo o la risoluzione di una *issue*:
      · "*": identificativo della *issue*.
      · Da chiudere una volta implementata e/o risolta la *issue*.
- **Repository del codice**:
  - ***PoC***: dedicata a contenere le parti sviluppate durante la fase di progettazione della Technology Baseline.

Formato dei file

- **Configurazione**: modificabili solo dall'amministratore.
  - **.gitignore**: regole per evitare il caricamento in repository dei file non necessari.
  - **File senza formati**: contengono configurazioni aggiuntive per Github.
- **Documentazione**: contiene i file sorgente e i compilati:
  - **.tex:** contiene il codice sorgente LaTeX del documento.
  - **.pdf:** documento compilato o particolari immagini vettoriali.
  - **.png:** identifica una immagine.
  - **.md:** identifica un file scritto in markdown, usato per appunti.

Project Board

Si utilizza la *project board* messa a disposizione dalla suite *GitHub*, dove ogni membro potrà vedere lo stato del progetto:

- Visione d'insieme sull'avanzamento del progetto.
- Elenco delle *issue* da svolgere, in corso e completate.

Issue

Si utilizza l'*issue tracking system* di *GitHub* per la gestione del ticketing. Ogni issue deve contenere:

- **Descrizione**: feature da implementare o problema da risolvere.
- **Etichetta**: tipo di *issue*. Può essere:
  - **Verbali**.
  - **Casi d'uso**.
  - **NdP**.
  - **PdP**.
  - **PdQ**.
  - **AdR**.
- **Milestone**: identificativo temporale.
- **Proprietari**: quali membri del gruppo hanno in carico la *issue*.

Si definiscono 3 stati di avanzamento della *issue*:

- **To do**: appena aperta, ancora da prendere in carico.
- **In progress**: presa in carico da uno o più membri del gruppo.
- **Closed**: soggetta a verifica e chiusa.

Strumenti

- ***GitHub***:
  - **GitHub desktop**: applicativo ufficiale disponibile sui principali sistemi operativi.
  - **Git CLI**: applicativo da terminale, disponibile per tutte le piattaforme, con tutti i comandi utili per Git.

3.2.4 Metriche

Nessuna in particolare.

3.2.5 Strumenti

Definiti nelle attività del processo.

3.3 Gestione dei cambiamenti

3.3.1 Descrizione

Organizzazione e attuazione di mezzi e procedure efficienti, efficaci e semplici per la gestione di problemi in qualunque processo essi si verifichino.

3.3.2 Scopo

Individuare, documentare, analizzare e risolvere le problematiche che ostacolano la buona riuscita del progetto. In particolare:

- Evitare che possano verificarsi nuovamente in futuro.
- Evitare propagazione degli errori.
- Individuare problemi ed errori più comuni e focalizzare maggiormente la verifica su di essi.

3.3.3 Attività

3.3.3.1 Istanziazione del processo

Descrizione

Definizione di un processo per la gestione e la risoluzione di un problema.

Ciclo risolutivo

1. **Individuazione**:
    - Inserimento del problema nel processo di gestione dei cambiamenti.
    - Notificare le figure interessate.
    - Identificare cause e possibili conseguenze.
2. **Inserimento e categorizzazione**:
    - Identificazione del problema: **PROB-[tipo]-[priorità]-[progressivo]**
        - **Tipo**:
            * S: problemi di sintassi e/o ortografici.
            * C: problemi sul contenuto dell'oggetto.
        - **Priorità**:
            * 1: alta, da trattare immediatamente.
            * 2: media, non urgente, ma comunque rilevante.
            * 3: bassa, poco urgente, poca propagazione potenziale di errori.
        - **Progressivo**: numero progressivo di identificazione.
3. Analisi origini e possibili tendenze del problema.
4. Valutazione a possibili soluzione del problema.
5. Risoluzione del problema.
6. Verifica eliminazione problema e sue eventuali tendenze.
7. Valutazione rinforzo sistema di verifica per evitare la ricomparsa del problema.

3.3.3.2 Risoluzione del problema

Descrizione

Procedure pratiche per la gestione dei problemi individuati.

Issue tracking system

Inserimento del problema nel sistema di ticketing integrato di *GitHub* (paragrafo 3.2.3.2).
Nel momento in cui avviene l'inserimento ci si rifà al processo sopra descritto, eventualmente annotando gli esiti delle fasi come commenti della *issue* aperta.
Si collegheranno i *commit* alla *issue*, rendendo ogni modifica risolutiva tracciabile.

3.3.4 Metriche di qualità

Nessuna in particolare.

3.3.5 Strumenti

- ***GitHub***: utilizzo dell'*Issue tracking system* integrato con la *Project board*.

3.4 Gestione della qualità

3.4.1 Descrizione

Istanziazione di controlli per la verifica della qualità su diversi fronti:

- Prodotto software.
- Documentazione.
- Processi, per garantire efficacia ed efficienza.

In ogni processo è prevista una sezione 'Metriche di qualità', dove vengono riportate le metriche di misurazione della qualità del processo e/o del prodotto.

3.4.2 Scopo

Assicurare che le componenti software, la documentazione e i loro vari processi soddisfino i requisiti e gli obiettivi precedentemente specificati.
Rendere misurabili e verificabili, attraverso metriche di qualità oggettive, i processi istanziati.
Garantire:

- Qualità del prodotto finale.
- Soddisfacimento requisiti concordati con il proponente.
- Registrazione dei risultati ottenuti.
- Riscontro di problematiche interne ed esterne.

3.4.3 Attività

3.4.3.1 Definizione obiettivi di qualità di prodotto

Descrizione

Attuazione dei processi di verifica e validazione delle componenti software.

- **Verifica**: processo atto a provare la correttezza e la qualità degli algoritmi previsti del sistema.
- **Validazione**: processo di controllo che valuta la conformità del prodotto agli usi previsti e rispetta i vincoli stabiliti.

Ciclo di verifica qualità

1. **Pianificazione**: delineare un piano per assicurare la qualità dei processi e dei compiti da svolgere. Il piano deve includere standard, metodologie, procedure e strumenti per perseguire la qualità.
2. **Esecuzione**: applicazione del piano stabilito.
3. **Accertamento**: controllo dei risultati raggiunti.
4. **Correzione**: adeguano strategie e metriche a risultati ottenuti (miglioramento).

3.4.3.2 Definizione obiettivi di qualità di processo

Descrizione

Attuazione di attività di controllo di qualità sui processi istanziati. Garantire:

- **Efficacia**: raggiungere i vincoli e i requisiti imposti dal committente.
- **Efficienza**: garantire costi ridotti mantenendo qualità di prodotto.

Ciclo di verifica qualità

Paragrafo 3.4.3.1.

3.4.3.3 Classificazione dei processi

Descrizione

Identificazione dei processi, in ambito qualità:

$$QP - [id]$$

dove:

- **QP**: qualità del processo.
- **[id]**: identificativo numerico incrementale del processo.

3.4.3.4 Classificazione delle caratteristiche del prodotto

Descrizione

Identificazione delle caratteristiche del prodotto, in ambito qualità:

$$QC - [id]$$

dove:

- **QC**: qualità delle caratteristiche del prodotto.
- **[id]**: identificativo numerico incrementale della caratteristica.

3.4.3.5 Classificazione delle metriche

Identificazione

Ad ogni metrica è associato un identificatore univoco. Il modello è il seguente:

$$MQ - [cat] - [id]$$

Dove:

- **MQ**: metrica che valuta la qualità.
- **[cat]** tipologia a cui appartiene la metrica, può assumere i valori:
  - **PS**: processi.
  - **PD**: prodotti.
  - **TS**: test.
- **[id]**: identificativo numerico incrementale della metrica.

Descrizione

Per ogni metrica si definiscono i seguenti campi:

- **Descrizione**: tipo di metrica e su cosa si basa.
- **Unità di misura**.
- **Fonte** (*facoltativa*): dove si può estrapolare la metrica.
- **Formula** (*facoltativa*): come calcolare la metrica.
- **Interpretazione risultati** (*facoltativa*): come interpretare univocamente la metrica.

3.4.4 Metriche di qualità

3.4.4.1 QP-1 Garanzia della qualità

MQ-PD-9 Metriche soddisfatte (MS)

- **Descrizione**: percentuale delle metriche soddisfatte rispetto alle metriche disponibili durante la misurazione.
- **Unità di misura**: %.
- **Formula**:

$$MS = \frac{\#metriche\_disponibili\_soddisfatte}{\#metriche\_disponibili\_totali} * 100$$

- **Interpretazione risultato**:
  - 0%: nessuna metrica disponibile ha dato risultato positivo.
  - 0% < x < 100%: alcune metriche disponibili hanno dato esito positivo.
  - 100%: tutte le metriche disponibili sono positive.

3.4.5 Strumenti

Nessuno in particolare.

3.5 Verifica

3.5.1 Descrizione

Individuare eventuali errori o difetti presenti nelle componenti software o nei documenti, in modo da correggerli e restituire un prodotto pronto a svolgere la validazione.

3.5.2 Scopo

Verificare che il prodotto sia conforme ai vincoli, completo e corretto. Definire:

- Procedure e criteri di accettazione.
- Attività di verifica.
- Test di verifica delle componenti software.
- Soglia per passaggio a validazione.

3.5.3 Attività

3.5.3.1 Analisi

Descrizione

Attività di controllo su oggetti statici (documentazione, codice) e dinamici (componenti software).

Analisi statica

Analisi eseguita sull'oggetto (documentazione e codice) senza eseguirlo. Prevedere metodi di lettura manuale o automatica, per individuare errori formali. Si utilizzano 2 tecniche principali:

- **Walktrough**: il team controlla nella totalità l'oggetto in cerca di difetti o errori, senza svolgere una ricerca specifica per un certo tipo di errore.
- **Inspection**: ricerca di errori specifici, analizzando le sole sezioni in cui si trovano solitamente gli stessi. Il verificatore prepara una lista di possibili errori (checklist) che vanno ricercati e corretti.

Analisi dinamica

Analisi effettuata eseguendo la componente software. Consiste principalmente nell'attività di testing.

3.5.3.2 Testing

Descrizione

Verificare che la componente soggetta al test svolga correttamente le attività e rispetti i vincoli per lei previsti. Sono inoltre utili per individuare possibili errori di funzionamento. Per ogni test vanno definiti i seguenti parametri:

- **Ambiente**: sistema (hardware e software) usato per eseguire il test.
- **Stato iniziale**: parametri del software al momento dell'esecuzione del test.
- **Input**: dati consegnati in input per l'esecuzione.

- **Output**: i risultati attesi in output rispetto allo specifico input.
- **Commenti addizionali**.

Il PdQ terrà memoria dei test, e dei relativi risultati, attraverso una tabella. Per ogni test si indica:

- **ID**: identificativo del test.
- **Descrizione**: quale caso specifico va a coprire il test.
- **Stato**:
  - **TBI**: To Be Implemented.
  - **AI**: Already Implemented.
- **Esito**:
  - **F**: Failed.
  - **S**: Success.

Test di unità

Test automatici effettuati sulle singole unità software, con input fornito e output atteso.
Devono essere eseguiti per primi, in quanto verificano l'integrità e la correttezza della singola unità, prima dell'integrazione con le altre.
Il modello di classificazione per questi test è il seguente:

$$TU - [id]$$

dove:

- **TU**: test di unità.
- **[id]**: identificatore numerico incrementale del test di unità.

Test di integrazione

Test eseguiti su un'insieme di unità che interagiscono tra loro, per controllare la corretta interazione tra le stesse.
Da svolgere in seguito ai test di unità.
Si eseguono espandendo sequenzialmente il sistema: si aggiungono le varie unità in modo sequenziale fino a raggiungere il sistema completo.
Il modello di classificazione per questi test è il seguente:

$$TI - [id]$$

dove:

- **TI**: test di integrazione.
- **[id]**: identificatore numerico incrementale del test d'integrazione.

Test di sistema

Test eseguiti sull'intero sistema.
Da eseguire una volta che sono state testate le singole unità e che il sistema è perfettamente integrato.
Il modello di classificazione per questi test è il seguente:

$$TS - [priorità] - [tipo] - [id]$$

dove:

- **TS**: test di sistema.
- **[priorità]**: che può assumere i valori:
  - **1**: Obbligatorio.
  - **2**: Desiderabile.
  - **3**: Utile o contrattabile.
- **[tipo]**:può assumere i valori:
  - **F**: Funzionale.
  - **Q**: Qualitativo.
  - **P**: Prestazionale.
  - **V**: Vincolo.
- **ID**: identificatore univoco del test, parte da 1.

Test di regressione

Test da eseguire a seguito di una modifica o dell'aggiunta di una nuova funzionalità.
Sono utili quindi a verificare che le unità funzionino correttamente anche dopo la modifica apportata.
Il modello di classificazione per questi test è il seguente:

$$TR - [id]$$

dove:

- **TR**: indica che è un test di regressione;
- **[id]**: identificatore numerico incrementale del test di regressione.

3.5.4 Metriche di qualità

3.5.4.1 QP-4 Verifica

MQ-TS-10 Statement coperti (SC)

- **Descrizione**: percentuale che indica il numero di statement eseguibili, posto un determinato input, rispetto a quelli totali.
- **Unità di misura**: %.
- **Formula**:

$$SC = \frac{\#statement\_eseguiti}{\#statement\_totali} * 100$$

- **Interpretazione risultato**:
  - 0%: copertura nulla.
  - 0% < x < 100%: copertura parziale.
  - 100%: copertura totale.

MQ-TS-11 Bug risolti (BR)

- **Descrizione**: percentuale che indica il numero di bug risolti, rispetto al numero totale di quelli scoperti.
- **Unità di misura**: %.
- **Formula**:

$$BR = \frac{\#bug\_risolti}{\#bug\_scoperti} * 100$$

- **Interpretazione risultato**:
  - 0%: nessun bug è stato risolto.
  - 0% < x < 100%: alcuni bug sono stati risolti.
  - 100%: tutti i bug sono stati risolti.

MQ-TS-12 Test superati (TS)

- **Descrizione**: percentuale che indica il numero di test superati, rispetto al totale dei test previsti.
- **Unità di misura**: %.
- **Formula**:

$$TS = \frac{\#test\_superati}{\#test\_totali} * 100$$

- **Interpretazione risultato**:
  - 0%: nessun test è stato superato.
  - 0% < x < 100%: alcuni test sono stati superati.
  - 100%: tutti i test sono stati superati.

MQ-TS-13 Test falliti (TF)

- **Descrizione**: percentuale che indica il numero di test falliti, rispetto al totale dei test previsti in un determinato periodo.
- **Unità di misura**: %.
- **Formula**:

$$TF = \frac{\#test\_falliti}{\#test\_totali\_previsti} * 100$$

- **Interpretazione risultato**:
  - 0%: nessun test è fallito.
  - 0% < x < 100%: alcuni test sono falliti.
  - 100%: tutti i test sono falliti.

3.5.5 Strumenti

- **Spell checker**: controllo ortografico integrato nell'ambiente di lavoro LaTeX.
- **Code checker**: come definito nel paragrafo 2.2.3.3.

3.6 Validazione

3.6.1 Descrizione

Verificare che il prodotto finale sia conforme alle aspettative del committente e del proponente attraverso un incontro con essi.
Da eseguire in seguito al processo di verifica.

3.6.2 Scopo

Garantire che il prodotto software finale rispetti le richieste e i vincoli dettati dal proponente. Deve:

- Soddisfare tutti i requisiti concordati con il proponente.
- Eseguire correttamente nel suo ambiente di utilizzo finale.

3.6.3 Attività

3.6.3.1 Testing con il committente

Descrizione

Esecuzione di una serie di test di accettazione in presenza del proponente. I test valuteranno:

- Soddisfacimento dei casi d'uso.
- Soddisfacimento dei requisiti obbligatori.
- Soddisfacimento di altri requisiti concordati con il committente.

I test vengono trattati come un qualsiasi altro tipo di test (paragrafo 3.5.3.2), senza tenere traccia del risultato.

Test di accettazione

Il modello di classificazione è il seguente:

$$TA - [priorità] - [tipo] - [id]$$

dove:

- **TA**: test di accettazione.
- **[priorità]**: priorità del requisito associato al test, può assumere i valori:
  - **1**: Obbligatorio.
  - **2**: Desiderabile.
  - **3**: Utile o contrattabile.
- **[tipo]**: tipologia del test, può assumere i valori:
  - **F**: Funzionale;
  - **Q**: Qualitativo;
  - **P**: Prestazionale;
  - **V**: Vincolo.
- **[id]**: identificatore numerico incrementale del test di accettazione.

3.6.4 Metriche di qualità

Nessuna in particolare.

3.6.5 Strumenti

Gli stessi per il processo di verifica (paragrafo 3.5.5).

4 Processi organizzativi

4.1 Gestione dei Processi

4.1.1 Descrizione

Insieme di attività a supporto dell'istanziazione e dell'attuazione degli altri processi.
L'insieme completo delle attività viene elencato nel PdP. Si trattano principalmente:

- **Analisi dei rischi**: classificazione e come mitigarli.
- **Assegnazione ruoli e compiti**.
- **Istanziazione dei processi**: realizzare il modello di sviluppo adottato.
- **Pianificazione**: stima dei costi in termini di tempo e risorse.
- **Revisione e valutazione periodica**.

4.1.2 Scopo

- Adottare un modello organizzativo per l'individuazione dei rischi e la loro mitigazione.
- Definire un modello di sviluppo da adottare.
- Pianificare il lavoro rispettando le scadenze temporali prefissate, equilibrando il lavoro tra i vari ruoli.
- Determinare i mezzi di comunicazione interni ed esterni al gruppo.
- Calcolare un preventivo in termini di ore e costi suddiviso per ruoli, controllando i risultati a consuntivo (di periodo).
- Determinare un bilancio finale sulle spese.

4.1.3 Attività

4.1.3.1 Pianificazione del lavoro

Descrizione

Preparazione dei piani di lavoro per l'esecuzione dei processi.
Deve essere garantita:

- Suddivisione dei ruoli, tenendo conto delle regole di rotazione.
- Equa assegnazione oraria dei ruoli ai membri (tutti i membri del gruppo devono svolgere ogni ruolo per un tempo consono).

4.1.3.2 Definizione ruoli di progetto

Descrizione

Definizione delle figure necessarie alla realizzazione del progetto. Per ogni figura si evidenziano:

- Compiti da svolgere.
- Responsabilità.

Responsabile

Funge da intermediario verso le figure esterne al gruppo, quali committente e proponente; è il responsabile ultimo dei risultati del progetto.
In particolare si occupa di:

- Pianificazione del lavoro e delle conseguenti scadenze.
- Approvazione ed emissione dei documenti.
- Coordinamento di attività, risorse e comunicazioni per l'intero gruppo.

Amministratore

Gestisce l'ambiente di sviluppo e svolge una funzione di supporto.
In particolare si occupa di:

- Garantire efficacia ed efficienza.
- Affiancare il responsabile nella redazione del PdP.
- Attuare le procedure per la gestione della qualità.
- Gestire le infrastrutture di supporto.
- Controllare le versioni e le configurazioni del prodotto.
- Risolvere eventuali problemi nella gestione dei processi.
- Gestire la documentazione.

Analista

È responsabile di tutte le attività di analisi svolte durante l'AdR, la cui fine comporta la fine di tutti i suoi incarichi.
In particolare si occupa di:

- Studiare il dominio applicativo del progetto.
- Definire i requisiti del progetto e le sue complessità.
- Redigere i documenti di AdR.

Progettista

Si occupa delle attività di progettazione architetturale e di dettaglio. Segue lo sviluppo ma non la manutenzione del prodotto.
In particolare deve:

- Prendere decisioni tecniche favorendo efficacia e efficienza.
- Definire l'architettura del prodotto da sviluppare in base alle tecnologie scelte e ai requisiti individuati nell'AdR.
- Redigere la TB e la parte pratica del PdQ.

Programmatore

Si occupa della parte di codifica in base alle specifiche fornite dal progettista, operando con ottica di manutenibilità del codice.
In particolare deve:

- Implementare, secondo la TB, le specifiche fornite dal progettista.
- Creare e gestire componenti di supporto per la verifica e la validazione del codice.

Verificatore

È il responsabile di tutte le attività di verifica dei documenti e del codice.
In particolare deve:

- Controllare i prodotti in fase di revisione, utilizzando le tecniche e gli strumenti definiti nelle NdP.
- Indicare eventuali errori riscontrati nell'oggetto in esame.
- Segnalare eventuali errori al membro autore dell'oggetto interessato.

4.1.3.3 Gestione delle comunicazioni

Descrizione

Definizione degli strumenti di comunicazione per gestire il lavoro, organizzare le riunioni e gestire agilmente problematiche dalla risoluzione facile.

Comunicazioni interne

Le comunicazioni interne al gruppo avvengono utilizzando principalmente il software di messaggistica *Telegram*, per le sue funzionalità e comodità.
Gli argomenti di comunicazione devono riguardare principalmente:

- **Lavoro**: discussione di task da svolgere e in corso.
- **Riunioni**: organizzazione degli incontri interni.
- **Richieste**: notificare dubbi, chiarimenti e richieste da rivolgere al proponente nel primo incontro utile.

Comunicazioni esterne

Le comunicazioni esterne, con committente e proponente, vengono trattate esclusivamente dal responsabile. Avvengono principalmente tramite due canali:

- **Posta elettronica**: account *GMail*: *seven.solutions.unipd@gmail.com*.
  Lo strumento è utilizzato esclusivamente dal responsabile, ma accessibile a tutti i membri del gruppo.
- **Gruppo *Telegram***: gruppo composto dai membri del gruppo (*Seven Solutions*) e dai proponenti (*Imola Informatica*) che hanno direttamente suggerito questo canale (*VI 26-11-2021*) che permette una comunicazione più rapida e comoda.

4.1.3.4 Gestione delle riunioni

Descrizione

Definizione delle regole per l'organizzazione delle riunioni, sotto il controllo del responsabile. Le riunioni possono richiedere un confronto diretto tra i membri del gruppo, e se necessario con il proponente. Per ogni riunione il responsabile nomina un segretario che avrà il compito di tener traccia della discussione effettuata per poter poi redigere il verbale della riunione.

Riunioni interne

Riguardano solo i membri del gruppo. Il responsabile dovrà:

- Organizzare la riunione assicurandosi la disponibilità dei membri coinvolti.

- Redigere un programma che verrà comunicato agli interessati sugli argomenti che verranno trattati durante la riunione.
- Creare la stanza virtuale per la riunione.

Le riunioni avverranno nella piattaforma *Zoom* dove il responsabile renderà disponibile il link per accedervi 5 minuti prima dell'orario d'inizio.

Riunioni esterne

Riguardano i membri del gruppo, uno o più rappresentanti dell'azienda proponente oppure il committente.
Le riunioni con il proponente avverranno tramite *Microsoft Teams* o *Zoom* previo accordo con lo stesso. Le riunioni con il committente avverranno tramite *Zoom*.

Verbali degli Incontri

Ad ogni incontro, sia esso interno od esterno, il segretario incaricato deve redigere il verbale. I verbali devono avere un nome univoco e che ne permetta una catalogazione temporale all'interno della repository. A questo scopo si è deciso di usare la seguente forma:

[data]_v[tipo incontro]

- **[data]**: data in cui è avvenuto l'incontro nel formato AAAA_MM_GG.
- **v**: verbale.
- **[tipo incontro]**:tipo di incontro:
  - **i**: incontro interno.
  - **e**: incontro esterno.

4.1.3.5 Gestione degli strumenti di coordinamento

Descrizione

I membri del gruppo devono:

- Poter controllare lo stato in cui si trovano tutti i task.
- Poter essere a conoscenza dei compiti assegnati e corrispettiva scadenza per il completamento, in modo da poter gestire il proprio carico di lavoro.

Il responsabile deve:

- Poter assegnare nuovi compiti ai membri del gruppo.
- Poter controllare lo stato di avanzamento dei task, a verifica che tutto proceda come da programma.

Per l'adempimento di tali necessita, si utilizza l'*issue tracking system* fornito da *GitHub* assieme al *repository* per il progetto. Le modalità di utilizzo sono descritte nel paragrafo 3.2.3.2

4.1.3.6 Gestione dei Rischi

Descrizione

Rilevati dei rischi durante lo svolgimento del progetto. Conseguente segnalazione, classificazione e documentazione. Gestione risolutiva e mitigazione come descritto nel paragrafo 3.3.3.1.

Identificazione

Per agevolare l'identificazione di ogni rischio e il suo riferimento nei documenti, è prevista la seguente codifica:

RIS-[tipo][priorità][probabilità]-[id]

dove:

- **RIS**: rischio.
- **[tipo]**: natura del rischio:
  - **O**: organizzativo, gestione del lavoro, risorse e costi.
  - **P**: personale, mancanza di conoscenze e/o capacità.
  - **R**: requisito, modifica durante lo sviluppo del progetto.
  - **S**: strumentale, mezzi software impiegati per lo sviluppo del progetto.
  - **T**: tecnologico, tecnologie *hardware* e *software* per lo sviluppo del progetto.
- **[priorità]**:
  - **1**: inaccettabile, da mitigare al più presto.
  - **2**: tollerabile, anche se richiederà mitigazione.
  - **3**: accettabile, mitigazione facoltativa.
- **[probabilità]**:
  - **A**: alta.
  - **M**: media.
  - **B**: bassa.
- **[id]**: identificatore numerico incrementale del rischio in base al tipo.

4.1.4 Metriche di qualità

4.1.4.1 QP-1 Gestione processi

Gestione delle risorse.

MQ-PS-13 Costo pianificato sostenuto (CPS)

- **Descrizione**: indica il costo pianificato per le realizzazione del progetto in un determinato momento.
- **Unità di misura**: euro.

MQ-PS-14 Costo attuale sostenuto (CAS)

- **Descrizione**: indica il costo attualmente sostenuto per le realizzazione di tutte le attività svolte fino a quel momento.
- **Unità di misura**: euro.

MQ-PS-15 Costo preventivato sostenuto (CVS)

- **Descrizione**: indica il costo pianificato per le realizzazione di tutte le attività svolte fino a quel momento.
- **Unità di misura**: euro.

MQ-PS-16 Varazione programmazione (VP)

- **Descrizione**: indica il rapporto tra lo stato di avanzamento attuale e quello programmato, determina se si è in anticipo, in ritardo o in linea con la programmazione (efficacia).
- **Unità di misura**: %.
- **Formula**:

$$VP = \frac{CVS - CPS}{CVS} * 100$$

- **Interpretazione risultato**:
  - < 0%: progetto indietro rispetto alla schedulazione.
  - 0%: progetto in linea con la schedulazione.
  - > 0%: progetto avanti rispetto alla schedulazione.

MQ-PS-17 Varazione costi (VC)

- **Descrizione**: indica il rapporto tra i costi effettivamente sostenuti e quelli programmati (efficienza).
- **Unità di misura**: %.
- **Formula**:

$$VC = \frac{CVS - CAS}{CVS} * 100$$

- **Interpretazione risultato**:
  - < 0%: costo minore rispetto alla schedulazione.
  - 0%: costo in linea con la schedulazione.
  - > 0%: costo maggiore rispetto alla schedulazione.

4.1.4.2 QP-2 Gestione rischi

MQ-PS-18 Rischi non previsti (RNP)

- **Descrizione**: metrica incrementale che aumenta ogni volta che si incontra un rischio non preventivato durante un periodo del progetto. Il numero si azzera ad ogni periodo.
- **Unità di misura**: numero intero, valore iniziale: 0.
- **Formula**:

$$RNP = RNP + 1$$

- **Interpretazione risultato**:
  - 0: non sono stati trovati rischi durante il periodo in oggetto.
  - >0: sono stati trovati rischi durante il periodo in oggetto.

4.1.5 Strumenti

- ***GitHub***:
  - Versionamento in rete del progetto.
  - Archivio e condivisione dei dati del progetto.
  - Monitorizzazione dello stato dei task assegnati ai membri del gruppo.
- ***Telegram***: strumento di comunicazione tra i membri.
  - Scambiarsi comunicazioni di vario genere sul progetto.
  - Organizzare riunioni interne ed esterne.
- ***Zoom***: riunioni tra i membri del gruppo, proponente e commitente.
- ***Microsoft teams***: riunioni con il proponente.
- ***GMail***: account di posta elettronica usata per le comunicazioni esterne.

4.2 Formazione

4.2.1 Descrizione

Documentarsi su linguaggi di programmazione, ambienti di sviluppo e strumenti di sviluppo per la realizzazione del progetto.
Le fonti vengono aggiunte man mano che le tecnologie vengono scelte e/o suggerite dal proponente.

4.2.2 Scopo

Ottenere una formazione di base individuale consona alla realizzazione del progetto perseguendo efficacia ed efficienza.
In particolare, si vuole:

- Avere le basi tecnologiche per la redazione dei documenti.
- Avere sufficienti conoscenze sulle librerie impiegate.
- Avere buone conoscenze su costrutti e sintassi dei linguaggi di programmazione scelti.

4.2.3 Attività

4.2.3.1 Gestione della formazione

Descrizione

Reperimento delle fonti a cui attingere per la formazione.

Fonti

- **Gestione di progetto**:
  - ***GitHub***
    - Quickstart
    - Workflow
- **Redazione dei documenti**:
  - LaTeX
- **Linguaggi di programmazione**:
  - Kotlin
  - Python
- **Framework**:
  - Django
- **Librerie**:
  - ChatKit
  - Chatterbot

4.2.4 Metriche di qualità

Nessuna in particolare.

4.2.5 Strumenti

Nessuno in particolare.