⟨CodeHex16⟩

unipd.codehex16@gmail.com

Norme di Progetto

Data	12/11/2024
Versione	1.0.0

Sommario

Norme di progetto

Ruoli

Matteo Bazzan	
Luca Ribon	Redattore, Verificatore
Francesco Fragonas	Redattore
Gabriele Magnelli	Redattore, Verificatore
Filippo Sabbadin	Redattore, Verificatore
Luca Rossi	Verificatore
Yi Hao Zhuo	Verificatore

Registro delle Versioni

Versione	Data	Autore	Cambiamenti	Verificatore
1.0.0	06/03/2025	Francesco Fragonas	Sistemazione riferimenti	Gabriele Magnelli
0.6.1	04/03/2025	Francesco Fragonas	Sistemazione indice	Gabriele Magnelli
0.6.0	26/02/2025	Gabriele Magnelli	Conclusa sezione verifica	Luca Ribon
0.5.0	12/11/2024	Luca Ribon	Correzione del contenuto, formattazione e stesura sezione Sviluppo	Luca Rossi
0.4.0	08/01/2025	Gabriele Magnelli	Stesura nuove sezioni, migliorie e correzioni varie	Yi Hao Zhuo
0.3.0	12/12/2024	Gabriele Magnelli	Redazione sezioni Processi di supporto e Processi organizzativi	Yi Hao Zhuo
0.2.0	30/11/2024	Francesco Fragonas	Redazione Processi di accordo	Filippo Sabbadin
0.1.1	30/11/2024	Francesco Fragonas	Revisione Introduzione	Filippo Sabbadin

Versione	Data	Autore	Cambiamenti	Verificatore
0.1.0	12/11/2024	Filippo Sabbadin	Prima stesura	Gabriele Magnelli

Indice

1. Introduzione 1
1.1. Scopo del documento 1
1.2. Scopo del prodotto 1
1.3. Glossario 1
1.4. Riferimenti 2
1.4.1. Riferimenti normativi 2
1.4.2. Riferimenti informativi 2

2. Processi primari 4
2.1. Processo di fornitura 4
2.1.1. Scopo e descrizione 4
2.1.2. Rapporti con il proponente 4
2.1.3. Documentazione prodotta 5
2.1.3.1. Valutazione dei capitolati 5
2.1.3.2. Preventivo dei costi 5
2.1.3.3. Analisi dei requisiti 5
2.1.3.4. Piano di progetto 5
2.1.3.5. Piano di qualifica 6
2.1.4. Strumenti utilizzati 6
2.2. Sviluppo 6
2.2.1. Scopo e descrizione 6
2.2.1.1. Analisi dei requisiti 7
2.2.1.2. Progettazione 8
2.2.1.3. Codifica e Testing 8

3. Processi di supporto 9
3.1. Documentazione 9
3.1.1. Scopo e descrizione 9
3.1.2. Composizione tipografica 9
3.1.3. Struttura dei documenti 9
3.1.3.1. Intestazione 9
3.1.3.2. Registro delle modifiche 10
3.1.3.3. Indice 10
3.1.3.4. Corpo del documento 10
3.1.4. Documenti del progetto 10
3.1.5. Elenchi puntati 11

3.1.6. Immagini use case 11
3.1.7. Formato delle date 11
3.1.8. Strumenti 11
3.2. Gestione della configurazione 11
3.2.1. Scopo e descrizione 11
3.2.2. Versionamento 11
3.2.3. Repository 12
3.2.3.1. Struttura della repository documentazione 12
3.2.4. Sincronizzazione 12
3.2.4.1. Branch 12
3.2.4.2. Pull request 13
3.2.5. Strumenti usati 14
3.2.6. Verifica 14
3.2.6.1. Scopo e descrizione 14
3.2.6.2. Analisi statica 14
3.2.6.3. Analisi dinamica 14
3.2.6.3.1. Test 15
3.2.7. Validazione 15
3.2.8. Gestione qualità 16
3.2.8.1. Scopo e descrizione 16

4. Processi organizzativi 17
4.1. Gestione dei processi 17
4.1.1. Scopo e descrizione 17
4.1.2. Ruoli 17
4.1.3. Ticketing (Issue Tracking System) 18
4.2. Coordinamento 19
4.2.1. Comunicazioni e Riunioni 19
4.2.2. Verbali 20
4.3. Miglioramento 20
4.3.1. Scopo e descrizione 20

1. Introduzione

1.1. Scopo del documento

Questo documento ha lo scopo di delineare le principali fasi di sviluppo, i ruoli e le responsabilità dei membri del team CodeHex16*. Al suo interno viene fornita una guida completa per tutte le Practice* adottate dal gruppo e per il Way of Working*, garantendo un approccio strutturato e organizzato alle attività collaborative.

Il documento non si limita a fornire una panoramica iniziale ma si propone come un riferimento dinamico, soggetto a revisioni e aggiornamenti continui. Tale approccio incrementale assicura che il contenuto resti sempre aggiornato rispetto alle esigenze del progetto e alle best practices emergenti, consentendo al gruppo di adattarsi rapidamente a nuovi requisiti o cambiamenti contestuali.

1.2. Scopo del prodotto

Il progetto prevede lo sviluppo di un Chatbot* avanzato, basato su modelli linguistici LLM* (Large Language Models), pensato per migliorare la comunicazione tra aziende fornitrici e i loro clienti. Questo assistente virtuale* permetterà agli utenti di ottenere rapidamente e in modo intuitivo informazioni dettagliate su prodotti o servizi offerti, eliminando la necessità di contattare direttamente l'azienda.

Il sistema includerà anche un'interfaccia dedicata per le aziende fornitrici, offrendo strumenti per gestire i clienti e i documenti di riferimento che contengono le informazioni necessarie. Questi documenti saranno utilizzati dal modello linguistico per generare risposte personalizzate e accurate, garantendo un'esperienza utente ottimale. L'intero sistema sarà accessibile tramite una Webapp*, assicurando una gestione efficiente e una fruizione semplice per tutti gli utenti coinvolti.

1.3. Glossario

Per agevolare la comprensione del presente documento, è stato predisposto un glossario che spiega il significato dei termini specifici utilizzati nel contesto del progetto. Per facilitare la comprensione, questi termini avranno il seguente stile: Esempio*

Le definizioni sono disponibili nel documento Glossario.pdf e possono essere consultate anche tramite la seguente pagina web: `https://codehex16.github.io/glossario`

1.4. Riferimenti

1.4.1. Riferimenti normativi

- Capitolato C7 - Assistente Virtuale Ergon: `https://www.math.unipd.it/~tullio/IS-1/2024/Progetto/C7.pdf` *(ultima consultazione 03-03-2025)*;

1.4.2. Riferimenti informativi

- Sito del gruppo CodeHex16: `https://codehex16.github.io/` *(ultima consultazione 06-03-2025)*;
- Repository della documentazione del progetto: `https://github.com/CodeHex16/documentazione` *(ultima consultazione 06-03-2025)*;
- Valutazione capitolati: `https://codehex16.github.io/docs/1%20-%20candidatura/Valutazione-Capitolati.pdf` *(versione 0.3.0)*;
- Preventivo costi e impegni: `https://codehex16.github.io/docs/1%20-%20candidatura/Preventivo-Costi-e-Impegni.pdf` *(versione 0.2.0)*;
- Analisi dei requisiti: `https://codehex16.github.io/docs/2%20-%20RTB/Analisi-dei-Requisiti.pdf` *(versione 0.9.0)*;
- Piano di progetto: `https://codehex16.github.io/docs/2%20-%20RTB/Piano-di-Progetto.pdf` *(versione 0.7.0)*;
- Piano di Qualifica: `https://codehex16.github.io/docs/2%20-%20RTB/Piano-di-Qualifica.pdf` *(versione 1.0.0)*;
- Standard ISO/IEC 12207:1995: `https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf` *(ultima consultazione 06-03-2025)*;
- Glossario:

- Documento: `https://codehex16.github.io/docs/glossario/glossario.pdf` *(versione 1.0.0)*;
- Pagina web: `https://codehex16.github.io/glossario.html` *(ultima consultazione 06-03-2025)*;

2. Processi primari

2.1. Processo di fornitura

Il processo di fornitura è strutturato in conformità agli esiti previsti dalla clausola 5.2 dello Standard* ISO/IEC 12207:1997. Tale processo include la definizione di requisiti concordati, l'analisi dei rischi associati, e la pianificazione di tempi e costi.

2.1.1. Scopo e descrizione

Il processo di fornitura è finalizzato a garantire la realizzazione di un prodotto o servizio che soddisfi i requisiti concordati tra Proponente* e Committente*. L'accordo tra le parti deve definire in modo chiaro i requisiti, le tempistiche e i costi da rispettare. Prima di stipulare tale accordo, il Fornitore* avrà condotto un'analisi dettagliata del progetto proposto, identificando i rischi correlati e stabilendo le linee guida necessarie per gestirli efficacemente.

2.1.2. Rapporti con il proponente

Il gruppo CodeHex16 manterrà un dialogo attivo e regolare con il Proponente per tutta la durata del progetto didattico, con l'obiettivo di raccogliere il maggior numero possibile di Feedback* sulla correttezza e qualità del lavoro svolto. La comunicazione si articolerà in due modalità principali:

1 - Scritta (asincrona*) utilizzata per comunicazioni di breve durata, condivisione di verbali e materiale informativo e attività di coordinamento;

2 - Incontri online (sincrona*) utilizzati per chiarimenti sul capitolato, approfondimenti relativi ai casi d'uso e requisiti e feedback sul lavoro svolto;

Il formato testuale è chiaro, ma ci sono metodi di comunicazione che hanno maggiore fluidità e precisione:

I Meeting* saranno organizzati con cadenza variabile e fissati tramite e-mail in base alle necessità riscontrate durante lo sviluppo del progetto. Tutti i dettagli discussi durante questi incontri saranno documentati in verbali, con particolare attenzione alle decisioni prese. I verbali saranno disponibili al seguente link: `Verbali esterni - https://github.com/CodeHex16/documentazione/tree/main/2%20-%20RTB/verbali/esterni`

2.1.3. Documentazione prodotta

In questa sezione viene illustrata la documentazione prodotta dal gruppo nel processo di fornitura, che sarà messa a disposizione del Proponente, Ergon Informatica, e dei Committenti, i professori Tullio Vardanega e Riccardo Cardin.

2.1.3.1. Valutazione dei capitolati

Nel documento `Valutazione Capitolati`, il gruppo ha analizzato tutte le proposte di capitolato, fornendo per ciascuna una breve descrizione, una panoramica dello stack tecnologico previsto e una valutazione finale. La scelta del capitolato è stata effettuata considerando diversi criteri, tra cui l'interesse dei membri del gruppo per il progetto, la sua rilevanza nel contesto lavorativo e la fattibilità complessiva.

2.1.3.2. Preventivo dei costi

Nel documento `Preventivo Costi e Impegni` è stata stabilita una data di consegna stimata del progetto, definita in accordo con tutti i membri del gruppo. La pianificazione tiene conto degli impegni personali di ciascun membro e prevede una stima delle ore settimanali da dedicare al progetto. Inoltre, dopo aver definito tutti i ruoli, è stata elaborata una tabella con la previsione delle ore che ogni membro deve svolgere per ciascun ruolo, garantendo una rotazione prestabilita per bilanciare equamente il carico di lavoro.

Probabilmente non va messa qui:

2.1.3.3. Analisi dei requisiti

Nel documento `Analisi dei Requisiti`, il gruppo ha definito tutti gli Use Case* e i requisiti, frutto di un'attenta analisi del capitolato e della comprensione dell'utilizzo finale del progetto. Il contenuto di questo documento è stato arricchito grazie alle riflessioni svolte con il referente dell'azienda proponente durante gli incontri svolti.

2.1.3.4. Piano di progetto

Nel documento `Piano di Progetto` è stato pianificato l'avanzamento del progetto suddiviso nei 3 periodi chiave (Candidatura, RTB e PB) con una particolare attenzione agli Sprint* settimanali effettuati. Per ognuno è stato descritto il lavoro svolto, il rendiconto delle ore e dei costi in base ai ruoli assegnati.
Inoltre è presente una retrospettiva per ogni periodo volto al raggiungimento di una milestone.

2.1.3.5. Piano di qualifica

Nel documento `Piano di Qualifica` vengono dichiarati gli obiettivi di qualità che il gruppo si prefigge di raggiungere durante lo sviluppo del progetto. Vengono inoltre descritte le metodologie di verifica e validazione adottate per garantire la qualità del prodotto finale, indicando anche le metriche di qualità utilizzate per misurare il grado di soddisfacimento degli obiettivi prefissati.

2.1.4. Strumenti utilizzati

Per lo svolgimento del progetto abbiamo utilizzato i seguenti strumenti:
- **Telegram*** per la comunicazione all'interno del gruppo;
- **Discord*** per gli incontri interni;
- **Zoom** per gli incontri esterni con il referente dell'azienda Ergon Informatica;
- **GitHub** per organizzare tutti i documenti e file sorgente del progetto tramite un repository;
- **GitHub Issue** per assegnare task* ad ogni membro avendo un rendiconto preciso dei ruoli e delle ore svolte per ogni sprint, con l'assegnazione di label e milestone specifiche;
- **GitHub Project** per visualizzare in modo più ordinato le issue e i loro dettagli;
- **Google Fogli** per organizzare incontri con la compilazione di un calendario settimanale e per fissare le ore svolte avendo una visione generale dell'andamento del progetto;
- **Typst** per la stesura di tutti i documenti e verbali;
- **Canva** per la realizzazione delle presentazioni per i Diari di Bordo settimanali;
- **Notion*** per organizzare appunti e documenti non ufficiali;

2.2. Sviluppo

Il processo di sviluppo è finalizzato alla realizzazione del prodotto software richiesto dal Proponente, seguendo le specifiche definite nel capitolato d'appalto. Questo processo include tutte le attività necessarie per la creazione del prodotto, dalla progettazione iniziale all'integrazione finale, garantendo che il software soddisfi i requisiti concordati e sia conforme alle aspettative del cliente.

2.2.1. Scopo e descrizione

Questo processo prevede le seguenti attività principali:
- Analisi dei requisiti;

- Progettazione;
- Codifica;
- Testing;

L'output atteso dal processo è un prodotto software funzionante, che soddisfi i requisiti concordati e che sia ampiamente testato. Perché rispetti i requisiti concordati la fase di codifica dovrà seguire le linee guida fissate durante l'analisi e di conseguenza durante la progettazione.

2.2.1.1. Analisi dei requisiti

Il processo di sviluppo inizia con l'analisi dei requisiti, durante la quale vengono identificati e definiti i requisiti del sistema. Il prodotto di questa attività è contenuto nel documento `Analisi dei Requisiti`. I requisiti descrivono:
- funzionalità richieste;
- funzionalità di supporto;
- funzionalità di sicurezza;
- funzionalità di interfaccia;

I **casi d'uso** descrivono le interazioni tra il prodotto e gli attori coinvolti, ed aiutano ad individuare ulteriori requisiti. Gli attori individuati sono:
- Amministratore;
- Cliente;
- Fornitore;
- Utente non registrato;

Ogni use case ha la seguente struttura:
- Diagramma del caso d'uso;
- Attori principali;
- Attori secondari se presenti;
- Descrizione;
- Precondizioni;
- Postcondizioni;
- Scenario principale;
- Generalizzazioni, estensioni e inclusioni, se presenti;

In questo modo si possono definire in modo dettagliato i requisiti funzionali, di qualità e di vincolo.

2.2.1.2. Progettazione

La progettazione del sistema è l'attività di definizione dell'architettura del software dal punto di vista logico; in questa fase si decide come soddisfare i requisiti identificati durante l'analisi.

In particolare vanno definiti i componenti software e le loro interazioni, prestando attenzione a mantenerli separati e indipendenti per garantire una maggiore manutenibilità e scalabilità del sistema; in questo passaggio è importante anche definire le unità architetturali.
Inoltre vanno definite le responsabilità che verranno applicate in fase di codifica assicurandosi di mantenere un livello di efficienza* e efficacia* il più alto possibile. L'approccio utilizzato in questa attività sarà sia top-down*, per scomporre il problema in sotto-problemi, sia bottom-up*, per ragionare sui singoli sotto-problemi e integrarli in una soluzione complessiva.

Al termine di questa attività ci si aspetta di avere un'architettura ben definita che preveda:
- Backend*:
  - API* per l'interazione con l'LLM;
  - API per l'interazione con il database;
  - backend per la gestione delle interfacce utente e di configurazione;
- Frontend*:
  - Interfaccia utente per il cliente;
  - Interfaccia per il fornitore;
- Database;

2.2.1.3. Codifica e Testing

In questa attività i Programmatori traducono l'output della Progettazione in codice sorgente*, in modo da integrare ogni unità prevista dall'architettura. Inoltre ogni unità sarà documentata e testata per garantire che soddisfi i requisiti definiti in fase di analisi e progettazione. Nello specifico la documentazione dovrà prevedere la documentazione dedicata all'utente finale e quella dedicata al manutentore. Inoltre nella documentazione verranno integrati anche i dettagli realtivi al testing eseguito sulle singole unità.

3. Processi di supporto

3.1. Documentazione

3.1.1. Scopo e descrizione

Lo scopo del processo di documentazione è quello di tracciare e quindi rendere immediatamente consultabile ogni attività e processo relativi al progetto. Inoltre, saranno riportate le decisioni prese e le norme scelte dal gruppo CodeHex16 che verranno rispettate da tutti i suoi membri al fine di procedere al meglio nello sviluppo del progetto.

3.1.2. Composizione tipografica

Per la composizione tipografica dei documenti si è deciso di usare Typst* per i seguenti motivi:
- Semplicità degli strumenti utilizzati per la stesura;
- Sintassi semplice;
- Compilazione immediata;

Grazie a Typst si riesce facilmente a creare e mantenere un documento non lasciando il lavoro di controllo grafico al gruppo, infatti si possono scrivere i vari tipi di documenti partendo dai template che si possono trovare nella repository **`documentazione`** nella cartella *template*.

3.1.3. Struttura dei documenti

Ci sono diversi tipi di documenti e generalmente sono organizzati nelle seguenti sezioni:

3.1.3.1. Intestazione

La prima pagina è l'intestazione del documento ed è composta generalmente dalle seguenti informazioni:
- **Titolo**: Titolo del documento che è anche il nome del file;
- **E-mail**: E-mail del gruppo;
- **Logo**: Logo del gruppo;
- **Data**: La data in cui il gruppo si è incontrato e/o in cui il documento è stato redatto;
- **Versione**: La versione corrente del documento;

- **Sommario**: Una breve descrizione del contenuto del documento;
- **Ruolo**: I ruoli dei membri del gruppo in relazione al documento;

Per i verbali vengono aggiunte le seguenti informazioni:
- **Tipo**: Il verbale può essere di tipo interno o esterno;
- **Ora**: L'orario in cui è avvenuto l'incontro di gruppo;
- **Ordine del giorno**: L'elenco degli argomenti principali discussi durante l'incontro, che sostituisce il sommario;
- **Presenze**: Per ogni individuo presente durante l'incontro viene registrato il suo nome, cognome e il tempo per cui è stato presente durante l'incontro;

3.1.3.2. Registro delle modifiche

La seconda pagina riguarda il registro modifiche il cui contenuto è organizzato mediante una tabella in cui vengono riportate le seguente informazioni:
- **Versione**: Indica il numero della versione del documento, seguendo il formato definito nella sezione «Versionamento» 3.2.2;
- **Data**: Data della versione in cui è redatto il documento;
- **Autore**: Autore di quella versione del documento, cioè il membro del gruppo che ha apportato le modifiche;
- **Cambiamenti**: I cambiamenti pricipali di quella versione del documento;
- **Verificatore**: Membro del gruppo che ha verificato il documento per quella versione;

3.1.3.3. Indice

La terza pagina, e se necessario le seguenti, è riservata all'indice del documento che elenca le sezioni di cui è composto i documento.

3.1.3.4. Corpo del documento

Il corpo del documento è strutturato in capitoli, ciascuno dei quali può essere suddiviso in sottocapitoli, che a loro volta possono contenere ulteriori suddivisioni.

3.1.4. Documenti del progetto

Verranno prodotti i seguenti documenti:
- **Norme di Progetto***;
- **Piano di Progetto**;
- **Analisi dei Requisiti**;
- **Piano di Qualifica**;
- **Glossario**;

- **Verbali esterni**;
- **Verbali interni**;

3.1.5. Elenchi puntati

Ogni voce di un elenco puntato finisce con «;».

3.1.6. Immagini use case

Per produrre i diagrammi uml degli use case il team ha usato il seguente sito : `draw.io`. Successivamente vengono inseriti nei documenti opportuni come immagini.

3.1.7. Formato delle date

Per le date viene utilizzato lo standard internazionale **ISO 8601** nella forma YYYY-MM-DD in cui:
- **YYYY**:Indica l'anno con 4 cifre;
- **MM**: Indica il mese con 2 cifre;
- **DD**: Indica il giorno con 2 cifre;

3.1.8. Strumenti

- **VS Code**: Editor di testo usato per scrivere i documenti;
- **Typst**: Linguaggio per la stesura dei documenti;
- **Github**: Servizio di hosting per il repository;

3.2. Gestione della configurazione

3.2.1. Scopo e descrizione

Il processo di gestione di configurazione identifica le norme adottate dal gruppo per garantire la tracciabilità della documentazione e del codice prodotto durante tutto l'arco di vita del progetto. Lo scopo principale è quello di organizzare la procedura di modifica della documentazione e del codice prodotto e di rendere immediatamente consultabili le varie modifiche apportate e i loro autori.

3.2.2. Versionamento

In generale una versione ha una sintassi del tipo X.Y.Z in cui:
- **X**: E' il numero di versione principale che viene incrementato ogni qual volta il documento sia terminato e pronto per una revisione;

- **Y**: E' il numero di versione secondaria che viene incrementato ogni qual volta il documento sia stato modificato in modo significativo;
- **Z**: E' il numero di versione di correzione, incrementato ogni qual volta il documento subisca correzioni minori;

In particolare, per i verbali vengono considerati solo i numeri Y e Z per il fatto di essere documenti molto brevi, in cui:
- **Y**: E' il numero di versione principale che viene incrementato ogni qual volta il documento sia terminato o venga modificato con correzioni importanti;
- **Z**: E' il numero di versione secondaria che viene incrementato ogni qual volta il documento sia stato modificato con piccole correzioni;

3.2.3. Repository

3.2.3.1. Struttura della repository documentazione

Il contenuto pronto e convalidato del repository è presente nel branch* main in cui sono presenti tutti i PDF dei documenti prodotti. Nel branch main si possono trovare diverse cartelle che servono per organizzare e dividere i vari tipi di documenti e codice, in particolare:
- In **1 - candidatura** si trovano i documenti relativi alla candidatura per la gara d'appalto dei capitolati;
- In **2 - RTB** si trovano i documenti relativi alla fase di progetto **RTB (Requirement and Technology Baseline)**;
- In **3 - PB** si trovano i documenti relativi alla fase di progetto **PB (Product Baseline)**;
- In **diari-di-bordo** sono presenti tutti i diari di bordo prodotti;
- In **glossario** è presente il documento Glossario;
- In **template** sono presenti i template per i vari documenti.

3.2.4. Sincronizzazione

La sincronizzazione avviene tramite repository condivise su `github` in cui ogni attività da svolgere è tracciata da una issue con il/i membro/i assegnato/i a tale issue così da sapere sempre chi la segue, o l'ha seguita.

3.2.4.1. Branch

Per gestire al meglio le varie issue e la documentazione si è deciso di creare dei branch appositi per ogni documento importante come:
- **Norme di Progetto**;

- **Piano di Progetto**;
- **Analisi dei Requisiti**;
- **Piano di Qualifica**;
- **Glossario**;

Inoltre, se più membri del gruppo lavorano allo stesso documento allora viene creato un branch per ogni membro con nome il nome del documento/attività e di chi lo sta svolgendo. Quindi in generale la sintassi del nome di un branch è la seguente: **[Titolo]-[Nome-Membro]-[Sprint]**. Una volta finito il lavoro da parte di tutti i membri che operano su quel documento, questo viene verificato (tramite pull request*) e viene eseguito il merge sul branch main, mentre i branch ausiliari vengono eliminati appena si ha la sicurezza che questi non sono più necessari.
Altri branch degni di nota sono:
- **diario-di-bordo**: Utilizzato per inserire i diari di bordo nel repository;

3.2.4.2. Pull request

Per quanto riguarda le pull request si è deciso che per ogni documento redatto viene richiesta la verifica tramite pull request. A questo punto i verificatori a cui è stata assegnata la issue di verifica di quel documento, tramite questa pull request, eseguono la verifica e se è tutto corretto viene fatto il merge delle modifiche apportate nel main, altrimenti il verificatore può correggere direttamente il documento, oppure scrivere un commento con delle indicazioni per le correzioni da svolgere.

A questo punto i verificatori incaricati del sprint corrente (vengono create delle issue per la verifica assegnandole a chi di dovere), tramite questa pull request:
1. Esegue il controllo locale

```
# Clonare il repository
git checkout <pr-branch-name>
```

2. Eseguono la verifica e aggiungere proprio nome nel Registro dell Versioni
3. Se è tutto corretto viene effettuato il commit

```
git add .
git commit -m "Verifica <pr-branch-name>...."
git push origin <pr-branch-name>
```

4. Dopo di che viene fatto il merge delle modifiche apportate nel main sulla pagina Pull Request del Github
5. Altrimenti il verificatore può correggere direttamente il documento, oppure scrivere un commento con delle indicazioni per le correzioni da svolgere.

In generale, le pull request vengono effettuate quando vi è una modifica interna al repository.

3.2.5. Strumenti usati

- **Git**: Software usato per il controllo della versione dei documenti e del codice;
- **Github**: Servizio di hosting per progetti sotware usato dal gruppo per coordinarsi sulle operazioni di versionamento e usato come **Issue Tracking System**;

3.2.6. Verifica

3.2.6.1. Scopo e descrizione

La verifica è un processo affidato ai verificatori e inizia quando la fase iniziale di progettazione viene avviata. Questo processo ha come obiettivo quello di garantire un certo grado di qualità di tutto quello che viene prodotto dal gruppo (documentazione, codice sorgente, test, ecc..) e di conformità rispetto alle aspettative. Quindi tramite tecniche di analisi e test tale processo mira a stabilire se ciò che viene prodotto dal team soddisfa i requisiti richiesti. Il documento che rispecchia questo processo è il **Piano di Qualifica** e definisce gli obiettivi da raggiungere, i criteri di accettazione e i metodi che verranno usati per eseguire la verifica in modo completo ed efficiente.

3.2.6.2. Analisi statica

L'analisi statica è un tipo di verifica che non richiede l'esecuzione del prodotto e ha come obiettivo la revisione critica del codice e della documentazione al fine di garantire conformità ai vincoli, assenza di difetti e presenza delle funzionalità e proprietà richieste. Questo tipo di analisi adotta, tipicamente, due metodi di lettura: l'**inspection** e il **walkthrough**. L'inspection, preferibile al walkthrough per velocità ed efficienza, consente di individuare tempestivamente potenziali difetti, errori e problemi. Il walkthrough mette in collaborazione il verificatore con l'autore del prodotto preso in analisi e ne prevede una lettura a pettine.

3.2.6.3. Analisi dinamica

L'analisi dinamica è un tipo di verifica che richiede l'esecuzione del sistema e delle sue componenti così da individuare difetti, problemi ed errori nel funzionamento al fine di garantire un certo grado di qualità nel prodotto finale. Il gruppo, per garantire tutto ciò, utilizzerà un insieme di test ripetibili e automatizzati, anche se sarà necessario l'uso di test manuali. Tali test si suddividono nei seguenti tipi:

3.2.6.3.1. Test

Test di unità
I test di unità sono test effettuati su singole componenti autonome del sistema e tali test possono essere:
- **Test funzionali** che verificano che l'output prodotto sia uguale a quello atteso;
- **Test strutturali** che verificano tutti i possibili cammini del codice;

Test di sistema
I test di sistema sono impiegati per verificare il corretto funzionamento del sistema e, in particolare, che tutti i requisiti richiesti siano soddisfatti.

Test di integrazione
I test d'integrazione verificano la corretta integrazione tra le varie componenti del sistema che sono già state testate singolarmente tramite i test di unità. E' possibile seguire due approcci per i test d'integrazione:
- **Bottom up**: si testano per prime le componenti che hanno meno dipendenze e maggior valore interno, cioè quelle più nascoste all'utente;
- **Top down**: si testano per prime le componenti che hanno il maggior numero di dipendenze e maggiormente visibili da parte dell'utente così da avere disponibilità immediata di tali componenti;

Test di regressione
I test di regressione vengono impiegati per assicurare che la correzione o la modifica delle componenti non causi problemi al livello di sistema. Tali test sono necessari per garantire che le modifiche non compromettano le funzionalità già testate e funzionanti, evitando quindi la comparsa di regressioni nel sistema.

Test di accettazione
I test di accettazione devono essere eseguiti insieme al committente al fine di verificare che il prodotto finale rispetti tutti i requisiti richiesti.

3.2.7. Validazione

La validazione è la verifica ultima per garantire che il prodotto sia in linea con le aspettative e che rispetti i requisiti richiesti e per questo è una fase molto importante nello sviluppo del progetto. Questo processo segue il processo di verifica e si sofferma su alcuni aspetti quali:

- Il prodotto finale deve funzionare correttamente ed essere conforme con la logica di progettazione;
- Il prodotto deve soddisfare completamente i requisiti specificati;
- Il prodotto deve essere intuitivo e di facile compresione e utilizzo, cioè deve essere usabile;
- Il prodotto deve essere efficace nel soddisfare le necessità del cliente;

3.2.8. Gestione qualità

3.2.8.1. Scopo e descrizione

Il processo di gestione della qualità ha come obiettivo quello di garantire che il software, la documentazione e tutto ciò che il team produce sia conforme ai requisiti di qualità specificati e richiesti. Processi utili a garantire un certo grado di qualità sono sicuramente i processi di verifica e validazione. Gli obiettivi e gli standard di qualità richiesti e che devono essere soddisfatti sono indicati nel documento **Piano di Qualifica**.

4. Processi organizzativi

4.1. Gestione dei processi

4.1.1. Scopo e descrizione

Il processo di gestione ha lo scopo di identificare le attività e i compiti che ogni membro del gruppo dovrà eseguire per proseguire nel progetto.

4.1.2. Ruoli

I ruoli svolti, a rotazione, dai membri del gruppo sono:

- **Responsabile**: Ha il compito primario di coordinare i membri del gruppo, inoltre deve:
  - Determinare le attività da svolgere, assegnarle e verificarne l'avanzamento;
  - Gestire i rapporti tra i membri del gruppo e i soggetti esterni;
  - Redigere i verbali sia interni che esterni;

  Il responsabile è una figura che sarà presente durante tutto l'arco del progetto.
- **Amministratore**: Ha il compito primario di controllare e gestire l'ambiente di lavoro, inoltre deve:
  - Stabilire gli strumenti necessari da usare durante il progetto;
  - Gestire i processi e risolverne gli eventuali problemi;

  L'amministratore è una figura che sarà presente durante tutto l'arco del progetto.
- **Progettista**: Ha lo scopo principale di determinare le scelte realizzative del progetto, in particolare deve:
  - Trovare l'architettura adeguata per gestire il progetto;

  Il progettista è una figura che sarà, principalmente, presente durante la parte di sviluppo del progetto.
- **Analista**: Ha il compito principale di trovare i requisiti che il progetto dovrà soddisfare e riportarli nel documento **Analisi dei Requisiti**, quindi deve:
  - Studiare i bisogni dei committenti;
  - Studiare i requisiti definendone la complessità;
  - Scrivere il documento **Analisi dei Requisiti**

  L'analista è una figura che sarà presente principalmente durante la prima parte del progetto in cui verrà analizzato e compreso appieno il capitolato. Solo in casi straor-

dinari, cioè se il gruppo dovesse cambiare i requisiti del progetto, allora l'Analista dovrà essere interpellato per apportare delle modifiche ai requisiti in questione.
- **Programmatore**: Ha il compito primario di svolgere l'attività di codifica e sviluppare l'architettura individuata dal Progettista, in particolare deve:
  - Scrivere codice mantenibile che rispetti le **Norme di Progetto**;
  - Creare test per la verifica e validazione del codice;

  Il programmatore è una figura che sarà presente durante la parte di sviluppo del progetto.
- **Verificatore**: Ha il compito pricipale di controllare e validare la documentazione e il codice prodotto, in particolare deve:
  - Controllare ogni documento a lui assegnato, verificarlo e in caso correggerlo o notificare il redattore, specificare cosa non è corretto e richiederne la correzione;
  - Controllare che tutto ciò che viene prodotto rispetti le **Norme di Progetto**;

  Il Verificatore è una figura che sarà presente durante tutto il progetto.

Ogni membro, in un dato momento, può svolgere un solo ruolo alla volta, ma durante lo sprint può assumere più ruoli.

4.1.3. Ticketing (Issue Tracking System)

Il gruppo sfrutta l'ITS* offerto da Github per gestire le attività da svolgere, cioè le **Issue**. Creare le issue è molto semplice e veloce; quando viene individuata un'attività specifica da svolgere viene creata una issue e viene assegnata, in modo coerente, ad un membro del gruppo. Quando viene creata una issue, questa sarà composta da:
- **Titolo**: Il nome della issue che identifica l'attività da svolgere;
- **Assegnatario/i**: Il/I membro/i del gruppo a cui è affidata la issue;
- **Etichetta/e**: Identifica il tipo di issue, come ad esempio i documenti su cui lavorare o la macro categoria delle attività da svolgere;
- **Stato**: Avanzamento della issue, nello specifico:
  - **Todo**: La issue è stata creata ma non ancora svolta;
  - **In corso**: La issue è in fase di svolgimento;
  - **Completato**: La issue è stata completata e chiusa;
- **Priorità**: Identifica l'urgenza con cui svolgere la issue;
- **Peso**: Stima del carico di lavoro relativo alla issue;
- **Sprint**: Lo sprint in cui questa issue deve, idealmente, essere svolta;
- **Ruolo**: Identifica il ruolo associato allo svolgimento di tale issue;
- **Ore**: Il numero di ore impiegate per svolgere la issue;
- **Repository**: Repository a cui è associata la issue;

- **Milestone**: Identifica la milestone a cui è assegnata la issue;

Più in particolare quando viene individuato un compito da svolgere vengono eseguiti i seguenti passi:
1. Viene individuato il compito da svolgere e viene creata la issue relativa;
2. La issue viene assegnata ad uno o più membri del gruppo e vengono inserite le informazioni della issue scritte appena sopra.
3. La issue viene completata e chiusa, oppure se è richiesta la verifica del lavoro svolto allora il verificatore, che ha una issue di verifica parallela, controlla il lavoro svolto:
   a) se corretto vengono confermate le modifiche e la issue principale e quella del verificatore vengono chiuse;
   b) altrimenti le issue rimangono aperte e il verificatore suggerisce dei cambiamenti e/o correzioni a carico dell'assegnatario, una volta apportate tali modifiche si torna al punto 3;

4.2. Coordinamento

4.2.1. Comunicazioni e Riunioni

Le comunicazioni principali che avvengono durante lo svolgimento del progetto sono di due tipi:
- **Comunicazioni interne**: Il gruppo utilizza **Telegram** e **Discord** per le comunicazioni principali interne, in particolare Telegram viene usato per messaggi brevi, veloci e informali, mentre Discord viene usato per discussioni e riunioni a distanza.
  Inoltre, in caso di problemi su Telegram, il gruppo può spostarsi su Discord per comunicare anche in modo non formale;
- **Comunicazioni esterne**: Per le comunicazioni esterne vengono usate la mail di gruppo **unipd.codehex16@gmail.com** e **Zoom** per chiarire dubbi e porre domande;

Anche le riunioni sono di due tipi:
- **Riunioni interne**: I membri del gruppo si riuniscono su **Discord** dove si fa il punto della situazione in quel momento e se necessario si introducono nuove attività da svolgere e/o si procede con quelle già indicate;
- **Riunioni esterne**: I membri del gruppo si riuniscono insieme al proponente, questa riunione di solito è richiesta dal gruppo per trattare problemi/dubbi di una certa importanza al fine di avere una migliore comprensione delle attività da svolgere e procedere con il progetto;

Sarà compito del **Responsabile** riassumere in un **verbale interno** o un **verbale esterno** quello che si è discusso durante la riunione in questione;

4.2.2. Verbali

I verbali redatti sono di due tipi:
- **Verbali interni**: Questo tipo di verbale è la trascrizione dei punti salienti di una riunione interna e, generalmente, l'obiettivo principale è quello di discutere delle attività svolte e delle attività da svolgere, stabilendo quindi nuove issue, se necessario;
- **Verbali esterni**: Questo tipo di verbale è la trascrizione dei punti più importanti riscontrati durante una riunione esterna, il cui obiettivo principale è quello di risolvere dubbi/problemi riscontrati durante l'avanzamento del progetto;

Nel caso in cui il Responsabile non fosse presente durante le riunioni, il verbale corrispondente verrà redatto dall'**Amministratore**; se neanche l'amministratore fosse presente allora sarà uno dei membri presenti alla riunione ad avere l'incarico di redigere il verbale.

4.3. Miglioramento

4.3.1. Scopo e descrizione

Il miglioramento è un processo sempre attivo che durerà per tutto il progetto il cui obiettivo è quello di controllare e migliorare tutto quello che viene prodotto mantenendo un elevato grado di qualità. In particolare si cerca e si cercherà di ruotare ruoli, identificare attività idonee ai membri del gruppo così da risolvere il maggior numero di problemi che si potrebbero venire a creare e/o colmare tempestivamente eventuali lacune mantenendo il lavoro del team elastico e flessibile.