Norme di Progetto

2025-04-05

Responsabile	Manuel Cinnirella, Giovanni Battista Matteazzi
	Alessandro Di Pasquale, Nicolò Bovo
	Massimo Chioru, Elia Leonetti
Redattori	Giovanni Battista Matteazzi
	Alessandro Di Pasquale, Nicolò Bovo
	Massimo Chioru, Manuel Cinnirella
Verificatori	Romeo Calearo, Giovanni Battista Matteazzi
	Romeo Calearo, Manuel Cinnirella
	Alessandro Di Pasquale, Nicolò Bovo

AlphaCode
Università Degli Studi di Padova
Versione 1.1.0

Registro delle modifiche

Vers.	Data	Descrizione	Autore	Verificatore
1.1.0	2025-08-08	Correzioni di struttura e coerenza	Massimo Chioru, Alessandro Di Pasquale, Nicolò Bovo	Romeo Calearo
1.0.0	2025-07-07	Controllo finale del documento	Massimo Chioru, Giovanni Battista Matteazzi	Romeo Calearo
0.9.0	2025-06-18	Estensione del documento	Alessandro Di Pasquale, Manuel Cinnirella	Giovanni Battista Matteazzi
0.5.0	2025-05-20	Estensione del documento	Massimo Chioru, Manuel Cinnirella	Romeo Calearo
0.1.0	2025-04-05	Bozza del documento	Alessandro Di Pasquale, Nicolò Bovo	Manuel Cinnirella

Indice

I. Introduzione 6
I - 1. Scopo del documento 6
I - 2. Scopo del progetto 6
I - 3. Riferimenti 6
I - 3.1. Riferimenti normativi 6
I - 3.2. Riferimenti informativi 7
II. Processi primari 7
II - 1. Fornitura 7
II - 1.1. Scopo 7
II - 1.2. Descrizione 7
II - 1.3. Aspettative 7
II - 1.4. Comunicazioni con il proponente 8
II - 1.5. Documentazione fornita 8
II - 1.5.1. Valutazione dei Capitolati 8
II - 1.5.2. Analisi dei Requisiti 9
II - 1.5.3. Piano di Progetto 9
II - 1.5.4. Piano di Qualifica 10
II - 1.5.5. Glossario 11
II - 2. Sviluppo 11
II - 2.1. Scopo 11
II - 2.1.1. Implementazione del processo 11
II - 2.2. Descrizione 12
II - 2.2.1. Analisi dei requisiti di sistema 13
II - 2.2.2. Analisi dei requisiti software 13
II - 2.2.3. Progettazione architetturale del sistema 14
II - 2.3. Progettazione architetturale del software 14
II - 2.4. Progettazione dettagliata del software 15
II - 2.5. Codifica e testing del software 15
II - 2.6. Integrazione del software 16
II - 2.7. Test di qualifica del software 16
II - 2.8. Integrazione di sistema 17
II - 2.9. Test di qualifica del sistema 17
II - 2.10. Installazione del software 18
II - 2.11. Supporto all'accettazione del software 18
III. Processi di supporto 19
III - 1. Documentazione 19
III - 1.1. Scopo 19
III - 1.2. Descrizione 19
III - 1.3. Aspettative 19
III - 1.4. Ciclo di vita 19
III - 1.5. Sistema di composizione tipografica 20
III - 1.6. Struttura dei documenti 20

III - 1.6.1. Intestazione 21
III - 1.6.2. Registro delle modifiche 21
III - 1.6.3. Indice 21
III - 1.6.4. Corpo del documento 21
III - 1.6.5. Corpo del verbale 21
III - 1.6.6. Documenti del progetto 22
III - 1.7. Regole stilistiche 22
III - 1.7.1. Convenzioni di denominazione dei file 22
III - 1.7.2. Stile del testo 22
III - 1.8. Formato delle date 23
III - 1.9. Strumenti 23
III - 2. Verifica 24
III - 2.1. Scopo e aspettative 24
III - 2.2. Descrizione 24
III - 2.3. Analisi statica 24
III - 2.4. Analisi dinamica 25
III - 2.4.1. Test di unità 25
III - 2.4.2. Test di integrazione 26
III - 2.4.3. Test di sistema 26
III - 2.4.4. Test di regressione 26
III - 2.4.5. Test di accettazione 27
III - 3. Validazione 27
III - 3.1. Scopo e aspettative 27
III - 3.2. Descrizione 27
III - 4. Gestione della configurazione 28
III - 4.1. Descrizione 28
III - 4.2. Scopo 28
III - 4.3. Versionamento 28
III - 4.4. Tecnologie utilizzate 28
III - 4.5. Repository 29
III - 4.5.1. Lista Repository 29
III - 4.5.2. Struttura delle repository 29
III - 5. Processi organizzativi 29
III - 5.1. Gestione dei processi 29
III - 5.1.1. Coordinamento 29
III - 5.2. Miglioramento 30
III - 5.2.1. Scopo 30
III - 5.2.2. Descrizione 30
III - 5.2.3. Metriche 30
III - 5.3. Formazione 30
III - 5.3.1. Scopo 30
III - 5.3.2. Descrizione 30
III - 5.3.3. Strumenti 31
III - 6. Standard di qualità 31
III - 6.1. Standard ISO/IEC 9126 per la qualità 31

III - 6.1.1. Funzionalità 31
III - 6.1.2. Affidabilità 31
III - 6.1.3. Usabilità 31
III - 6.1.4. Efficienza 32
III - 6.1.5. Manutenibilità 32
III - 6.1.6. Portabilità 32
IV. Metriche per la qualità 32
IV - 1. Metriche per la qualità del processo 32
IV - 1.1. Processi primari 32
IV - 1.1.1. Fornitura 32
IV - 1.1.2. Sviluppo 34
IV - 1.2. Processi di supporto 34
IV - 1.2.1. Documentazione 34
IV - 1.2.2. Gestione della qualità 34
IV - 1.3. Processi organizzativi 35
IV - 1.3.1. Gestione dei processi 35
IV - 2. Metriche per la qualità del prodotto 35
IV - 2.1. Funzionalità 35
IV - 2.2. Affidabilità 36
IV - 2.3. Efficienza 36
IV - 2.4. Usabilità 36
IV - 2.5. Manutenibilità 37

I. Introduzione

I - 1. Scopo del documento

Il presente documento si propone di definire in modo organico e condiviso il «Way of Working$_G$» adottato dal gruppo AlphaCode$_G$ per l'esecuzione del progetto didattico, allineandosi ai processi di Project Planning, Project Monitoring & Control e Organizational Project-Enabling così come descritti nella norma ISO/IEC 12207:1995$_G$. In particolare, esso raccoglie e formalizza le best practice, i ruoli, le responsabilità e le attività che ciascun membro del team dovrà seguire, al fine di garantire un approccio professionale, coerente e riproducibile lungo tutto il ciclo di vita del progetto.

Per la sua redazione si è seguito un metodo incrementale: inizialmente è stato sviluppato un impianto base che descrive i processi fondamentali e i deliverable principali; successivamente, con l'avanzare del lavoro e il confronto continuo all'interno del team, sono state aggiunte e raffinate procedure operative, modelli di documentazione e linee guida per la gestione delle modifiche e dei rischi. Ogni nuova versione di questo documento terrà conto delle decisioni prese dal gruppo durante le revisioni periodiche, garantendo così flessibilità e capacità di adattamento senza compromettere la coerenza con gli standard internazionali.

I membri del gruppo AlphaCode si impegnano a consultare regolarmente questo documento e a rispettarne le indicazioni, assicurando così uniformità nei processi di pianificazione, monitoraggio e controllo del progetto. Tale impegno non solo favorisce la trasparenza nei confronti del proponente, ma consente anche di mantenere alta la qualità del lavoro e di tracciare in modo puntuale l'avanzamento delle attività, come prescritto dal processo di Configuration Management della ISO/IEC 12207:1995$_G$.

I - 2. Scopo del progetto

Lo scopo di NearYou$_G$ è progettare, realizzare e validare una piattaforma di advertising personalizzata in tempo reale$_G$, capace di unire flussi GPS$_G$, profilazione utente e AI generativa$_G$ per produrre annunci altamente contestuali. Oltre a implementare simulatori di posizionamento, pipeline$_G$ Kafka$_G$⟶ClickHouse$_G$⟶PostGIS$_G$ e modelli LLM$_G$, il team AlphaCode$_G$ condurrà misurazioni mirate per valutare throughput$_G$ e latenza$_G$ delle componenti di streaming$_G$ (Kafka$_G$) e orchestrazione$_G$ (Airflow$_G$), l'efficacia e i tempi di risposta delle query geospaziali$_G$ in PostGIS$_G$, e la velocità di generazione e consegna degli annunci via FastAPI$_G$. Analizzeremo inoltre l'impatto delle operazioni AI$_G$ su CPU e memoria e testeremo la resilienza dei microservizi$_G$ (producer$_G$, consumer$_G$, webapp$_G$) in caso di picchi di traffico o failure$_G$ di rete. Per confermare scalabilità$_G$, robustezza delle logiche di prossimità e manutenibilità del codice, AlphaCode$_G$ eseguirà stress test$_G$ ad alto carico su tutte le pipeline$_G$, verificando la correttezza funzionale e le performance sotto condizioni estreme.

I - 3. Riferimenti

I - 3.1. Riferimenti normativi

- Capitolato d'appalto C4: NearYou, sistema di advertising

- Standard ISO/IEC 12207:1995

I - 3.2. Riferimenti informativi

- I processi di ciclo di vita del software
- Glossario (v1.0.0)

II. Processi primari

II - 1. Fornitura

II - 1.1. Scopo

Il processo di fornitura, così come previsto dal capitolato NearYou$_G$ e in linea con i principi della ISO/IEC 12207:1995$_G$, costituisce il momento in cui il team AlphaCode$_G$ definisce formalmente i termini della collaborazione con il proponente (*Sync Lab*$_G$). Attraverso un contratto condiviso, vengono stabiliti i requisiti funzionali e non funzionali - dal simulatore GPS$_G$ fino all'integrazione con i modelli di AI generativa$_G$ - i vincoli tecnologici e di sicurezza (ad esempio l'adozione di SSL/TLS$_G$ e di database PostGIS$_G$/ClickHouse$_G$) e le milestone temporali per la consegna di ciascun deliverable. Questo accordo non è un semplice atto burocratico, ma la bussola che guida ogni attività del gruppo, permettendo di tracciare costantemente lo stato di avanzamento, individuare eventuali scostamenti e riallinearsi alle attese del proponente. Solo dopo aver formalizzato questi punti, AlphaCode$_G$ procede alla redazione del Piano di Progetto, che dettaglia in modo sistematico tempi, risorse, ruoli e modalità di controllo necessari per una realizzazione ordinata e trasparente.

II - 1.2. Descrizione

Questo processo è diviso nelle seguenti fasi:

- Inizializzazione;
- Preparazione delle risposte alle richieste;
- Contrattazione;
- Pianificazione;
- Esecuzione e controllo;
- Revisione e valutazione;
- Consegna e completamento.

II - 1.3. Aspettative

Il gruppo *AlphaCode*$_G$ si impegna a mantenere una collaborazione costante e professionale con l'azienda proponente *Sync Lab*$_G$, facendo riferimento ai contatti indicati nel capitolato, in particolare i referenti tecnici Davide Zorzi e Andrea Dorigo, con Federico Pallaro sempre in copia nelle comunicazioni.

Attraverso un dialogo continuo, il gruppo intende assicurarsi che il lavoro svolto sia allineato alle richieste del capitolato, ricevendo feedback regolari e chiarendo tempestivamente eventuali dubbi. L'obiettivo è garantire che vincoli, requisiti e scadenze siano rispettati, offrendo trasparenza sullo stato di avanzamento del progetto e favorendo una collaborazione efficace durante tutte le fasi dello sviluppo.

II - 1.4. Comunicazioni con il proponente

Per garantire un flusso comunicativo efficace e trasparente, il gruppo AlphaCode$_G$ ha concordato con l'azienda proponente *Sync Lab*$_G$ una cadenza settimanale per gli incontri esterni tramite piattaforma Google Meet, durante i quali vengono affrontate le principali criticità e discussi i progressi del progetto. Tali riunioni avvengono generalmente in modalità telematica e vedono la partecipazione attiva dei referenti tecnici indicati nel capitolato.

Gli incontri hanno lo scopo di:

- chiarire dubbi sui requisiti o sui vincoli progettuali;
- discutere le tecnologie adottate o future integrazioni;
- raccogliere feedback sulle componenti sviluppate.

Ogni colloquio viene documentato attraverso la stesura di un Verbale Esterno, che include data, partecipanti e contenuti rilevanti emersi dalla discussione. I verbali vengono salvati in una cartella dedicata all'interno della repository ufficiale del progetto, in modo da garantirne la tracciabilità e l'accessibilità per tutte le parti coinvolte.

II - 1.5. Documentazione fornita

Il gruppo AlphaCode$_G$ si impegna a produrre e consegnare una documentazione completa e strutturata, rivolta sia all'azienda proponente *Sync Lab*$_G$, sia ai committenti Prof. Tullio Vardanega e Prof. Riccardo Cardin.

II - 1.5.1. Valutazione dei Capitolati

Il documento di Valutazione dei Capitolati ha lo scopo di analizzare in maniera approfondita i progetti proposti dalle aziende nella giornata del 2025/03/04. Per ogni capitolato presentato, il gruppo AlphaCode$_G$ ha identificato le esigenze del proponente, le potenziali soluzioni tecniche e le eventuali criticità che avrebbero potuto ostacolare lo sviluppo.

La valutazione si articola in sezioni ben definite che comprendono:

- **Descrizione**: include il nome del progetto, l'azienda proponente e le informazioni principali relative al prodotto, così come presentato;
- **Dominio applicativo**: definisce il contesto in cui il progetto si colloca, con particolare attenzione agli utenti finali e agli scenari d'uso;
- **Dominio tecnologico**: elenca gli strumenti, linguaggi, framework e infrastrutture richiesti;
- **Aspetti positivi**: evidenzia le opportunità di apprendimento e di innovazione legate al progetto;
- **Fattori critici**: analizza eventuali rischi o complessità, sia tecniche che organizzative;
- **Conclusione**: motiva la scelta o la non scelta del capitolato sulla base di criteri condivisi dal gruppo.

Nel caso specifico, il gruppo AlphaCode$_G$ ha scelto di aderire al Capitolato C4 - NearYou$_G$, proposto dall'azienda *Sync Lab*$_G$, per l'interesse dimostrato verso le tecnologie di streaming$_G$ dati, la componente geospaziale$_G$ e l'integrazione con tecniche di AI$_G$ generativa.

II - 1.5.2. Analisi dei Requisiti

Il documento di Analisi dei Requisiti ha il compito di descrivere nel dettaglio le funzionalità che il sistema NearYou$_G$ dovrà offrire, oltre ai vincoli da rispettare affinché il prodotto risulti conforme alle aspettative espresse nel capitolato e alle necessità del proponente *Sync Lab*$_G$.

All'interno del documento vengono fornite le seguenti informazioni:

- **Descrizione del prodotto**: viene delineato l'obiettivo generale del sistema, ovvero la creazione di una piattaforma per l'advertising personalizzato basata su flussi GPS$_G$, profilazione utenti e AI$_G$ generativa, con evidenza sulle sue principali funzionalità in ambito di raccolta, trasformazione e analisi dei dati.
- **Casi d'uso**: il sistema viene descritto attraverso una serie di scenari d'uso concreti che illustrano come gli utenti interagiscono con la piattaforma. Ogni caso d'uso evidenzia lo scenario, gli attori coinvolti (ad esempio: utenti, amministratori$_G$, processi automatici) e le azioni che questi possono compiere.
- **Requisiti funzionali e non funzionali**: sono elencati tutti i requisiti identificati, siano essi forniti esplicitamente dal proponente o dedotti dal gruppo sulla base delle funzionalità richieste. Tali requisiti vengono classificati come:
  - **Obbligatori**: indispensabili per il funzionamento minimo del sistema;
  - **Desiderabili**: utili al miglioramento della qualità e dell'esperienza utente;
  - **Opzionali**: funzionalità aggiuntive implementabili in funzione del tempo disponibile.
  - **Facoltativi**: funzionalità che potrebbero essere implementate in futuro, ma non sono essenziali per il rilascio iniziale.

L'analisi dei requisiti rappresenta la base per l'intero processo di sviluppo e verrà aggiornata in modo incrementale$_G$ in caso emergano nuove necessità o cambiamenti durante le fasi progettuali.

II - 1.5.3. Piano di Progetto

Il Piano di Progetto è un documento strategico, redatto e mantenuto dal Responsabile$_G$ di Progetto con il supporto dell'Amministratore$_G$, che ha lo scopo di guidare lo sviluppo del sistema NearYou$_G$ attraverso una pianificazione strutturata, realistica e monitorabile. Viene costantemente aggiornato per riflettere l'evoluzione delle attività, dei vincoli e delle decisioni progettuali.

Il documento comprende:

- **Analisi dei rischi**: il gruppo identifica e classifica le possibili problematiche che potrebbero compromettere il corretto svolgimento delle attività, distinguendo tra:
  - *Rischi organizzativi*, legati a dinamiche di gruppo, pianificazione e comunicazione;
  - *Rischi tecnologici*, relativi all'integrazione di strumenti complessi come Kafka$_G$, ClickHouse$_G$, PostGIS$_G$ e modelli AI$_G$ generativi.

Per ogni rischio vengono definite misure preventive e correttive per contenerne l'impatto.

- **Modello di sviluppo**: viene illustrata la strategia metodologica adottata dal team AlphaCode$_G$, che segue un approccio incrementale$_G$ ed iterativo, ispirato a pratiche agili,

per garantire una costante validazione degli obiettivi raggiunti e la massima adattabilità ai feedback. Al termine della fase di RTB, è stato deciso di adottare, per la fase di PB$_G$, il modello con sprint Scrum, così da strutturare il lavoro in cicli brevi e focalizzati, favorendo trasparenza, ispezione e adattamento continuo, e consentendo una fase di correzione e ritorno sugli elementi che necessitano di miglioramento.

- **Pianificazione**: il lavoro viene suddiviso in periodi chiave (milestone), ognuno dei quali è corredato da una panoramica delle attività previste, dei ruoli assegnati, delle responsabilità individuali e delle stime di impegno in termini di ore/uomo.
- **Preventivo**: include una stima temporale dettagliata del carico di lavoro previsto per ogni membro del gruppo, tenendo conto delle complessità tecniche e delle dipendenze tra le attività.
- **Consuntivo**: al termine di ciascun periodo pianificato, viene effettuata un'analisi delle attività realmente svolte rispetto a quanto previsto. Questo confronto consente di individuare eventuali scostamenti, valutarne le cause ed effettuare un aggiornamento del piano nelle fasi successive.

Il Piano di Progetto è uno strumento fondamentale per mantenere il controllo sul progresso del lavoro e per garantire il raggiungimento degli obiettivi nei tempi e modi concordati con il proponente.

II - 1.5.4. Piano di Qualifica

Il Piano di Qualifica è il documento che delinea le strategie, le metriche e le attività adottate dal gruppo AlphaCode$_G$ per garantire la qualità del sistema NearYou$_G$, sia a livello di processo che di prodotto. Rappresenta un riferimento centrale per tutti i membri del team durante lo sviluppo, al fine di assicurare che il risultato finale sia conforme alle aspettative del proponente e agli standard richiesti.

Il documento è strutturato in quattro sezioni principali:

- **Qualità di processo**: vengono definite le metriche da rispettare per monitorare l'efficacia e l'efficienza del ciclo di sviluppo. Tra questi parametri figurano la tracciabilità dei requisiti, la copertura della verifica e il rispetto delle scadenze prefissate.
- **Qualità di prodotto**: si stabiliscono i criteri per valutare le caratteristiche del software, come manutenibilità, affidabilità, usabilità e performance. Questi aspetti saranno misurati tramite indicatori oggettivi e soggettivi, per garantire un risultato stabile e robusto.
- **Test**: la sezione include la descrizione delle attività di test previste (test funzionali) e i criteri di accettazione per ogni funzionalità. I test saranno fondamentali per verificare che ogni componente del sistema rispetti i requisiti.
- **Valutazioni per il miglioramento**: al termine di ogni ciclo di verifica, verranno registrati i risultati ottenuti, segnalate le criticità emerse e proposte azioni correttive. Questo approccio incrementale$_G$ permetterà di migliorare costantemente la qualità complessiva del progetto.

Il Piano di Qualifica assume quindi un ruolo chiave nell'assicurare che il progetto NearYou$_G$ sia sviluppato secondo metodologie rigorose e orientate alla qualità, permettendo di mantenere sotto controllo tutti gli aspetti critici legati alla verifica e alla validazione del sistema.

II - 1.5.5. Glossario

Il Glossario costituisce un dizionario tecnico completo che raccoglie e definisce tutti i termini specialistici, acronimi e concetti utilizzati nella documentazione del progetto *NearYou*$_G$. Questo documento serve come riferimento unificato per garantire una comprensione univoca e coerente della terminologia adottata.

Il documento presenta:

- **Organizzazione alfabetica**: tutti i termini sono ordinati alfabeticamente per facilitare la consultazione rapida e l'individuazione delle definizioni necessarie.
- **Definizioni tecniche**: ogni termine include una spiegazione chiara e precisa del suo significato nel contesto del progetto, con particolare attenzione ai concetti legati a:
  - Tecnologie utilizzate (Docker$_G$, Kafka$_G$, ClickHouse$_G$, PostGIS$_G$, LLM$_G$, ecc.)
  - Metodologie di sviluppo (design patterns, architetture, testing)
  - Metriche e indicatori di progetto (EV, AC, CPI, SPI, ecc.)
  - Terminologia del dominio applicativo (advertising, geolocalizzazione, streaming$_G$ dati)
- **Acronimi e abbreviazioni**: espansione completa di tutte le sigle utilizzate nei documenti, con spiegazione del loro significato e contesto d'uso.
- **Concetti del dominio**: definizione di termini specifici del settore advertising e delle tecnologie geospaziali$_G$, essenziali per la comprensione del progetto.

Il Glossario garantisce che tutti i stakeholder del progetto - membri del team, committenti e proponente - condividano una comprensione uniforme della terminologia utilizzata, riducendo ambiguità e migliorando la qualità della comunicazione tecnica.

II - 2. Sviluppo

II - 2.1. Scopo

In conformità allo standard ISO/IEC 12207:1995$_G$, il processo di sviluppo ha lo scopo di definire e gestire le attività tecniche necessarie per trasformare i requisiti del proponente in un prodotto software conforme e funzionante. Per il progetto NearYou$_G$, il processo include l'analisi dei requisiti, la progettazione architetturale e di dettaglio, l'implementazione delle componenti, l'integrazione dei vari moduli, il collaudo del sistema, la sua installazione in ambiente operativo e l'accettazione finale da parte del committente. Tutte le attività sono guidate da pratiche di sviluppo incrementali$_G$ e iterative, con l'obiettivo di assicurare qualità, tracciabilità dei requisiti e aderenza ai vincoli tecnologici stabiliti nel capitolato.

II - 2.1.1. Implementazione del processo

L'implementazione del processo di sviluppo per il progetto NearYou$_G$ ha avuto inizio con la scelta di un modello di ciclo di vita incrementale$_G$ ed iterativo, compatibile con i vincoli

temporali e con l'evoluzione graduale dei requisiti. Il team AlphaCode$_G$ ha definito in modo chiaro i ruoli e le responsabilità di ogni membro, assicurando la copertura di aspetti fondamentali come la redazione della documentazione, la gestione delle configurazioni (tramite Git$_G$), la risoluzione dei problemi, e il supporto alla qualità.

Per mantenere coerenza e tracciabilità nel processo, sono stati adottati standard di codifica, strumenti di automazione (come Docker$_G$, Gitpod e Airflow$_G$), metodi di verifica continua e linguaggi adeguati allo scopo.

Il gruppo ha anche pianificato l'utilizzo di risorse condivise, ambienti di sviluppo replicabili e tool di gestione collaborativa (come GitHub$_G$ e Discord), per garantire trasparenza, coordinamento e continuità.

II - 2.2. Descrizione

Le attività che compongono il processo di sviluppo del progetto NearYou$_G$, strutturate secondo lo standard ISO/IEC 12207:1995$_G$, sono organizzate in fasi sequenziali e iterabili, ciascuna mirata alla costruzione incrementale$_G$ del sistema:

- **Definizione del processo di sviluppo**: formalizzazione del flusso operativo e delle pratiche adottate dal gruppo, incluso l'uso di strumenti e metodologie agili.
- **Analisi dei requisiti di sistema**: studio dei requisiti funzionali e non funzionali forniti dal capitolato, con valutazione di vincoli architetturali e tecnologici.
- **Progettazione dell'architettura di sistema**: definizione dei macro-componenti e della loro interazione (es. Kafka$_G$, ClickHouse$_G$, PostGIS$_G$, WebApp$_G$).
- **Analisi dei requisiti software**: individuazione delle funzionalità specifiche di ogni modulo software, come il consumer$_G$ Kafka$_G$, i DAG$_G$ di Airflow$_G$, o la web app FastAPI$_G$.
- **Progettazione architetturale del software**: suddivisione in microservizi$_G$, definizione delle interfacce, scelte tecnologiche (containerizzazione$_G$, sicurezza SSL/TLS$_G$, ecc.).
- **Progettazione dettagliata del software**: specifica dei componenti interni, logiche di business, gestione degli errori, logging$_G$, sicurezza e configurabilità.
- **Codifica e testing unitario**: sviluppo del codice con pratiche di qualità e test delle singole unità.
- **Integrazione progressiva**: combinazione delle componenti in ambienti Docker$_G$ tramite Docker Compose$_G$.
- **Test di qualifica del software**: verifica che ogni modulo soddisfi i requisiti previsti (es. throughput$_G$ del producer$_G$, correttezza del consumer$_G$, efficienza dei DAG$_G$).
- **Integrazione di sistema**: test del sistema completo, inclusi pipeline$_G$ Kafka$_G$→ClickHouse$_G$→PostGIS$_G$ e moduli AI$_G$.
- **Test di qualifica di sistema**: valutazione delle prestazioni, affidabilità, sicurezza e scalabilità$_G$ del sistema.
- **Installazione del sistema in ambiente target**: configurazione degli ambienti Gitpod e/o locali tramite Docker$_G$.
- **Supporto all'accettazione**: fornitura di documentazione tecnica e dimostrazione del prodotto, con eventuali sessioni di collaudo da parte del proponente.

II - 2.2.1. Analisi dei requisiti di sistema

Nel contesto dell'Analisi dei Requisiti di Sistema, il processo ha coinvolto il team AlphaCode$_G$ nell'esecuzione e nel supporto delle attività previste dal capitolato NearYou$_G$. La prima fase ha richiesto l'analisi dell'uso specifico del sistema in sviluppo, volto alla realizzazione di una piattaforma per l'advertising personalizzato basata sull'integrazione di dati GPS$_G$, profilazione utente e AI generativa$_G$. Tale analisi ha permesso di definire accuratamente i requisiti di sistema.

Questi requisiti includono funzionalità come il consumo e la gestione in tempo reale$_G$ di flussi Kafka$_G$, la persistenza efficiente in ClickHouse$_G$, l'esecuzione di query$_G$ spaziali in PostGIS$_G$, la generazione dinamica di annunci tramite modelli LLM$_G$, nonché la comunicazione sicura tramite certificati SSL/TLS$_G$. Sono stati inoltre definiti i requisiti relativi alle performance del sistema, all'affidabilità, alla manutenibilità, alle interfacce, all'operatività e alla sicurezza, insieme ai vincoli di progettazione, ambienti d'esecuzione (containerizzati$_G$) e requisiti di qualifica.

La fase successiva ha previsto la valutazione di tali requisiti in base alla loro rintracciabilità rispetto ai bisogni espressi dal proponente, alla coerenza interna, alla testabilità e alla fattibilità architetturale. È stata inoltre verificata la sostenibilità delle operazioni e della manutenzione a lungo termine. I risultati delle analisi sono stati accuratamente documentati nel relativo documento di Analisi dei Requisiti, che costituisce base fondante per le attività di progettazione e sviluppo.

II - 2.2.2. Analisi dei requisiti software

Nel contesto del progetto NearYou$_G$, l'analisi dei requisiti software ha avuto lo scopo di identificare e documentare in maniera strutturata tutte le caratteristiche funzionali e non funzionali necessarie alla realizzazione degli elementi software che compongono la piattaforma. Per ciascun modulo identificato (es. microservizi$_G$ Kafka$_G$, consumer$_G$, producer$_G$, FastAPI$_G$, DAG$_G$ Airflow$_G$), il team ha definito:

- *Specifiche funzionali e prestazionali*, tenendo conto delle capacità di throughput$_G$, latenza$_G$ di risposta, volume dati gestito e resilienza;
- *Interfacce esterne*, in particolare tra Kafka$_G$ ↔ ClickHouse$_G$, ClickHouse$_G$ ↔ FastAPI$_G$ e tra API$_G$ ↔ frontend;
- *Requisiti di sicurezza*, relativi alla comunicazione cifrata tramite SSL/TLS$_G$, gestione dei certificati SSL/TLS$_G$ e protezione dei dati trasmessi e memorizzati;
- *Aspetti legati all'usabilità e all'interazione utente-sistema nella webapp$_G$*, anche in scenari con accesso simultaneo e consultazione dei dati tramite dashboard$_G$
- *Definizione dei dati gestiti*, tra cui eventi GPS$_G$, utenti, POI, annunci e le strutture dei relativi database$_G$ (ClickHouse$_G$ e PostGIS$_G$);
- *Requisiti di deploy$_G$*, operatività e manutenzione dei container Docker$_G$ e dei servizi;
- *Documentazione prevista per l'utente e per lo sviluppatore* (es. API$_G$ reference, struttura delle query$_G$ in Grafana$_G$);

A completamento dell'analisi, ogni requisito è stato valutato in base a:

- Rintracciabilità rispetto ai requisiti di sistema;
- Coerenza interna ed esterna con le specifiche architetturali;

- Fattibilità di sviluppo e mantenimento.

II - 2.2.3. Progettazione architetturale del sistema

Nel contesto della progettazione architetturale del sistema NearYou$_G$, la prima fase ha previsto la definizione di un'architettura ad alto livello basata su una composizione di microservizi$_G$ containerizzati$_G$, orchestrati tramite Docker Compose$_G$. Sono stati identificati i principali componenti del sistema, tra cui: generatori di dati GPS$_G$, broker$_G$ Kafka$_G$ per lo streaming$_G$ sicuro (SSL/TLS$_G$), Bytewax$_G$ come stream processor$_G$, database$_G$ analitico ClickHouse$_G$, sistema geospaziale$_G$ PostGIS$_G$, orchestratore$_G$ Airflow$_G$, API$_G$ FastAPI$_G$ per la generazione di annunci personalizzati tramite modelli LLM$_G$ e interfaccia Grafana$_G$ per la visualizzazione dei dati.

Tutti i requisiti del sistema sono stati allocati con precisione ai rispettivi elementi architetturali, distinguendo tra componenti software, operazioni automatizzate (es. DAG$_G$ di Airflow$_G$) e interazioni manuali (es. generazione certificati, monitoraggio$_G$).

Successivamente è stata condotta una valutazione dell'architettura proposta, con particolare attenzione alla tracciabilità dei requisiti, alla coerenza tra i moduli, all'adeguatezza degli standard tecnologici adottati, alla fattibilità implementativa dei singoli elementi software e alla manutenibilità del sistema.

II - 2.3. Progettazione architetturale del software

Per ciascun elemento software identificato nel progetto NearYou, l'attività di progettazione architetturale comprende i seguenti compiti:

- **Conversione dei requisiti in architettura**: Gli sviluppatori traducono i requisiti software in una struttura architetturale ad alto livello, identificando chiaramente i componenti software necessari (ad esempio: modulo simulazione GPS$_G$, stream processor$_G$ Kafka$_G$/Bytewax$_G$, servizio LLM$_G$ con LangChain, dashboard$_G$ analitica Grafana$_G$, web-app frontend). Ogni requisito viene assegnato a un componente specifico, facilitando la successiva progettazione dettagliata. L'intera architettura viene documentata tramite diagrammi C4 (System, Container, Component).
- **Progettazione di alto livello delle interfacce**: Definizione e documentazione del progetto preliminare delle interfacce tra i componenti software (REST API$_G$, WebSocket$_G$, formati di messaggi JSON$_G$/Avro$_G$) e le interfacce esterne al sistema (ad esempio database$_G$ geospaziali$_G$).
- **Progettazione di alto livello del database$_G$**: Definizione della struttura preliminare del database$_G$ geospaziale$_G$ (PostGIS$_G$) e analitico (ClickHouse$_G$), indicando soltanto schemi tabellari principali, indici geospaziali$_G$ e strategie di partizionamento dei dati temporali, senza entrare nei dettagli dei campi o dei tipi.
- **Valutazione dell'architettura e interfacce**: Esecuzione di una valutazione strutturata dell'architettura proposta e dei progetti preliminari delle interfacce e dei database$_G$, verificando rintracciabilità dei requisiti, coerenza interna ed esterna, adeguatezza delle tecnologie suggerite (Kafka$_G$, Bytewax$_G$, LangChain, PostGIS$_G$), fattibilità della progettazione dettagliata e semplicità di manutenzione.

- **Revisione formale**: Conduzione di una o più revisioni formali (peer review) con tutto il team AlphaCode$_G$ e l'azienda proponente, validando l'architettura definita e apportando eventuali miglioramenti derivanti dai feedback raccolti.

II - 2.4. Progettazione dettagliata del software

Per ciascun componente software identificato nel progetto NearYou$_G$, l'attività prevede:

- **Progettazione dettagliata dei componenti software**: Definizione puntuale di unità software (moduli Python, funzioni, classi, query$_G$ SQL) per simulazione dati, stream processing$_G$ (Kafka$_G$/Bytewax$_G$), generazione annunci (LangChain), dashboard$_G$ analitica e web-app, in modo che ciascuna unità sia immediatamente codificabile e testabile.
- **Progettazione dettagliata delle interfacce**: Documentazione tecnica precisa delle API$_G$ REST, WebSocket$_G$ e schemi JSON$_G$/Avro$_G$, incluse specifiche complete per la codifica diretta senza ambiguità.
- **Progettazione dettagliata del database$_G$**: Definizione completa della struttura delle tabelle PostGIS$_G$ (positions, places, ads) e ClickHouse$_G$, specificando campi, tipi di dato, indici e partizioni.
- **Aggiornamento preliminare della documentazione utente (in fase di progettazione)**: Revisione e ampliamento della documentazione utente con dettagli tecnici aggiornati e indicazioni operative più complete.
- **Aggiornamento requisiti e pianificazione integrazione software**: Revisione delle strategie e dei criteri per test di integrazione software (Docker Compose$_G$).
- **Valutazione della progettazione dettagliata e test**: Analisi documentata sulla base di rintracciabilità requisiti, coerenza architetturale, semplicità di manutenzione, fattibilità dei test unitari e di integrazione.
- **Revisione formale interna**: Una o più sessioni di revisione obbligatorie del team AlphaCode$_G$ per verificare la progettazione.

II - 2.5. Codifica e testing del software

Per ciascuna unità software del progetto NearYou, il programmatore svolge le seguenti attività:

- **Codifica unità software e database$_G$**: Sviluppo e documentazione delle unità software (Python 3.12, JavaScript$_G$) e relativi schemi di database$_G$ (PostGIS$_G$, ClickHouse$_G$).
- **Esecuzione e documentazione test unitari**: Conduzione sistematica dei test automatizzati per verificare conformità ai requisiti funzionali e tecnici, riportando chiaramente i risultati in report strutturati.
- **Aggiornamento documentazione utente (in corso d'opera, post-codifica)**: Revisione e integrazione continua delle guide operative e manuali utente per riflettere tutte le modifiche effettuate nel software.
- **Aggiornamento requisiti e piano integrazione software**: Affinamento dei requisiti e del piano di test di integrazione basato su Docker Compose$_G$.

- **Valutazione codice e test**:
  - Tracciabilità codice-requisiti;
  - Coerenza interna ed esterna delle unità;
  - Adeguata copertura dei test (≥ 80 %);
  - Conformità agli standard di codifica adottati;
  - Fattibilità tecnica di integrazione e manutenzione.

II - 2.6. Integrazione del software

Per il progetto NearYou$_G$, l'integrazione software comprende:

- **Sviluppo del Piano di Integrazione**: Documentazione delle procedure$_G$ e sequenze di integrazione delle unità software (moduli Python, servizi Docker$_G$), inclusi requisiti di test, dati, responsabilità assegnate e pianificazione delle attività.
- **Esecuzione integrazione e test**: Unione progressiva delle unità software secondo il piano, effettuando test di integrazione con ambienti Docker Compose$_G$ per verificare il rispetto dei requisiti funzionali e non funzionali definiti nel capitolato.
- **Aggiornamento della documentazione utente (post-integrazione)**: Revisione e aggiornamento della documentazione operativa per riflettere i cambiamenti dovuti all'integrazione.
- **Preparazione Test di Qualificazione del Software**: Creazione di casi e procedure di test specifiche per ciascun requisito del software integrato, assicurando che l'insieme integrato soddisfi tutti i criteri di qualificazione previsti (copertura ≥ 80%).
- **Valutazione integrazione software**: Verifica con particolare attenzione a:
  - Tracciabilità completa ai requisiti di sistema;
  - Coerenza interna/esterna dell'integrazione;
  - Adeguata copertura dei test;
  - Rispetto dei metodi e standard adottati;
  - Conformità dei risultati ai requisiti previsti;
  - Fattibilità operativa e manutentiva della piattaforma.

II - 2.7. Test di qualifica del software

Per il progetto NearYou$_G$, l'attività comprende:

- **Esecuzione test di qualificazione**: Test approfonditi delle funzionalità software integrate, secondo i requisiti definiti nel Piano di Qualifica (copertura test ≥ 80%). I risultati sono formalmente documentati.
- **Aggiornamento documentazione utente (post-test di qualifica)**: Revisione della documentazione utente per riflettere tutte le modifiche e miglioramenti introdotti dopo i test.
- **Valutazione del prodotto software**: Revisione formale basata su:
  - Copertura completa dei requisiti software;
  - Conformità ai risultati attesi;
  - Fattibilità tecnica e operativa dell'integrazione e dei test di sistema;

- Manutenibilità e operatività futura del sistema.

Preparazione per integrazione di sistema: Aggiornamento del prodotto software e definizione di una baseline$_G$ stabile del codice e del design software, pronta per la fase successiva di integrazione di sistema.

II - 2.8. Integrazione di sistema

Inizialmente, le componenti software del sistema NearYou$_G$ devono essere integrate tra loro e con gli elementi di simulazione hardware (ad esempio, generatori di dati GPS$_G$ e sensori simulati), operazioni manuali (come la configurazione dei dataset utente) e altri sistemi esterni (ad esempio, modelli LLM$_G$ e piattaforme di visualizzazione). Gli aggregati, come il flusso dati (dai simulatori al message broker$_G$, allo stream processing$_G$, al database$_G$ e alla AI generativa$_G$), devono essere testati incrementalmente$_G$ durante lo sviluppo, verificando la conformità ai requisiti funzionali e non funzionali definiti nel capitolato. L'integrazione e i risultati dei test devono essere documentati in modo dettagliato, inclusi schemi logici del comportamento dei simulatori e report di copertura dei test (≥80%).

Successivamente, per ciascun requisito di qualificazione del sistema (MVP o prodotto completo), è necessario sviluppare e documentare:

- **Casi di test**: input (es. percorsi GPS$_G$ predefiniti, profili utente), output attesi (es. annunci contestuali generati da LLM$_G$), criteri di accettazione (es. latenza$_G$ massima consentita, precisione geospaziale$_G$);
- **Procedure$_G$ di test**: esecuzione manuale o automatizzata, convalidata in presenza della Proponente.

La valutazione del sistema integrato deve considerare i seguenti criteri, documentando i risultati:

- **Copertura dei test**: rispetto ai requisiti di sistema (es. supporto di più sorgenti dati, interpolazione contestuale, gestione di percorsi non predefiniti);
- **Appropriatezza dei metodi**: allineamento con le tecnologie proposte (Kafka$_G$, Bytewax$_G$, LangChain) e standard di settore per lo stream processing$_G$ e l'AI$_G$
- **Conformità ai risultati attesi**: coerenza tra annunci generati e contesto utente (es. target giovane vs. coppia), corretto salvataggio dei dati su storage ottimizzato (ClickHouse$_G$, PostGIS$_G$);
- **Fattibilità del test di qualificazione**: capacità di replicare scenari reali (es. picchi di dati, failure$_G$ di rete) e integrazione con strumenti di visualizzazione (Superset, web-app);
- **Fattibilità di operatività e manutenzione**: modularità dell'architettura, documentazione delle scelte progettuali (es. utilizzo di HiveMQ vs. RabbitMQ) e gestione degli eventuali problemi aperti.

II - 2.9. Test di qualifica del sistema

Il testing di qualificazione per NearYou$_G$ deve verificare:

- Conformità ai requisiti (simulatori GPS$_G$, LLM$_G$, dashboard$_G$ base);
- Funzionalità critiche: generazione annunci contestuali (es. target specifici), integrazione con broker$_G$ (Kafka$_G$/HiveMQ), persistenza dati (ClickHouse$_G$/PostGIS$_G$).

Criteri di valutazione:

- **Copertura test**: Requisiti avanzati (interpolazione contestuale, visualizzazione lato utente).
- **Conformità**: Correlazione annunci-contesto utente, precisione geospaziale$_G$, prestazioni storage.
- **Fattibilità operativa**: Architettura modulare, documentazione chiara (scelte tecnologiche, es. LangChain).

Post-verifica:

- Consegnare pacchetto software con codice, configurazioni, dashboard$_G$ (Superset/Grafana$_G$) e web-app demo.
- Stabilire baseline$_G$ tramite versionamento (Git$_G$) e documentazione progettuale aggiornata.

II - 2.10. Installazione del software

Il fornitore deve sviluppare un piano di installazione del sistema NearYou$_G$, considerando:

- **Ambiente di destinazione**: Configurazione di server/cloud per componenti chiave (message broker$_G$, stream processing$_G$, database$_G$, LLM$_G$);
- **Risorse necessarie**: Librerie Python (es. Faker$_G$), container Docker$_G$ (per Kafka$_G$, Bytewax$_G$, Superset), accesso a modelli LLM$_G$ (es. API$_G$ OpenAI$_G$, Groq$_G$ Cloud o open-source);
- **Supporto alla configurazione**: Assistenza alla Proponente (*Sync Lab*$_G$) per integrare dataset predefiniti, percorsi simulati e parametri di personalizzazione annunci.

In caso di sostituzione di sistemi esistenti, garantendo la compatibilità dei formati dati. Il piano deve includere schemi logici e istruzioni dettagliate per l'avvio delle simulazioni (GPS$_G$, stato fisico).

Esecuzione dell'installazione: Il fornitore installerà il software conformemente al piano, garantendo:

- **Inizializzazione corretta**: Avvio dei broker$_G$ (Kafka$_G$/HiveMQ), connessione allo stream processing$_G$ (Bytewax$_G$), configurazione dello storage (ClickHouse$_G$/PostGIS$_G$) e integrazione con LLM$_G$ (LangChain);
- **Funzionamento atteso**: Verifica che i dati simulati generino annunci contestuali e che le dashboard$_G$ (Superset/Grafana$_G$/web-app) visualizzino posizioni e messaggi in tempo reale$_G$

II - 2.11. Supporto all'accettazione del software

Il team di sviluppo dovrà coordinarsi in tutte le fasi di verifica e validazione della soluzione, ponendo particolare attenzione all'integrazione dei simulatori GPS$_G$, al corretto funzionamento del message broker$_G$ per lo streaming$_G$ dei dati e all'implementazione dei moduli di AI generativa$_G$ per la creazione di annunci personalizzati. Le prove, siano esse a livello unitario o integrato, devono dimostrare che l'intero sistema - dalla raccolta e

memorizzazione dei dati alla visualizzazione in dashboard$_G$ - soddisfi i requisiti tecnici e funzionali previsti.

Per ogni fase, è necessario documentare accuratamente i risultati, includendo le metriche prestazionali e il livello di copertura dei test (minimo 80%), in modo da identificare e risolvere eventuali criticità.

III. Processi di supporto

III - 1. Documentazione

III - 1.1. Scopo

Il processo di documentazione ha lo scopo di garantire tracciabilità, coerenza e qualità nelle attività del progetto **NearYou**$_G$, allineandosi alle specifiche del Capitolato C4 di *Sync Lab*$_G$. Le norme definiscono linee guida per la gestione della documentazione tecnica e di progetto, con particolare attenzione all'integrazione di tecnologie avanzate (AI$_G$, streaming$_G$ dati, geolocalizzazione) e al rispetto dei criteri di completamento definiti dal proponente.

III - 1.2. Descrizione

Questa sezione regolamenta la creazione, revisione e approvazione di tutti i documenti relativi al progetto, inclusi:

- Documentazione tecnica (architettura, scelte tecnologiche, test);
- Manuali utente e cliente;
- Documentazione per gli stakeholder (demo, presentazioni finali).

Le norme garantiscono conformità con i requisiti del capitolato, come l'uso di **LangChain** per gli LLM$_G$, l'integrazione di **Apache Kafka**$_G$ per lo streaming$_G$, e i criteri di testing (copertura ≥ 80%).

III - 1.3. Aspettative

- **Uniformità**: Utilizzo di template predefiniti per tutti i documenti, inclusi sezioni dedicate a:
  - Scelte tecnologiche (es. motivazioni per Kafka vs RabbitMQ);
  - Dettagli sull'integrazione LLM e personalizzazione annunci;
  - Risultati dei test end-to-end e metriche di copertura.
- **Conformità al Capitolato**: Rispetto delle specifiche *Sync Lab*$_G$ (es. dashboard$_G$ con Superset/Grafana$_G$, simulazione dati GPS$_G$).
- **Tracciabilità**: Registrazione delle decisioni progettuali e allineamento con i referenti aziendali.

III - 1.4. Ciclo di vita

Il ciclo di vita di un documento prevede sette fasi principali:

1. **Definizione o modifica del template:** Nella prima fase si procede con la definizione o la modifica del modello (template) specifico per il documento. Questo template stabilisce la struttura, gli stili di formattazione e include elementi essenziali quali titolo,

autore, data di creazione e altre informazioni basilari necessarie per la corretta identificazione del documento.

2. **Pianificazione e assegnazione delle sezioni:** Successivamente, le varie sezioni del documento vengono accuratamente pianificate e assegnate ai rispettivi Redattori, ciascuno responsabile per la redazione della propria parte in linea con le specifiche delle Norme di Progetto.
3. **Raccolta dei contenuti e stesura:** In questa fase intermedia, i Redattori raccolgono tutto il materiale necessario e procedono a realizzare una prima versione (bozza) del documento.
4. **Redazione definitiva del documento:** Durante la quarta fase, i Redattori elaborano ulteriormente le proprie sezioni, correggendo e perfezionando i contenuti, in caso di necessità, così da assicurare la conformità al modello stabilito e alle Norme di Progetto.
5. **Verifica dei contenuti:** La quinta fase prevede un controllo approfondito da parte dei Redattori sui contenuti redatti, assicurandosi che questi rispettino integralmente le Norme di Progetto e siano privi di errori logici, tecnici o di compilazione.
6. **Revisione generale del documento:** In questa fase, il documento viene sottoposto a una revisione da parte di un Verificatore incaricato, che ha il compito di assicurare la coerenza complessiva, la correttezza delle modifiche effettuate e la rispondenza del documento agli standard di qualità previsti.
7. **Approvazione finale e pubblicazione:** Infine, il documento giunge alla fase conclusiva, in cui un Responsabile effettua l'approvazione finale, certificandone la qualità e l'idoneità al rilascio. Dopo l'approvazione ufficiale, il documento viene pubblicato e distribuito nella sua versione definitiva.

III - 1.5. Sistema di composizione tipografica

Per garantire coerenza e semplicità nella produzione della documentazione, il gruppo ha adottato **Typst**$_G$ come strumento principale, in sostituzione di LaTeX$_G$. La scelta è motivata dai seguenti vantaggi:

- **Semplicità d'uso**: Sintassi simile a Markdown, ideale per redattori con competenze eterogenee.
- **Programmabilità avanzata**: Supporto nativo per logiche condizionali e gestione dinamica dei contenuti, utile per documenti tecnici complessi (es. integrazione di diagrammi architetturali dal capitolato).
- **Velocità di compilazione**: Generazione immediata di output PDF, ottimizzando i tempi di revisione.
- **Coerenza grafica**: Template predefiniti nella cartella `template/` della repository `AlphaCode#apice("G")-docs-file`, allineati alle richieste del capitolato.

III - 1.6. Struttura dei documenti

Ogni documento segue una struttura standardizzata per garantire tracciabilità e chiarezza, conforme alle aspettative del proponente *Sync Lab*$_G$.

III - 1.6.1. Intestazione

La prima pagina include:

- **Logo del gruppo e dell'università** (`imgs/group_logo.png`).
- **Nome del documento** (es. «Analisi dei Requisiti»).
- **Data** della creazione del documento.
- **Responsabile**$_G$**, Redattori, Validatori**: È un elenco dei ruoli ricoperti dai vari membri nel corso della stesura del documento (es. risultano due responsabili$_G$ elencati nel caso in cui una versione sia stata scritta sotto un determinato responsabile$_G$ e la successiva in una rotazione dei ruoli differente). Ruoli definiti in base alle competenze (es. redattore esperto in AI$_G$ per sezioni LLM$_G$).
- **Tabella versionamento**: Partendo dalla più recente in alto verso la prima in basso, contiene una breve descrizione delle modifiche fatte, chi le ha applicate e chi le ha successivamente verificate. In caso di unica versione, la suddetta tabella non risulterà visibile.
- **Versione attuale**:(e.g. `v0.3.1`).

III - 1.6.2. Registro delle modifiche

La seconda pagina è dedicata alla tabella di versionamento, la quale permette di tenere traccia delle modifiche applicate al documento. La tabella riporta le seguenti informazioni:

- Versione: il numero di versione del documento;
- Data: data di stesura della relativa versione del documento;
- Descrizione: un'introduzione sintetica alle modifiche effettuate.
- Autori: membri che hanno effettuato le modifiche;
- Validatori: membri che hanno approvato le modifiche;

III - 1.6.3. Indice

Organizzato per capitoli e sezioni, con riferimenti alle pagine.

III - 1.6.4. Corpo del documento

Strutturato in capitoli e sottosezioni con eventuali:

- **Descrizioni tecniche**: Dettagli su tecnologie (es. **PostGIS**$_G$ per dati geospaziali$_G$).
- **Diagrammi**: Riproduzione degli schemi architetturali del capitolato.

III - 1.6.5. Corpo del verbale

Include:

- **Informazioni sulla riunione**:
  - Luogo (fisico o virtuale), data, ora, partecipanti (interni/esterni).
- **Ordine del giorno**: Elenco temi discussi (es. «Integrazione LLM$_G$ con Apache Bytewax$_G$»).
- **Sintesi e decisioni**: Breve riassunto e accordi presi.
- **Consigli ricevuti**: Solo per i verbali esterni, riguarda quanto ci è stato suggerito dall'azienda su come procedere.
- **Attività individuate**: In formato tabulare, comprendendo l'eventuale Issue di GitHub$_G$ associata, la descrizione dell'attività e gli assegnatari (stabiliti sulla base dei loro ruoli secondo la rotazione corrente). Sarà attribuito un ID tra i seguenti:
  - ORG: attività di organizzazione;

- DOCS: attività relativa alla documentazione;
- incremento$_G$ N: attività relativa all'incremento$_G$ di numero N;
- Code/project: attività relativa allo sviluppo software.

III - 1.6.6. Documenti del progetto

Elenco dei documenti obbligatori, allineati al capitolato C4:

1. **Norme di Progetto** (*questo documento*).
2. **Piano di Progetto**: Timeline e allocazione risorse.
3. **Piano di Qualifica**: Metriche di testing (copertura ≥ 80%).
4. **Analisi dei Requisiti**: Casi d'uso (es. simulazione GPS$_G$, generazione annunci).
5. **Glossario**: Definizioni tecniche (es. «LLM$_G$», «stream processing$_G$»).
6. **Lettera di presentazione**: Lettera di presentazione ad RTB compiuta.
7. **Verbali Interni/Esterni**: Registri degli incontri con *Sync Lab*$_G$.

III - 1.7. Regole stilistiche

III - 1.7.1. Convenzioni di denominazione dei file

I PDF presenti nella repository dei documenti seguono precise convenzioni per la denominazione, riassunte di seguito:

Formato PascalCase per i nomi dei documenti, ovvero la prima lettera di ciascuna parola è maiuscola e non sono presenti spazi tra le parole. Assenza di spazi, sostituiti semplicemente dal formato PascalCase. Indicazione della data (formato YYYY_MM_DD) presente nel nome dei file relativi ai verbali. Indicazione esplicita della versione, posizionata alla fine del nome del file con formato vX.X.X.

In base a queste regole, i file avranno la seguente struttura di denominazione:

- **Norme di Progetto**: `NormeDiProgetto_YYYY_MM_DD_vX.X.X`
- **Piano di Progetto**: `PianoDiProgetto_YYYY_MM_DD_vX.X.X`
- **Piano di Qualifica**: `PianoDiQualifica_YYYY_MM_DD_vX.X.X`
- **Analisi dei Requisiti**: `AnalisiDeiRequisiti_YYYY_MM_DD_vX.X.X`
- **Glossario**: `Glossario_YYYY_MM_DD_vX.X.X`
- **Verbali Interni**: `VI_YYYY_MM_DD_vX.X.X`
- **Verbali Esterni**: `VE_YYYY_MM_DD_vX.X.X`

Si precisa inoltre che i file sorgenti .typ includono la versione nel nome, la quale verrà aggiunta anche al documento PDF dopo la compilazione, insieme a signed per indicare che il file è firmato digitalmente.

L'utilizzo uniforme e rigoroso di questo formato facilita notevolmente l'impiego di Git$_G$, permettendo di sfruttare appieno le funzioni di tracciamento$_G$ e confronto delle versioni (funzione «diff»)

III - 1.7.2. Stile del testo

Nei documenti (ad esclusione dei verbali) sarà obbligatorio utilizzare i seguenti stili tipografici:

Corsivo per:

- Il nome del gruppo (*AlphaCode*$_G$);
- Il nome dell'azienda proponente (*Sync Lab*$_G$).

Grassetto per:
- Termini chiave o importanti;
- Parole introduttive seguite da una descrizione negli elenchi puntati.

`Font monospace` per:
- I nomi dei documenti;
- I nomi dei file;
- I nomi delle repository;
- I nomi delle cartelle;
- I nomi dei branch;
- Frammenti o esempi di codice.

Sottolineato per:
- I collegamenti (link);
- Gli indirizzi email.

LETTERE MAIUSCOLE per:
- Le iniziali dei nomi propri;
- Gli acronimi;
- Le iniziali dei ruoli assegnati ai membri del gruppo.

III - 1.8. Formato delle date

Tutte le date presenti nei nomi dei documenti devono rispettare il formato seguente: **YYYY_MM_DD**, dove:

- **YYYY** indica l'anno con quattro cifre;
- **MM** indica il mese con due cifre;
- **DD** indica il giorno con due cifre.

Le date riportate all'interno dei documenti, invece, seguiranno la formattazione **YYYY-MM-DD**, come da ISO 8601.

III - 1.9. Strumenti

Gli strumenti selezionati dal gruppo *AlphaCode*$_G$ per la produzione e gestione della documentazione sono:
- **Typst**$_G$: linguaggio per la composizione e stesura dei documenti, utilizzando typst.app;
- **GitHub**$_G$: piattaforma di hosting e controllo versione, con la repository ufficiale del gruppo situata al seguente indirizzo: https://github.com/AlphaCodeSWE/AlphaCode-docs-file;
- **Capitolato**: il documento ufficiale fornito dalla proponente (*Sync Lab*$_G$) contenente i requisiti del progetto.

III - 2. Verifica

III - 2.1. Scopo e aspettative

All'interno del ciclo di vita del software, la verifica rappresenta un processo continuo e strutturato che prende avvio fin dalle prime fasi di progettazione e prosegue fino alla manutenzione post-rilascio. Questo processo ha lo scopo di garantire che ogni artefatto prodotto - siano essi codice sorgente, documentazione, casi di test o altri deliverable - rispetti pienamente i requisiti prefissati.

L'obiettivo principale è assicurare che ciascun prodotto realizzato dal gruppo *AlphaCode*$_G$ sia conforme agli standard qualitativi attesi in termini di correttezza, completezza e coerenza. Questo viene ottenuto mediante attività di revisione e controllo sistematico che fanno uso di tecniche di analisi e test ben definite.

Per raggiungere tali scopi, è fondamentale seguire procedure consolidate, applicare criteri di verifica affidabili e validare ogni deliverable al termine della sua produzione.

III - 2.2. Descrizione

Il processo di verifica è svolto da uno o più membri del gruppo *AlphaCode*$_G$ incaricati come Verificatori$_G$, e viene applicato in modo sistematico a tutti gli artefatti del progetto. La verifica non si configura come un'attività isolata o occasionale, bensì come un processo ciclico e reiterato, che accompagna costantemente lo sviluppo e si adatta all'evoluzione del progetto.

Il riferimento centrale per tutte le attività di verifica è il Piano di Qualifica, un documento che definisce con chiarezza:

- Gli obiettivi specifici della verifica;
- I criteri di accettazione per i diversi artefatti;
- Le tecniche e i metodi adottati per ciascuna attività di controllo.

Le attività di verifica applicabili sono descritte nelle sezioni seguenti.

III - 2.3. Analisi statica

L'analisi statica è una tecnica di verifica che viene svolta senza la necessità di eseguire il prodotto software. Essa si concentra sull'esame diretto del codice sorgente e della documentazione prodotta, al fine di individuare eventuali incongruenze, difetti o violazioni delle convenzioni stabilite nel progetto. Questa forma di analisi può essere applicata a qualunque artefatto del progetto e si basa principalmente su due modalità operative:

- **Walkthrough**: una lettura condivisa e collaborativa del contenuto, condotta tra il Verificatore$_G$ e l'autore dell'artefatto. Serve a individuare anomalie o aree da migliorare, promuovendo anche un confronto diretto e costruttivo.
- **Inspection**: una modalità più strutturata e formale rispetto al walkthrough, che si avvale di una checklist predefinita per analizzare sistematicamente il contenuto. L'inspection è particolarmente indicata per la sua efficienza nel rilevare errori in modo tempestivo e puntuale.

L'analisi statica risulta particolarmente efficace nelle prime fasi del progetto, quando la documentazione è ancora contenuta e facilmente gestibile, consentendo una verifica rapida e accurata di tutti i materiali prodotti. Si configura come una fase fondamentale per ridurre la propagazione di errori nelle fasi successive.

III - 2.4. Analisi dinamica

L'analisi dinamica è una tecnica di verifica che si basa sull'esecuzione effettiva del software per individuare malfunzionamenti, bug e comportamenti inattesi. Tale approccio consente di valutare la qualità del prodotto osservandone il comportamento in condizioni controllate e riproducibili.

Nel contesto del progetto *NearYou*$_G$ di *Sync Lab*$_G$, l'analisi dinamica riveste un ruolo fondamentale, in particolare per testare la correttezza dell'elaborazione dei dati in streaming$_G$, la generazione di messaggi pubblicitari tramite LLM$_G$ e la coerenza delle visualizzazioni fornite tramite dashboard$_G$.

Le principali attività previste in questa fase sono i test, ovvero l'esecuzione del codice con input definiti per verificarne il comportamento atteso. L'efficacia di un test deriva da due caratteristiche chiave:

- **Decidibilità**: un test è considerato valido se, a parità di input, produce sempre lo stesso output;
- **Ripetibilità**: il test deve poter essere eseguito più volte in ambienti diversi senza variazioni di risultato dovute a fattori esterni.

Per massimizzare la copertura e l'affidabilità, il gruppo *AlphaCode*$_G$ dovrà definire un opportuno dominio di test, ovvero l'insieme dei casi di prova necessari a verificare i requisiti funzionali e non funzionali, con attenzione particolare alle componenti più critiche, come i sistemi di geolocalizzazione, i filtri per il targeting$_G$ e i prompt AI$_G$. L'automazione dei test sarà supportata mediante strumenti come:

- **Driver**: componenti attivi che simulano input o comportamenti dell'utente o di sistemi esterni;
- **Stub**: elementi passivi che replicano funzionalità non ancora disponibili o isolate;
- **Logger**: moduli di tracciamento che registrano gli output e i risultati delle esecuzioni.

III - 2.4.1. Test di unità

I test di unità sono progettati per verificare singole componenti del sistema in modo isolato, come ad esempio:

- un modulo di simulazione GPS$_G$
- una funzione per il salvataggio su database$_G$
- un handler per l'elaborazione dei prompt.

Questi test vengono generalmente scritti durante lo sviluppo e sono cruciali per intercettare errori nelle fasi iniziali. Possono essere classificati in:

- **Test funzionali**: verificano che, dato un input specifico, l'output prodotto sia corretto e conforme ai requisiti funzionali attesi (es. il generatore di messaggi produce output personalizzati corretti).

- **Test strutturali**: mirano a esplorare tutti i possibili cammini logici del codice, assicurando che ogni ramo o condizione venga eseguita almeno una volta, migliorando la copertura del codice e riducendo il rischio di errori nascosti.

III - 2.4.2. Test di integrazione

I **test di integrazione** vengono progettati in seguito alla realizzazione dei test di unità, durante la fase di progettazione architetturale. Il loro scopo è verificare il corretto funzionamento delle interazioni tra i vari moduli software che, singolarmente, sono già stati testati e validati.

Nel contesto del progetto **NearYou**$_G$, questi test saranno particolarmente rilevanti per garantire il corretto scambio di dati tra i simulatori di posizione, i moduli di stream processing$_G$, il motore di generazione AI$_G$ e le dashboard$_G$ di visualizzazione.

L'integrazione tra i moduli avviene secondo un approccio incrementale$_G$, che consente di individuare e isolare più facilmente eventuali malfunzionamenti. In caso di errori critici, sarà possibile annullare temporaneamente le modifiche e ripristinare una versione stabile del sistema.

Si possono adottare due strategie principali per condurre i test:

- **Approccio Top-down**: si parte dalle componenti più esterne e interattive, come le interfacce utente o i sistemi di output, integrando progressivamente i moduli di livello inferiore. Questo approccio permette di testare fin da subito le funzionalità principali del sistema e garantisce la disponibilità anticipata di funzionalità ad alto impatto.
- **Approccio Bottom-up**: si inizia con i componenti più interni e a basso livello, come i servizi di gestione dati, database$_G$ o API$_G$ locali, per poi risalire alle componenti esterne. Questo metodo consente di costruire una base solida e ben verificata prima di affrontare l'integrazione con moduli più complessi.

III - 2.4.3. Test di sistema

I **test di sistema** sono eseguiti dopo il completamento dei test di integrazione e hanno l'obiettivo di verificare che l'intero sistema risponda correttamente ai requisiti definiti nell'Analisi dei Requisiti e nel Capitolato fornito da *Sync Lab*$_G$.

Nel caso del progetto **NearYou**$_G$, ciò include la verifica del corretto funzionamento di tutte le funzionalità principali: dalla generazione dei messaggi pubblicitari personalizzati alla corretta visualizzazione su mappa, fino all'aggiornamento in tempo reale$_G$ delle posizioni dei veicoli.

Questi test vengono condotti in un ambiente il più possibile simile a quello di produzione e coprono l'intera infrastruttura software, valutando la robustezza, l'affidabilità e la rispondenza dell'applicazione alle aspettative del committente.

III - 2.4.4. Test di regressione

I **test di regressione** vengono eseguiti per assicurarsi che eventuali modifiche o migliorie introdotte nel sistema (nuove funzionalità, bug fix, refactoring) non compromettano il comportamento corretto di funzionalità già verificate in precedenza.

Si tratta di una ripetizione selettiva di test di unità, integrazione e sistema già validati, mirata a identificare possibili effetti collaterali non intenzionali. Questo tipo di test è essenziale in progetti iterativi e modulari come **NearYou**$_G$, dove l'aggiunta di nuove componenti (es. nuove fonti dati, nuovi moduli di personalizzazione) può facilmente introdurre errori in parti già funzionanti.

III - 2.4.5. Test di accettazione

I **test di accettazione** rappresentano l'ultima fase del processo di verifica e hanno lo scopo di validare il prodotto finale rispetto ai requisiti espressi dal committente, ovvero *Sync Lab*$_G$.

Nel caso del progetto **NearYou**$_G$, i test di accettazione verificheranno che:

- il sistema generi correttamente messaggi pubblicitari personalizzati;
- i dati siano raccolti, processati e visualizzati in tempo reale$_G$
- la piattaforma rispetti tutte le funzionalità minime previste dalla consegna della P.o.C.$_G$ o, eventualmente, quelle della versione completa.

Solo il superamento di questa fase certifica ufficialmente la qualità e l'idoneità del software per l'utilizzo previsto.

III - 3. Validazione

III - 3.1. Scopo e aspettative

La validazione rappresenta una fase fondamentale del ciclo di sviluppo, poiché ha l'obiettivo di accertare che il prodotto software realizzato rispecchi fedelmente le aspettative e i requisiti indicati dal committente, in questo caso *Sync Lab*$_G$.

Questa fase non si limita alla sola verifica tecnica, ma coinvolge un'analisi complessiva dell'esperienza d'uso e della reale efficacia del sistema rispetto agli obiettivi stabiliti. I principali aspetti su cui si concentra sono:

- **Aderenza ai requisiti**: il sistema deve implementare in modo completo e corretto tutte le funzionalità richieste nel capitolato;
- **Corretto funzionamento**: il comportamento del software deve risultare coerente con la logica di progettazione e senza anomalie durante l'esecuzione;
- **Usabilità**: l'interfaccia utente e le interazioni devono risultare semplici, intuitive e comprensibili per l'utente finale;
- **Efficacia**: il sistema deve essere in grado di soddisfare le esigenze operative, migliorando la rilevanza e l'impatto della comunicazione pubblicitaria.

L'obiettivo ultimo è ottenere un prodotto completo, funzionante e apprezzabile da parte del committente, sia dal punto di vista tecnico che funzionale.

III - 3.2. Descrizione

La fase di validazione si basa in larga parte sui risultati ottenuti durante la fase di verifica, sfruttando i test documentati nelle sezioni precedenti (unità, integrazione, sistema, regressione e accettazione). L'elemento determinante per concludere con successo la validazione è il superamento del test di accettazione.

Questo test rappresenta la conferma finale che il sistema NearYou$_G$ sviluppato dal gruppo *AlphaCode*$_G$ risponde in maniera piena ai requisiti funzionali e qualitativi attesi. Solo dopo l'esito positivo di questa fase il prodotto potrà essere considerato validato e pronto per l'utilizzo o per ulteriori estensioni.

III - 4. Gestione della configurazione

III - 4.1. Descrizione

Il processo di gestione della configurazione viene applicato lungo l'intero ciclo di vita del progetto e definisce le regole adottate dal gruppo *AlphaCode*$_G$ per garantire la tracciabilità e il controllo delle modifiche apportate ai documenti e al codice sorgente. Tale processo assicura che ogni artefatto prodotto sia identificabile, monitorabile$_G$ e ripristinabile, contribuendo a mantenere ordine e coerenza all'interno del progetto.

III - 4.2. Scopo

Lo scopo della gestione della configurazione è garantire una corretta organizzazione e supervisione di tutte le modifiche effettuate sulla documentazione e sul codice. Ogni cambiamento sarà tracciabile e documentato, permettendo in qualsiasi momento di risalire:

- alla motivazione della modifica;
- all'autore che l'ha effettuata;
- al contesto temporale e funzionale del cambiamento.

Questo approccio consente una maggiore affidabilità e facilita l'analisi retrospettiva del lavoro svolto.

III - 4.3. Versionamento

Per gestire in modo ordinato le modifiche ai documenti, *AlphaCode*$_G$ adotta una convenzione di versionamento nel formato X.Y.Z, così definita:

- **X**: identifica una nuova release stabile (o di base), (es. revisione, rilascio ufficiale). Può essere incrementata solo previa validazione;
- **Y**: rappresenta una nuova release «minore», introduce aggiunte o funzionalità base;
- **Z**: nuova release di patch che indica una modifica puntuale o minore, spesso una «messa a punto» (correzioni, nessuna aggiunta considerevole).

Ad ogni incremento di una cifra, tutte quelle alla sua destra vengono azzerate (es. da `1.2.3` a `1.3.0`, oppure da `2.0.5` a `2.1.0`).

III - 4.4. Tecnologie utilizzate

Per implementare efficacemente la gestione della configurazione, il gruppo *AlphaCode*$_G$ utilizza i seguenti strumenti:

- **Git**$_G$: sistema di controllo versione distribuito per tracciare ogni modifica ai documenti e al codice sorgente;
- **GitHub**$_G$: piattaforma di hosting per repository, gestione versioni, pull request e issue tracking$_G$
- **GitPod**: ambiente di sviluppo cloud che facilita la configurazione e l'accesso uniforme agli ambienti di lavoro;

- **Docker**$_G$: per la containerizzazione$_G$ delle componenti software, garantendo coerenza tra ambienti di sviluppo e di produzione;
- **Discord**: per la comunicazione e coordinamento del team durante le attività di configurazione.

III - 4.5. Repository

III - 4.5.1. Lista Repository

Il gruppo utilizza le seguenti repository all'interno dell'organizzazione GitHub$_G$ «AlphaCodeSWE»:

- `AlphaCode#apice("G")-docs-file` : contiene tutta la documentazione del progetto;
- `NearYou#apice("G")-Code` : contiene il codice sorgente dell'applicazione, inclusi simulatori GPS$_G$, moduli di stream processing$_G$, e componenti di visualizzazione;

III - 4.5.2. Struttura delle repository

La repository dei documenti è organizzata con le seguenti cartelle principali:

- `documents/` : contiene i documenti finali in formato PDF;
- `sources/` : contiene i file sorgenti Typst;
- `template/` : contiene il template condiviso per la documentazione;
- `template/assets/` : contiene immagini e altri file utilizzati nei documenti.
- `template/fonts/` : contiene eventuali font aggiuntivi utilizzati nei documenti.

La repository del codice è organizzata secondo i principali componenti dell'architettura NearYou$_G$:

- `airflow/` : codice per gestire il caricamento periodico delle informazioni sui negozi associati nella mappa;
- `certs/` : certificati per Kafka$_G$
- `deployment#apice("G")/` : configurazione per Kafka$_G$, Docker$_G$ e programma di avvio;
- `docs/` : documenti per informazioni varie;
- `monitoring#apice("G")/` : codice di Grafana$_G$ (dashboard$_G$ amministratore$_G$) e Prometeus (monitoraggio$_G$ risorse hardware$_G$);
- `requirements/` : librerie necessarie per il corretto funzionamento del codice python;
- `services/` : codice di dashboard$_G$ utente e generatore di messaggi AI$_G$
- `src/` : contiene cache$_G$, logger$_G$ e simulatore utenti.
- Variabili di sistema e configurazione vari (`.env` , `.compose` , `.gitpod` , etc.).

Questa struttura può essere soggetta a cambiamenti giunti alla fase di PB$_G$.

III - 5. Processi organizzativi

III - 5.1. Gestione dei processi

III - 5.1.1. Coordinamento

Il coordinamento all'interno del gruppo *AlphaCode*$_G$ avviene principalmente attraverso:

- **Riunioni periodiche**: incontri per la revisione delle attività, aggiornamenti sullo stato di avanzamento e pianificazione delle prossime attività;

- **Comunicazioni interne**: utilizzo di Discord e Telegram per comunicazioni sincrone e asincrone tra i membri del team;
- **Comunicazioni con il proponente**: interazioni via email e Google Meet con i referenti di *Sync Lab*$_G$, seguendo le modalità indicate nel capitolato.

Ogni riunione viene documentata attraverso un verbale, che include:
- Data, ora e partecipanti;
- Ordine del giorno;
- Decisioni prese;
- Attività assegnate.

III - 5.2. Miglioramento

III - 5.2.1. Scopo
Il processo di miglioramento ha lo scopo di ottimizzare continuamente le pratiche di lavoro adottate dal gruppo *AlphaCode*$_G$, identificando inefficienze e implementando soluzioni per aumentare la produttività e la qualità del lavoro svolto.

III - 5.2.2. Descrizione
Il miglioramento dei processi si articola in:

- **Identificazione di problemi**: raccolta di feedback dai membri del team e analisi dei dati di performance;
- **Proposta di soluzioni**: discussione e valutazione di potenziali miglioramenti;
- **Implementazione dei cambiamenti**: applicazione delle soluzioni scelte;
- **Valutazione dell'efficacia**: monitoraggio dei risultati e raccolta di ulteriori feedback.

III - 5.2.3. Metriche
Per valutare l'efficacia del processo di miglioramento, vengono utilizzate le seguenti metriche:

- **Velocità di sviluppo**: numero di task completati;
- **Tasso di regressione**: numero di bug rilevati dopo il rilascio;
- **Tempo di risoluzione dei problemi**: media del tempo necessario per risolvere un'issue;
- **Copertura dei test**: percentuale di codice coperto dai test automatizzati.

III - 5.3. Formazione

III - 5.3.1. Scopo
Il processo di formazione mira a garantire che tutti i membri del gruppo *AlphaCode*$_G$ possiedano le competenze necessarie per contribuire efficacemente al progetto **NearYou**$_G$, con particolare attenzione alle tecnologie di streaming$_G$ dati, database$_G$ geospaziali$_G$ e modelli di linguaggio.

III - 5.3.2. Descrizione
La formazione si articola in:

- **Sessioni di studio individuali**: ogni membro studia autonomamente le tecnologie assegnate;
- **Workshop interni**: sessioni collaborative (sincrone) o spiegazioni scritte (asincrone) per la condivisione di conoscenze;
- **Sessioni tecniche con *Sync Lab***: incontri specifici con i referenti aziendali per approfondimenti su tecnologie come Apache Kafka, ClickHouse e LangChain;
- **Pair programming**: sviluppo collaborativo per trasferimento di competenze tra membri del team.

III - 5.3.3. Strumenti

Per supportare il processo di formazione, vengono utilizzati:

- **Repository privata** per la condivisione di materiale formativo;
- **Documentazione ufficiale** delle tecnologie adottate;
- **Ambienti sandbox** per sperimentazione e prove tecniche;
- **Tutorial e corsi online** per linguaggi, stream processing$_G$, AI generativa, etc.

III - 6. Standard di qualità

III - 6.1. Standard ISO/IEC 9126 per la qualità

Per la valutazione della qualità del software prodotto, *AlphaCode* fa riferimento allo standard ISO/IEC 9126$_G$, che identifica sei caratteristiche principali del software di qualità:

III - 6.1.1. Funzionalità

La piattaforma NearYou deve garantire:

- **Adeguatezza**: implementazione completa delle funzionalità di profilazione, generazione di annunci personalizzati e visualizzazione su mappa;
- **Accuratezza**: precisione nella geolocalizzazione e nella personalizzazione degli annunci;
- **Interoperabilità**: capacità di integrazione tra i diversi componenti (streaming, database, LLM);
- **Sicurezza**: protezione dei dati sensibili degli utenti.

III - 6.1.2. Affidabilità

Il sistema deve garantire:

- **Maturità**: stabilità durante l'elaborazione dei dati in streaming;
- **Tolleranza agli errori**: capacità di gestire errori nei flussi di dati o nella generazione degli annunci;
- **Recuperabilità**: ripristino del funzionamento dopo eventuali crash o errori.

III - 6.1.3. Usabilità

L'interfaccia utente deve assicurare:

- **Comprensibilità**: facilità di interpretazione degli annunci generati;
- **Apprendibilità**: curve di apprendimento rapide per utenti e client;
- **Operabilità**: semplicità di interazione con la dashboard e la web-app.

III - 6.1.4. Efficienza

Il sistema deve garantire:

- **Comportamento temporale**: tempi di risposta adeguati per la generazione degli annunci;
- **Utilizzo risorse**: ottimizzazione nell'uso di CPU e memoria, specialmente per i componenti di stream processing.
- **Richieste API**: ottimizzazione numero di chiamate all'API$_G$ di A.I. per ridurre i costi.

III - 6.1.5. Manutenibilità

Il codice deve garantire:

- **Analizzabilità**: facilità nell'individuazione di eventuali bug;
- **Modificabilità**: semplicità nell'aggiungere nuove funzionalità o sorgenti dati;
- **Stabilità**: minimizzazione degli effetti collaterali in caso di modifiche;
- **Testabilità**: copertura di test ≥ 80%.

III - 6.1.6. Portabilità

Il software deve garantire:

- **Adattabilità**: capacità di funzionare in diversi ambienti;
- **Installabilità**: facilità di setup tramite configurazione containerizzata;
- **Conformità**: aderenza agli standard tecnologici indicati nel capitolato.

IV. Metriche per la qualità

IV - 1. Metriche per la qualità del processo

IV - 1.1. Processi primari

IV - 1.1.1. Fornitura

- **MPC-PV**:
  - **Nome**: *Planned Value*
  - **Descrizione**: Rappresenta il valore che si stima di aver generato fino al momento corrente.
  - **Formula**:

$$\mathrm{PV} = \%\mathrm{LSP} * \mathrm{BAC}$$

  - **Parametri**:
    - **%LSP**: Percentuale di Lavoro Svolto secondo Pianificazione (ore previste rispetto al totale delle ore disponibili).
    - **BAC**: *Budget at Completion*, cioè il costo complessivo del progetto, definito nella fase di candidatura.
- **MPC-EV**:
  - **Nome**: *Earned Value*
  - **Descrizione**: Rappresenta il valore del lavoro realmente completato fino al momento corrente.
  - **Formula**:

$$\mathrm{EV} = \%\mathrm{LSE} * \mathrm{BAC}$$

  - **Parametri**:
    - **%LSE**: Percentuale di Lavoro Svolto Effettivamente (ore utilizzate rispetto al totale disponibile).
    - **BAC**: *Budget at Completion.*

- **MPC-AC**:
  - **Nome**: *Actual Cost*
  - **Descrizione**: Rappresenta i costi realmente sostenuti dall'avvio del progetto fino al momento corrente. È possibile recuperare tale valore in ogni momento attraverso la consultazione del Piano di Progetto.

- **MPC-EAC**:
  - **Nome**: *Estimated at Completion*
  - **Descrizione**: Rappresenta il costo previsto per completare il progetto mantenendo l'attuale livello di efficienza nell'impiego delle risorse.
  - **Formula**:

$$\mathrm{EAC} = \frac{\mathrm{BAC}}{\mathrm{CPI}}$$

  - **Parametri**:
    - **BAC**: *Budget at Completion.*
    - **CPI**: *Cost Performance Index.*

- **MPC-SV**:
  - **Nome**: *Schedule Variance*
  - **Descrizione**: Esprime se il progetto è in anticipo (valore positivo), allineato (valore zero) o in ritardo (valore negativo) rispetto alla pianificazione.
  - **Formula**:

$$\mathrm{SV} = \mathrm{EV} - \mathrm{PV}$$

  - **Parametri**:
    - **EV**: *Earned Value.*
    - **PV**: *Planned Value.*

- **MPC-BV**:
  - **Nome**: *Budget Variance*
  - **Descrizione**: Esprime se i costi sostenuti finora per il progetto sono inferiori (valore positivo) o superiori (valore negativo) rispetto a quanto previsto.
  - **Formula**:

$$\mathrm{BV} = \mathrm{PV} - \mathrm{AC}$$

  - **Parametri**:
    - **PV**: *Planned Value.*
    - **AC**: *Actual Cost.*

- **MPC-ETC**:

- **Nome**: *Estimate to Complete*
- **Descrizione**: Costo previsto per completare il progetto nella situazione attuale.
- **Formula**:

$$\mathrm{ETC} = \mathrm{EAC} - \mathrm{AC}$$

- **Parametri**:
  - **EAC**: *Estimated at Completion.*
  - **AC**: *Actual Cost.*

IV - 1.1.2. Sviluppo

- **MPC-RSI**:
  - **Nome**: Indice di Stabilità dei Requisiti
  - **Descrizione**: Indice che quantifica le modifiche dei requisiti durante lo sviluppo.
  - **Formula**:

$$\mathrm{ISR} = 100 - \left(\frac{\mathrm{RM} + \mathrm{RC} + \mathrm{RA}}{\mathrm{RT}}\right) * 100$$

  - **Parametri**:
    - **RM**: Numero di Requisiti Modificati.
    - **RC**: Numero di Requisiti Cancellati.
    - **RA**: Numero di Requisiti Aggiunti.
    - **RT**: Numero Totale di Requisiti definiti inizialmente.

IV - 1.2. Processi di supporto

IV - 1.2.1. Documentazione

- **MPC-GP**:
  - **Nome**: Indice «Gulpease»
  - **Descrizione**: Esprime il grado di leggibilità di un testo:
    - Inferiore a 80: difficile da leggere per chi possiede la licenza$_G$ elementare.
    - Inferiore a 60: difficile da leggere per chi possiede la licenza$_G$ media.
    - Inferiore a 40: difficile da leggere per chi possiede la licenza$_G$ superiore.
  - **Formula**:

$$\mathrm{IG} = 89 + \frac{300 * \mathrm{NDF} - 10 * \mathrm{NDL}}{\mathrm{NDP}}$$

  - **Parametri**:
    - **NDF**: Numero di Frasi contenute nel testo.
    - **NDL**: Numero di Lettere contenute nel testo.
    - **NDP**: Numero di Parole contenute nel testo.

- **MPC-CO**:
  - **Nome**: Correttezza Ortografica
  - **Descrizione**: Esprime il numero di errori ortografici rilevati nel testo.

IV - 1.2.2. Gestione della qualità

- **MPC-PMS**:

- **Nome**: Percentuale di Metriche Soddisfatte
- **Descrizione**: Esprime la percentuale di metriche che rispettano gli obiettivi minimi di qualità stabiliti dal Piano di Qualifica.
- **Formula**:

$$\text{PMS} = \left(\frac{\text{MS}}{\text{MT}}\right) * 100$$

- **Parametri**:
  - **MS**: Numero di Metriche Soddisfatte.
  - **MT**: Numero di Metriche Totali.

IV - 1.3. Processi organizzativi

IV - 1.3.1. Gestione dei processi

- **MPC-ET**:
  - **Nome**: Efficienza Temporale
  - **Descrizione**: Esprime la percentuale di tempo disponibile che il $team_G$ è riuscito a utilizzare per svolgere attività produttive, ovvero quelle che contribuiscono direttamente al raggiungimento degli obiettivi del progetto.
  - **Formula**:

$$\text{ET} = \left(\frac{\text{OO}}{\text{OP}}\right) * 100$$

  - **Parametri**:
    - **OO**: Ore di Orologio (tempo complessivo trascorso).
    - **OP**: Ore Produttive (tempo effettivamente impiegato in attività produttive).

IV - 2. Metriche per la qualità del prodotto

IV - 2.1. Funzionalità

- **MDP-ROS**
  - **Nome**: Requisiti Obbligatori Soddisfatti
  - **Descrizione**: Calcola la percentuale di requisiti obbligatori soddisfatti dal prodotto.
  - **Formula**:

$$\%\text{ROS} = \frac{\text{NROS}}{\text{NTRO}} * 100$$

  - **Parametri**:
    - **NROS**: Numero di Requisiti Obbligatori Soddisfatti.
    - **NTRO**: Numero Totale di Requisiti Obbligatori.

- **MDP-RDS**
  - **Nome**: Requisiti Desiderabili Soddisfatti
  - **Descrizione**: Calcola la percentuale di requisiti desiderabili soddisfatti dal prodotto.

- **Formula**:

$$\%\text{RDS} = \frac{\text{NRDS}}{\text{NTRD}} * 100$$

- **Parametri**:
  - **NRDS**: Numero di Requisiti Desiderabili Soddisfatti.
  - **NTRD**: Numero Totale di Requisiti Desiderabili.

- **MDP-RFS**
  - **Nome**: Requisiti Facoltativi Soddisfatti
  - **Descrizione**: Calcola la percentuale di requisiti facoltativi soddisfatti dal prodotto.
  - **Formula**:

$$\%\text{RFS} = \frac{\text{NRFS}}{\text{NTRF}} * 100$$

  - **Parametri**:
    - **NRFS**: Numero di Requisiti Facoltativi Soddisfatti.
    - **NTRF**: Numero Totale di Requisiti Facoltativi.

IV - 2.2. Affidabilità

- **MDP-CC**
  - **Nome**: Code Coverage
  - **Descrizione**: Calcola la percentuale di codice$_G$ eseguita durante i *test*$_G$. Per questo progetto$_G$ è necessaria una copertura pari o superiore all'80%.
- **MDP-BC**
  - **Nome**: Branch Coverage
  - **Descrizione**: Determina la percentuale di rami decisionali del codice$_G$ eseguiti durante i *test*$_G$.
- **MDP-PTCP**
  - **Nome**: Passed Test Cases Percentage
  - **Descrizione**: Calcola la percentuale di *test*$_G$ superati rispetto ai test$_G$ eseguiti.

IV - 2.3. Efficienza

- **MDP-UR**
  - **Nome**: Utilizzo di Risorse
  - **Descrizione**: Valuta l'efficienza del sistema$_G$ relativamente all'utilizzo delle risorse *hardware*$_G$, come CPU$_G$, memoria e altre risorse di sistema$_G$.

IV - 2.4. Usabilità

- **MDP-FU**
  - **Nome**: Facilità di Utilizzo
  - **Descrizione**: Calcola il numero di errori commessi dagli utenti durante l'utilizzo. Un valore basso indica un'interfaccia intuitiva.
- **MDP-TA**
  - **Nome**: Tempo di Apprendimento

- **Descrizione**: Stima il tempo necessario a un utente per apprendere l'utilizzo del *software*$_G$.

IV - 2.5. Manutenibilità

- **MDP-CCM**
  - **Nome**: Complessità Ciclomatica Media
  - **Descrizione**: Stima la complessità media del codice$_G$ sorgente mediante la misurazione del numero medio di cammini indipendenti attraverso i grafi di controllo del flusso di tutti i metodi del sistema$_G$.
  - **Formula**:

$$\mathrm{CCM} = \frac{\sum_{i=1}^{n} \mathrm{CC}_i}{n}$$

- **MDP-CS**
  - **Nome**: Code Smell
  - **Descrizione**: Identifica potenziali problemi di progettazione$_G$ o codice$_G$ che potrebbero necessitare di manutenzione.