

Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA “TULLIO LEVI-CIVITA”

CORSO DI LAUREA IN INFORMATICA

**Integrazione agente IA in un sito e-commerce
con interfaccia conversazionale**

Tesi di Laurea

Relatore

Prof. Vardanega Tulio

Laureando

Eghosa Matteo Igbinedion-Osamwonyi

Matricola 2042888

Sommario

Il presente elaborato documenta le attività svolte durante il periodo di stage (circa 300 ore) presso *Halue S.r.l.* e descrive l’inserimento nel contesto di stage, la progettazione e la realizzazione di un prototipo di interfaccia conversazionale per un sito e-commerce nel settore skincare. Il lavoro comprende l’analisi dei requisiti, la progettazione architetturale di un sistema agentico integrato con modelli di linguaggio e retrieval-augmented generation (RAG), l’implementazione di connettori verso piattaforme (ad es. Shopify e Sanity), lo sviluppo del proof-of-concept e la verifica tramite test end-to-end.

Struttura del documento. Il testo è organizzato nei seguenti capitoli e sezioni principali:

1. **Capitolo 1 — Azienda:** contesto aziendale, obiettivi, tecnologie e processi interni.
2. **Capitolo 2 — Stage:** descrizione del progetto di stage, motivazioni, pianificazione, metodo di lavoro e strumenti adottati.
3. **Capitolo 3 — Sviluppo:** requisiti funzionali e non funzionali, progettazione dell’architettura (storefront, integrazioni, database vettoriale), descrizione delle funzionalità implementate (chat, function calling, chaining, multi-agent, token streaming), test effettuati e problemi riscontrati.
4. **Capitolo 4 — Conclusioni:** valutazione degli obiettivi raggiunti lato progetto e lato personale, retrospettiva e la valutazione dell’esperienza di stage.
5. **Appendici e materiali complementari:** glossario, elenchi di figure e tabelle.

Convenzioni tipografiche. Per garantire chiarezza e uniformità nella scrittura del documento sono state adottate le seguenti convenzioni tipografiche:

Lingua: italiano.

Carattere e corpo: Times New Roman (o equivalente); corpo del testo 12 pt; titoli dei capitoli 14 pt (o come richiesto dal regolamento), interlinea 1.5.

Intestazioni: numerazione gerarchica (es. 1, 1.1, 1.1.1), titoli dei paragrafi in grassetto.

Allineamento e margini: testo giustificato; margini standard (ad es. 2.5 cm su tutti i lati) salvo diversa indicazione.

Numerazione pagine: numerazione romana per frontespizi e sommario (i, ii, ...); numerazione araba a partire dall’introduzione / Capitolo 1.

Figure e tabelle: didascalie concise sotto la figura/tabella; numerazione progressiva (Figura 1.1, Tabella 2.1); riferimenti alle figure nel testo.

Codice e output: font monospace (es. *Consolas* o *Courier New*), corpo 10 pt, blocchi di codice delimitati e con caption descrittiva se presenti.

Abbreviazioni e termini tecnici: alla prima occorrenza la forma estesa seguita dall'acronimo tra parentesi; uso coerente dell'acronimo in seguito.

Citazioni e bibliografia: seguire lo stile indicato dal relatore (se non specificato, mantenere uno stile coerente come APA o IEEE in tutto il documento).

Note tipografiche: termini in lingua straniera o parole chiave in corsivo; evitare uso eccessivo di maiuscole e colori non necessari.

Indice

| | | |
|----------|----------------------------|----------|
| 1 | Azienda | 1 |
| 1.1 | Descrizione generale | 1 |
| 1.2 | Obiettivi e valori | 2 |
| 1.3 | Tecnologie utilizzate | 3 |
| 1.4 | Processi interni | 3 |
| 2 | Stage | 4 |
| 2.1 | Strategia | 4 |
| 2.2 | Progetto di stage | 4 |
| 2.3 | Motivo della scelta | 4 |
| 2.4 | Pianificazione | 4 |
| 2.4.1 | Pianificazione settimanale | 4 |
| 2.4.2 | Requisiti | 5 |
| 2.5 | Metodo di lavoro | 5 |
| 2.6 | Tecnologie utilizzate | 5 |
| 2.6.1 | Sanity | 5 |
| 2.6.2 | Shopify | 5 |
| 3 | Sviluppo | 6 |
| 3.1 | Requisiti | 6 |
| 3.1.1 | Obbligatori | 6 |
| 3.1.2 | Facoltativi | 6 |
| 3.2 | Progettazione | 6 |
| 3.2.1 | Architettura storefronte | 6 |
| 3.2.2 | Comunicazione con shopify | 6 |
| 3.2.3 | Comunicazione con sanity | 6 |
| 3.2.4 | Sclera LLM | 6 |
| 3.2.5 | Comunicazione con agente | 6 |
| 3.2.6 | Flusso agentico | 7 |
| 3.2.7 | Database vettoriale | 7 |
| 3.2.8 | Architettura applicazione | 7 |
| 3.3 | Funzionalità | 7 |
| 3.3.1 | Chat page | 7 |
| 3.3.2 | Sanity function calling | 7 |
| 3.3.3 | Shopify function calling | 7 |
| 3.3.4 | Prompt engineering | 7 |
| 3.3.5 | Chaining | 7 |
| 3.3.6 | Multi-agents | 7 |
| 3.3.7 | Token streaming | 7 |
| 3.3.8 | Logs | 8 |
| 3.3.9 | Creazione interfaccia | 8 |
| 3.4 | Test | 8 |
| 3.4.1 | Sanity test | 8 |
| 3.4.2 | Shopify test | 8 |
| 3.5 | Problemi riscontrati | 8 |

| | |
|---|-----------|
| <i>INDICE</i> | iv |
| 3.5.1 Tempo invio risposta | 8 |
| 3.5.2 Tempo creazione interfaccia | 8 |
| 3.6 Prodotto finale | 8 |
| 4 Conclusioni | 9 |
| 4.1 Obiettivi soddisfatti | 9 |
| 4.1.1 Obiettivi effettivi | 9 |
| 4.1.2 Obiettivi personali | 9 |
| 4.2 Retrospettiva | 9 |
| 4.3 Valutazione esperienza | 9 |
| Glossario | 10 |

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Azienda

1.1 Descrizione generale

Indico come ho raccolto le informazioni e quanto le ritengo attendibili, prima di entrare nel dettaglio. Le osservazioni che seguono si basano principalmente sulla partecipazione a *meeting* (incontri di lavoro) con il tutor e su osservazioni effettuate sul posto, osservando i membri del team durante il lavoro. Per quanto riguarda le tecnologie e i processi quotidiani che ho visto direttamente, considero le informazioni affidabili; per decisioni strategiche e dati aziendali che mi sono stati riferiti in parte oralmente e in parte raccolti indirettamente, l'accuratezza è medio-bassa; per elementi finanziari o metriche non accessibili rimane bassa.

Halue S.r.l. è una società benefit di consulenza tecnica orientata all'adozione di soluzioni basate sull'intelligenza artificiale e alla fornitura di servizi a clienti sia *B2C* (business-to-consumer: azienda verso consumatore) sia *B2B* (business-to-business: azienda verso azienda). L'attività dell'azienda si dirama in vari servizi tra i quali: la progettazione e gestione di soluzioni *e-commerce* (commercio elettronico), l'implementazione di sistemi di gestione delle relazioni con i clienti (*CRM*, Customer Relationship Management), servizi legati all'IA (es. sviluppo di agenti intelligenti) e percorsi di formazione rivolti ai clienti.

L'organizzazione operativa riscontrata è tipica di una piccola realtà di consulenza: un unico team i cui membri alternano i ruoli di verificatore e sviluppatore. Si alternano attività di *delivery* (consegna/erogazione del servizio) su progetti su misura e attività di supporto e manutenzione post-consegna. Dal punto di vista operativo, ho potuto ricevere riscontro sull'uso dell'approccio agile, con gestione tramite *backlog* (elenco prioritizzato di attività) e strumenti di tracciamento; anche la comunicazione interna è strutturata a seconda delle necessità, lo sviluppo avviene su *branch* (rami di sviluppo) con pratiche di revisione del codice. Ho osservato inoltre la presenza di ambienti separati per il *testing* (collaudo) e per la produzione, e sono venuto a conoscenza dell'esistenza di procedure di rilascio automatizzate adattate ai singoli progetti.

La clientela a cui la società si rivolge, sulla base dei progetti osservati e in linea con il mio progetto di stage, va dalle piccole e medie imprese con esigenze di commercio elettronico fino a grandi committenti che richiedono integrazioni complesse e soluzioni *CRM* articolate; sono presenti commesse in ambito privato (*retail*, distributori, operatori *B2B*) e interventi rivolti a processi interni di organizzazioni di maggiori dimensioni.

La propensione all'innovazione da parte, dell'azienda è percepibile dai seguenti fattori: attenzione alla formazione interna, approcci architetturali moderni e sperimentazione su progetti legati all'IA. Tale propensione è comunque bilanciata da un approccio pragmatico al fine di privilegiare la stabilità quando richiesto da vincoli operativi o di tempo.

1.2 Obiettivi e valori

Obiettivi strategici.

- **Innovazione tecnologica applicata al cliente.** Sviluppare *proof-of-concept* (*PoC*: prototipo sperimentale) e soluzioni basate su architetture *cloud-native* (*container*: contenitori software; *serverless*: architetture senza gestione diretta del server) per aumentare la scalabilità dei servizi e ridurre il time-to-market.
- **Automazione della manutenzione.** Adottare la gestione dell'infrastruttura come codice e soluzioni di monitoraggio con *alerting* automatico per minimizzare gli interventi manuali e ridurre i tempi di ripristino.
- **Orientamento al mercato e alla clientela.** Ampliare la base clienti includendo grandi imprese e pubbliche amministrazioni, proponendo offerte modulari e scalabili.
- **Crescita delle competenze interne.** Promuovere formazione strutturata, *mentorship* (affiancamento professionale) e progetti di ricerca e sviluppo interni per mantenere aggiornato il capitale di competenze e favorire il ricambio tecnologico.
- **Sostenibilità ed efficienza.** Ottimizzare l'uso delle risorse infrastrutturali per contenere i costi operativi e ridurre l'impatto ambientale dei servizi erogati.

Valori promossi internamente.

- **Collaborazione e condivisione della conoscenza.** Pratiche operative quali *pair programming*, *code review* e sessioni tecniche ricorrenti consolidano la diffusione delle competenze e riducono il rischio di silos conoscitivi.
- **Trasparenza e responsabilità.** Cultura del *feedback* aperto, *ownership* delle attività e retrospettive regolari per migliorare processi e risultati.
- **Sperimentazione e apprendimento continuo.** Allocazione di tempo e risorse a *PoC*, *hackathon* ed attività formative; si valorizza un approccio che considera il fallimento rapido come leva di apprendimento.
- **Etica e compliance.** Attenzione alla normativa (privacy, sicurezza dei dati) e processi dedicati quando si lavora per committenze pubbliche o in ambiti regolamentati.

Collegamento tra obiettivi, valori e contesto operativo. Le scelte tecnologiche adottate (microservizi, containerizzazione, *pipeline CI/CD*, monitoraggio centralizzato) e i processi interni (sviluppo *Agile*, manutenzione gestita tramite *ticketing*, rilasci pianificati) sono coerenti con gli obiettivi strategici descritti. La clientela eterogenea impone di coniugare rapidità di innovazione con rigore normativo e di processo: questo equilibrio viene perseguito traducendo i valori dichiarati in pratiche operative concrete (ad esempio *code review* per la qualità, *pipeline* per la sicurezza e la ripetibilità). La propensione all'innovazione si manifesta tramite investimenti in *PoC*, collaborazioni esterne (partner tecnologici, atenei) e iniziative che rafforzano i valori di apprendimento e sperimentazione.

Impatto sulle relazioni interne e sul mio inserimento. Gli obiettivi e i valori aziendali hanno influenzato il mio ruolo operativo. Sono stato inserito in un contesto di lavoro con ampia autonomia, pur mantenendo un costante contatto con il team e il loro approccio. Mi è stata assegnata una postazione in ufficio; il mio account è stato aggiunto al canale Slack del team per facilitare le comunicazioni, e mi è stata lasciata piena libertà nello svolgimento delle attività che caratterizzano il processo di sviluppo del progetto.

1.3 Tecnologie utilizzate

Dal punto di vista delle piattaforme e degli strumenti operativi, le tecnologie principali rilevate sono le seguenti:

- *Salesforce Commerce Cloud* e relativi moduli *Service Cloud* / *Marketing Cloud* (suite enterprise per commercio, servizio clienti e automazione marketing): utilizzati per soluzioni enterprise (*B2B/B2C*) con esigenze avanzate di integrazione, automazione e gestione della *customer experience*;
- *Bloomreach* (piattaforma per ricerca avanzata e personalizzazione della customer journey): impiegato in progetti di scala;
- Cloud e piattaforme di deployment: *AWS*, *Google Cloud* e *Heroku* per hosting, provisioning di risorse scalabili e ambienti di *staging* e *production*;
- Strumenti di integrazione: approcci punto-a-punto e soluzioni *iPaaS* (*Integration Platform as a Service*: piattaforme per orchestrare integrazioni) per orchestrare flussi dati fra *CRM*, *ERP*, piattaforme e-commerce e *CMS*;
- Strumenti di gestione progetto e comunicazione: *Jira* per il tracciamento delle attività e *Slack* per la comunicazione operativa quotidiana;
- Controllo versione e pipeline: uso diffuso di *Git*, pipeline di integrazione continua e *test* automatici (unitari e d'integrazione) prima del rilascio.

A livello operativo queste tecnologie si traducono in pratiche concrete osservate però solo parzialmente durante lo stage: sviluppo su rami funzionali con *code review*, creazione di ambienti di *staging* e *production* su cloud, esecuzione di *test* automatici.

1.4 Processi interni

Le informazioni qui riportate derivano da osservazione diretta del mio ambiente di lavoro e da brevi confronti con il tutor interno; durante i due mesi di stage ho svolto il progetto in larga parte in autonomia, perciò la mia percezione dei processi si basa su quanto ho letto nella documentazione di progetto e nelle comunicazioni occasionali con il team.

L'organizzazione operativa osservata è essenziale e pratica:

- assegnazione delle attività tramite *Jira*.
- sviluppo locale con *Git* (sistema di controllo versione) e test principalmente locali; per i rilasci si utilizzavano ambienti di *staging* (ambiente di prova) e *production* (ambiente operativo), attivati tramite script semplici forniti nel repository;
- comunicazione informale prevalente su *Slack* e incontri occasionali per allineamenti con il tutor; non ho partecipato quotidianamente a *stand-up* (brevi riunioni di allineamento) strutturati data la natura autonoma del mio incarico;
- gestione delle richieste post-rilascio tramite apertura di *ticket* su *Jira*.

Nel corso dello stage ho anche percepito una propensione dell'azienda all'adozione di soluzioni moderne (per esempio componenti legati all'*AI* (intelligenza artificiale) o architetture *headless* (architetture con frontend separato)), tuttavia l'integrazione di queste tecnologie nel progetto a cui ho lavorato è avvenuta in modo limitato e sperimentale.

Capitolo 2

Stage

Introduzione al capitolo sullo stage

2.1 Strategia

Sezione che riporterà come lo stage si inserisce nella visione strategica da parte dell'aziendale (e dunque la propensione dell'azienda per l'innovazione). Qui descriverò parzialmente il punto 2 (perché) riportato nel file Struttura relazione finale.pdf. e lo concluderò nella sezione successiva.

2.2 Progetto di stage

Sezione in cui verrà illustrato il progetto di stage ricevuto, esplicitando le problematiche applicative che l'organizzazione intende affrontare con il tirocinio, gli obiettivi specifici prefissati e i vincoli operativi e temporali associati. Verrà inoltre evidenziato il rapporto tra la proposta di stage e la strategia più ampia dell'ente ospitante in materia di innovazione (con riferimento al ruolo e alla posizione assunta dal tutor aziendale emerse nel primo incontro) nonché le attività di supporto previste prima, durante e dopo il periodo di tirocinio. Qui concluderò la trattazione del punto 2 (perché) riportato nel file Struttura relazione finale.pdf.

2.3 Motivo della scelta

Sotto-sezione in cui verrà descritto il motivo per cui ho preferito scegliere di fare lo stage presso questa azienda rispetto ad altre, quali sono i miei obiettivi personali che mi sono auto-assegnato nello svolgimento del progetto e come si interconnettono con gli obiettivi dell'azienda.

2.4 Pianificazione

Sezione che descriverà la pianificazione riportata nel piano di lavoro.

2.4.1 Pianificazione settimanale

Sotto-sezione che riporterà in lista il contenuto del lavoro pianificato per lo stage, suddiviso nelle settimane definite a priori.

2.4.2 Requisiti

Sotto-sezione che riporterà la lista dei requisiti per il progetto presenti nel piano di lavoro.

2.5 Metodo di lavoro

Sezione che riporterà il flusso di lavoro utilizzato per lo sviluppo del progetto in accordo con il tutor aziendale. Verranno riportati pianificazione, interazioni con il tutor aziendale, revisioni di progresso, uso di diagrammi, tecniche di analisi e tracciamento dei requisiti, strumenti di verifica, ecc. Qui descriverò il punto 3.a (cosa e come) riportato nel file Struttura relazione finale.pdf.

2.6 Tecnologie utilizzate

2.6.1 Sanity

Sotto-sezione che riporterà la spiegazione e la logica della scelta del CMS Sanity.

2.6.2 Shopify

Sotto-sezione che riporterà la spiegazione e la logica della scelta delle API di Shopify.

Capitolo 3

Sviluppo

Introduzione al capitolo sullo sviluppo

3.1 Requisiti

3.1.1 Obbligatori

Sotto-sezione che riporterà la lista dei requisiti obbligatori discussi e studiati nel dettaglio.

3.1.2 Facoltativi

Sotto-sezione che riporterà la lista dei requisiti facoltativi discussi e studiati nel dettaglio.

3.2 Progettazione

3.2.1 Architettura storefronte

Sotto-sezione che riporterà la descrizione dell'architettura sviluppata per lo storefront. verranno descritti le componenti, le classi e le interazioni tra loro.

3.2.2 Comunicazione con shopify

Sotto-sezione che riporterà la descrizione delle comunicazione con le API di Shopify da parte del sistema.

3.2.3 Comunicazione con sanity

Sotto-sezione che riporterà la descrizione delle comunicazione con le API di Sanity da parte del sistema.

3.2.4 Scleta LLM

Sotto-sezione che riporterà la spiegazione e la logica relativa alla scelta dell'LLM.

3.2.5 Comunicazione con agente

Sotto-sezione che riporterà la descrizione delle comunicazione con l'agente da parte del sistema.

3.2.6 Flusso agentico

Sotto-sezione che riporterà la descrizione del flusso agentico scelto per l'agente.

3.2.7 Database vettoriale

Sotto-sezione che riporterà la spiegazione e la logica della scelta del database vettoriale.

3.2.8 Architettura applicazione

Sotto-sezione che riporterà la descrizione architetture dell'applicazione con citate tutte le componenti architetture principali e descritto come si legano tra loro. Verrà riportato un diagramma rappresentativo del flusso dell'applicazione.

3.3 Funzionalità

3.3.1 Chat page

Sotto-sezione che riporterà la descrizione dell'implementazione della pagine sul frontend per l'interazione con l'agente. Verranno riportati porzioni di codice e immagini per agevolare l'esposizione.

3.3.2 Sanity function calling

Sotto-sezione che riporterà lo scopo dei tool utili all'interazione con le api di sanity, il perché sono stati differenziati per come sono nel progetto e il perché nell'uso di wrappers. Verranno riportati porzioni di codice per agevolare l'esposizione.

3.3.3 Shopify function calling

Sotto-sezione che riporterà lo scopo dei tool utili all'interazione con le api di Shopify, il perché sono stati differenziati per come sono nel progetto e il perché nell'uso di wrappers. Verranno riportati porzioni di codice per agevolare l'esposizione.

3.3.4 Prompt engineering

Sotto-sezione che riporterà la descrizione del prompt engineering usato ai fini di ricevere risposte più consone e coerenti possibili con i requisiti desiderabili. Verranno riportati esempi per agevolare l'esposizione.

3.3.5 Chaining

Sotto-sezione che riporterà la descrizione dell'implementazione del pattern chaining all'interno del flusso agentico. Verranno riportati porzioni di codice e immagini rappresentative del pattern per agevolare l'esposizione.

3.3.6 Multi-agents

Sotto-sezione che riporterà la descrizione dell'implementazione del pattern multi-agent all'interno del flusso agentico. Verranno riportati porzioni di codice e immagini rappresentative del pattern per agevolare l'esposizione.

3.3.7 Token streaming

Sotto-sezione che riporterà la descrizione dell'implementazione dello streaming dei token di risposta generati dall'agente e dei log ai fini dell' UX Verranno riportati porzioni di codice per agevolare l'esposizione.

3.3.8 Logs

Sotto-sezione che riporterà la descrizione dell'output dei logs riassuntivi degli step compiuti e del relativo tempo impiegato dall'agente, il perché, il come (in streaming e non) e il loro ruolo nel contesto dell'esperienza utente. Verranno riportati porzioni di codice per agevolare l'esposizione.

3.3.9 Creazione interfaccia

Sotto-sezione che riporterà la descrizione del processo decisionale nel modo e nel perché riguardo alla creazione di un formato strutturato come output dell'agente e la sua conversione in un'interfaccia sul frontend. Verranno riportati porzioni di codice per agevolare l'esposizione.

3.4 Test

Sezione che riporterà la descrizione dei test scritti ed eseguiti per i tool chiamabili dal sistema. Verranno riportati porzioni di codice per agevolare l'esposizione.

3.4.1 Sanity test

Sotto-sezione che riporterà la spiegazione e il codice relativo ai test scritti e svolti per i tool finalizzati all'interazione tra il sistema e le API di Sanity.

3.4.2 Shopify test

Sotto-sezione che riporterà la spiegazione e il codice relativo ai test scritti e svolti per i tool finalizzati all'interazione tra il sistema e le API di Shopify.

3.5 Problemi riscontrati

Sezione che riporterà I problemi progettuali, tecnologici e applicativi che ho affrontato. Qui descriverò il punto 3.b (cosa e come) relativo al file Struttura relazione finale.pdf.

3.5.1 Tempo invio risposta

Sotto-sezione che riporterà la problematica e la soluzione adottata relativa al tempo richiesto per l'invio di una risposta da parte dell'agente in questione, dall'istante in cui è stato interrogato.

3.5.2 Tempo creazione interfaccia

Sotto-sezione che riporterà la problematica e la soluzione adottata relativa al tempo richiesto per la creazione di un interfaccia dall'istante in cui è stato interrogato l'agente in questione.

3.6 Prodotto finale

Sotto-sezione che riporterà i risultati che ho raggiunto, sia sul piano qualitativo che su quello quantitativo. Qui descriverò il punto 3.c (cosa e come) relativo al file Struttura relazione finale.pdf.

Capitolo 4

Conclusioni

Introduzione al capitolo delle conclusioni

4.1 Obiettivi soddisfatti

Sezione che riporterà gli obiettivi e i risultati raggiunti sia sul piano qualitativo che su quello quantitativo sia su base personale che sui dati di fatto. Qui descriverò il punto 4.a relativo al file Struttura relazione finale.pdf.

4.1.1 Obiettivi effettivi

Sotto-sezione che riporterà gli obiettivi e i risultati raggiunti sui dati di fatto.

4.1.2 Obiettivi personali

Sotto-sezione che riporterà gli obiettivi e i risultati raggiunti dalla mia prospettiva.

4.2 Retrospettiva

Sezione in cui farò la retrospettiva dell'esperienza di stage.

4.3 Valutazione esperienza

Sotto-sezione che riporterà una descrizione dell'esperienza dalla mia personale prospettiva. Qui descriverò il punto 4.C relativo al file Struttura relazione finale.pdf.

Glossario

A

API Application Programming Interface – In informatica il termine indica ogni insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma. La finalità è ottenere un'astrazione tra l'hardware e il programmatore o tra software a basso e quello ad alto livello, semplificando così il lavoro di programmazione.

Agile Agile – Approccio allo sviluppo del software e alla gestione progetti basato su iterazioni brevi, feedback continui e adattamento ai cambiamenti. Include metodologie come Scrum e Kanban.

AI Artificial Intelligence (Intelligenza Artificiale) – Insieme di tecniche e algoritmi che consentono a un sistema di simulare capacità cognitive tipiche dell'essere umano: apprendimento, ragionamento, percezione e interazione.

Alerting Alerting – Sistema di notifiche automatiche che avvisa quando si verificano anomalie, errori o condizioni critiche in applicazioni e infrastrutture.

B

B2C Business-to-Consumer – Modello di business in cui un'azienda vende prodotti o servizi direttamente al consumatore finale. Esempi tipici: e-commerce al dettaglio, servizi in abbonamento per privati.

B2B Business-to-Business – Modello di business in cui le transazioni avvengono tra imprese; prodotti o servizi rivolti ad altre aziende (software aziendale, forniture industriali, consulenza).

Backlog Backlog – Elenco prioritizzato di attività, requisiti o user story che rappresentano il lavoro da svolgere su un prodotto o progetto. Nel contesto Agile esistono tipicamente un product backlog (tutte le feature/attività previste) e uno sprint backlog (selezione per uno sprint).

Branch Branch (ramo) – Linea di sviluppo separata in un sistema di controllo versione (es. Git). I branch permettono di lavorare contemporaneamente su feature, fix o release senza intaccare il ramo principale; tipiche operazioni correlate: commit, merge, rebase e pull request.

Build Build – Processo di compilazione e packaging del codice sorgente in un artefatto eseguibile o distribuibile (ad esempio un file binario, una libreria, un container).

C

CI/CD Continuous Integration / Continuous Delivery – Insieme di pratiche che prevedono integrazione continua del codice tramite test automatici (CI) e rilascio frequente e affidabile del software (CD).

CMS Content Management System – Piattaforma software che permette di creare, modificare e pubblicare contenuti digitali (pagine web, articoli, media) con minime conoscenze tecniche; esempi: WordPress, Drupal.

CRM Customer Relationship Management – Sistema e strategia per gestire le relazioni con i clienti: archiviazione anagrafiche, storico interazioni, gestione vendite e opportunità, automazione marketing e supporto post-vendita. Esempi: Salesforce, HubSpot.

Cloud-native Cloud-native – Architettura e modalità di sviluppo che sfrutta appieno le caratteristiche del cloud (scalabilità, resilienza, containerizzazione, microservizi).

Compliance Compliance – Conformità a normative, regolamenti o standard di settore (es. GDPR, ISO 27001).

Container Container – Tecnologia che consente di pacchettizzare applicazioni e dipendenze in un'unità isolata e portabile, eseguibile in diversi ambienti.

Code review Code Review – Revisione del codice sorgente da parte di uno o più membri del team, con l'obiettivo di migliorare qualità, leggibilità e sicurezza.

D

Delivery Delivery – Consegna di un prodotto, servizio o funzionalità al cliente o all'ambiente di produzione; include fasi di build, test, integrazione e deployment. In ambito Agile si parla spesso di "continuous delivery" per indicare rilascio frequente e affidabile.

Deploy Deploy (deployment) – Processo che porta il codice da un ambiente di sviluppo/testing all'ambiente di produzione (o ad un ambiente target). Comprende build, configurazione, esecuzione di script di migrazione, verifica post-deploy e possibilità di rollback. Spesso automatizzato tramite pipeline CI/CD.

F

Feedback Feedback – Informazione restituita a un individuo o a un team sul lavoro svolto, utile per migliorare prodotti, processi e comportamenti.

Frontend Frontend – Parte visibile di un'applicazione o sito web con cui interagisce direttamente l'utente (UI/UX).

G

Git Git – Sistema distribuito di controllo versione, utilizzato per tracciare modifiche al codice sorgente e coordinare il lavoro tra più sviluppatori.

H

Hackathon Hackathon – Evento intensivo, spesso di breve durata (24-48 ore), in cui team di sviluppatori e designer collaborano per creare prototipi o soluzioni innovative.

Headless **Headless** – Architettura in cui il frontend è separato dal backend (ad esempio in un CMS), permettendo maggiore flessibilità e integrazione multicanale.

I

IaC Infrastructure as Code – Pratica di gestire e configurare infrastrutture tramite file di codice, versionabili e automatizzabili, invece che con operazioni manuali.

M

Meeting **Incontro** (fisico o virtuale) tra partecipanti con uno scopo definito (allineamento, decisione, pianificazione); solitamente ha un'agenda, una durata prevista e produce output (verbale, action items).

Mentorship **Mentorship** – Relazione in cui una persona più esperta guida e supporta la crescita professionale di un collega meno esperto.

Microservizi **Microservizi** – Architettura software in cui le funzionalità di un'applicazione sono suddivise in servizi indipendenti che comunicano tra loro tramite API.

Monitoraggio **Monitoraggio** – Attività di osservazione costante di sistemi e applicazioni per garantire disponibilità, prestazioni e sicurezza.

Manutenzione reattiva **Manutenzione reattiva** – Approccio alla manutenzione che interviene solo dopo il verificarsi di un guasto o problema.

MTTR Mean Time To Recovery/Repair – Indicatore che misura il tempo medio necessario per ripristinare un servizio dopo un guasto.

O

Ownership **Ownership** – Assunzione di responsabilità da parte di un individuo o team sul corretto svolgimento e completamento di un'attività.

P

Project manager **Project Manager** – Figura responsabile della pianificazione, esecuzione e chiusura di un progetto: coordina risorse, gestisce tempi, costi e rischi, mantiene comunicazione con gli stakeholder e assicura il raggiungimento degli obiettivi.

Pair programming **Pair Programming** – Tecnica di sviluppo in cui due programmatori lavorano insieme alla stessa postazione: uno scrive il codice (driver), l'altro lo rivede in tempo reale (observer).

Patch **Patch** – Correzione software rilasciata per risolvere bug, vulnerabilità o migliorare funzionalità.

Pipeline **Pipeline** – Sequenza automatizzata di fasi che portano il codice sorgente dalla scrittura allo sviluppo, test, integrazione e distribuzione.

PoC Proof of Concept – Prototipo o esperimento preliminare volto a dimostrare la fattibilità tecnica o commerciale di un'idea o soluzione.

PMI Piccole e Medie Imprese – Categoria di imprese definita in base a numero di dipendenti e fatturato annuo; spesso con esigenze e budget diversi dalle grandi aziende.

Production Production – Ambiente operativo in cui un'applicazione è effettivamente disponibile e utilizzata dagli utenti finali.

R

Rollout Rollout – Processo di distribuzione e attivazione di una nuova funzionalità o servizio; può essere eseguito in modo graduale (phased rollout, canary release, feature flags) per minimizzare rischi e permettere rollback controllati.

RAG Retrieval-Augmented Generation – Approccio nell'intelligenza artificiale che combina un modulo di recupero (retriever) di documenti o frammenti rilevanti da una base di conoscenza con un modello generativo che produce la risposta finale basandosi sulle informazioni recuperate. Vantaggi: risposte più informate e aggiornabili; svantaggi: qualità dipendente dal retrieval e dalla gestione delle fonti.

Repository Repository – Archivio centralizzato dove viene conservato e versionato il codice sorgente di un progetto.

Release Release – Rilascio pianificato di una versione del software, solitamente accompagnato da note di rilascio che descrivono nuove funzionalità e fix.

S

Storefront Interfaccia pubblica di un negozio online: pagine di prodotto, categorie, carrello e checkout — ovvero il "volto" e l'esperienza cliente dell'e-commerce.

Slack Piattaforma di messaggistica aziendale in tempo reale per la comunicazione dei team, con canali, messaggi diretti, condivisione file e integrazioni con altri strumenti.

Store Store (negozio) – Punto di vendita o repository: può indicare un negozio fisico, uno store online (la parte che espone e vende prodotti agli utenti) oppure, in contesti software, un archivio/repository dove sono conservati pacchetti o risorse.

Serverless Serverless – Modello di esecuzione in cloud in cui l'infrastruttura è gestita dal provider e gli sviluppatori si concentrano solo sul codice, pagando a consumo.

Script Script – Sequenza di comandi automatizzati che eseguono operazioni ripetitive (ad esempio build, deploy, manutenzione).

Staging Staging – Ambiente di prova che replica la produzione e serve a testare le modifiche prima del rilascio.

Stand-up Stand-up – Breve riunione quotidiana (tipica di Scrum) in cui i membri del team condividono progressi, piani e ostacoli.

Sviluppo locale Sviluppo locale – Attività di sviluppo e test svolta sulla macchina del programmatore, prima di caricare il codice su repository condivisi o ambienti remoti.

Security-by-Design Security-by-Design – Principio secondo cui la sicurezza deve essere integrata fin dalle prime fasi di progettazione di un sistema, non aggiunta a posteriori.

T

Ticket Registro/formalizzazione di una richiesta, problema o attività (supporto, sviluppo, manutenzione); contiene descrizione, priorità, assegnatario, stato e storico delle azioni intraprese.

Testing Testing – Insieme di attività volte a verificare che un sistema o componente soddisfi i requisiti e funzioni correttamente. Tipologie comuni: unit testing, integration testing, system testing, end-to-end (E2E), acceptance testing e regression testing. Può essere manuale o automatizzato.

Time-to-market Time-to-Market – Tempo che intercorre tra l'ideazione di un prodotto/servizio e la sua effettiva disponibilità sul mercato.

U

UML Unified Modeling Language – In ingegneria del software, UML (Unified Modeling Language) è un linguaggio di modellazione e specifica basato sul paradigma object-oriented. Viene usato per descrivere soluzioni analitiche e progettuali con diagrammi formali (classi, casi d'uso, sequenze, ecc.).

Upskilling Upskilling – Processo di aggiornamento mirato delle competenze, in particolare tecniche e digitali, per adattarsi a nuove esigenze lavorative o tecnologiche.

V**W****X****Y****Z**