Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica

Dal linguaggio naturale all'itinerario: un sistema AI per l'estrazione di informazioni turistiche

Tesi di Laurea Triennale

Relatore

Prof. Baldan Paolo

Laureando

Alessandro Benin

Matricola 2042356

Anno Accademico 2024-2025

© Alessandro Benin, Luglio 2025. Tutti i diritti riservati. Tesi di Laurea Triennale: "*Dal linguaggio naturale all'itinerario: un sistema AI per l'estrazione di informazioni turistiche*", Università degli Studi di Padova, Dipartimento di Matematica "Tullio Levi-Civita".

"L'intelligenza artificiale è un mezzo non un fine"

— Gregorio Piccoli

Ringraziamenti

Desidero esprimere la mia gratitudine al professor Baldan Paolo, mio relatore, per l'aiuto e il sostegno che mi ha dato durante la stesura dell'elaborato.

Vorrei anche ringraziare, con affetto, i miei genitori per il loro sostegno, il grande aiuto e la loro presenza in ogni momento durante gli anni di studio.

Come potrei non ringraziare i miei amici per questi anni incredibili, per le mille avventure vissute e per i ricordi che porterò sempre con me? Un ringraziamento speciale, ovviamente, va a "Le tre file the best": avete reso l'università un'esperienza speciale e a tratti leggendaria. Grazie di tutto, siete stati una parte fondamentale di questo viaggio.

Infine, il mio ringraziamento più grande e sentito va a Maddalena. La sua pazienza nei momenti di stress, la sua forza nei miei momenti di dubbio e il suo costante incoraggiamento sono stati il mio punto di riferimento. Grazie per aver condiviso con me le fatiche e le soddisfazioni di questo lungo percorso e per avermi spinto, ogni giorno, a non accontentarmi e a dare il massimo. Questo traguardo non è solo un punto di arrivo per me, ma una base solida per il nostro futuro insieme.

Padova, Luglio 2025 *Alessandro Benin*

Sommario

Il presente documento descrive l'attività svolta durante il periodo di stage curricolare, della durata complessiva di circa 320 ore, dal laureando Alessandro Benin presso l'azienda Zucchetti S.p.A., nel periodo compreso tra il 23 aprile e il 20 giugno 2025.

L'obiettivo principale del progetto era lo sviluppo di una WebApp per un'agenzia viaggi, dotata di un sistema di chat intelligente. I messaggi inviati dagli utenti venivano elaborati da *NuExtract*$_G$, un modello di linguaggio *Large Language Model*$_G$(LLM) specializzato nell'estrazione automatica di informazioni da testo in linguaggio naturale verso un template strutturato. I dati così ottenuti erano destinati alla generazione automatica di commesse o fatture di pagamento per i clienti.

Lo sviluppo del progetto ha seguito metodologie *Agile*$_G$, con l'adozione del *Framework*$_G$ *Scrum*$_G$. Tutte le componenti implementate sono state accuratamente documentate e sottoposte a test per verificarne il corretto funzionamento.

Indice

1 Il lavoro in Zucchetti S.p.a. **1**

1.1 L'azienda 1

1.1.1 Sede di Lavoro 2

1.2 PROGETTO 3

1.2.1 Interesse e Obiettivi del Progetto 3

1.2.2 Percorso e risultati ottenuti 3

1.3 Organizzazione del testo 5

2 Processi e metodologie **6**

2.1 Introduzione al progetto 6

2.2 Scopo del progetto 6

2.3 Motivazioni alla Base della Scelta Progettuale 8

3 Descrizione dello stage **9**

3.1 Pianificazione del lavoro 9

3.1.1 Settimana 1 9

3.1.2 Settimana 2 10

3.1.3 Settimana 3 11

3.1.4 Settimana 4 12

3.1.5 Settimana 5 13

3.1.6 Settimana 6 14

3.1.7 Settimana 7 15

3.1.8 Settimana 8 16

3.1.9 Ripartizione delle ore 17

3.2 Analisi preventiva dei rischi 18

3.3 Obiettivi richiesti 19
3.4 Tecnologie Scelte 21
3.4.1 Backend 22
3.4.2 Frontend 32
3.4.3 Strumenti di sviluppo e supporto 34

4 Analisi dei requisiti **37**
4.1 Casi d'uso 37
4.1.1 Attori 37
4.1.2 Elenco casi d'uso 38
4.2 Tracciamento dei requisiti 45
4.3 Tabelle dei requisiti 46

5 Progettazione e codifica **51**
5.1 Struttura a Tre Livelli 51
5.1.1 Livello di Presentazione: Frontend 52
5.1.2 Livello Applicativo: Backend 53
5.2 Algoritmi e Metodi rilevanti 55
5.2.1 Estrazione di Dati Strutturati con NuExtract 2.0 55
5.2.2 Generazione di Risposte Dinamiche con Qwen 3 57
5.2.3 Correzione e Normalizzazione dei Dati tramite Embedding Semantico 59

6 Competenze Maturate e Criticità Affrontate nel Progetto **60**
6.1 Risultati ottenuti 60
6.2 Competenze acquisite 63
6.3 Difficoltà affrontate 64

7 Conclusioni **65**
7.1 Possibili sviluppi futuri dell'applicazione 65
7.2 Retrospettiva 66

Glossario **i**

Sitografia v

Bibliografia vii

Elenco delle figure

1.1 Logo Zucchetti S.p.A. 1

3.1 Diagramma di GANTT 18
3.2 Logo Python 22
3.3 Logo FastApi+Unicorn 22
3.4 Logo NuExtract 23
3.5 Tabella ZeroShot inglese 24
3.6 Tabella ZeroShot multilingua 25
3.7 Logo Ollama 25
3.8 Logo QWEN 3 27
3.9 Logo Java 28
3.10 Logo Maven 29
3.11 Logo Spring Boot e Drools 30
3.12 Logo Node.js 32
3.13 Logo Vite 32
3.14 Logo React 33
3.15 Logo GitHub 34
3.16 Logo Cursor 35
3.17 Logo Lucidchart 35
3.18 Logo Microsoft Teams 36

4.1 Use Case 1: Invio Dati - Errore nella richiesta 38
4.2 Use Case 3: attesa risposta- vari scenari 39
4.3 Use Case 5: menu di uscita 40
4.4 Use Case 8: Salta Template 42

4.5 Use Case 10: Scorciatoia: Ctrl + R 43
4.6 Use Case 6: Pagina Riepilogo 44

5.1 inizio di Chat 52

6.1 riepilogo costi 61
6.2 riepilogo della vacanza 61
6.3 riepilogo della vacanza 61
6.4 riepilogo della vacanza 62
6.5 riepilogo della vacanza 62
6.6 riepilogo della vacanza 63

Elenco delle tabelle

3.1 Settimana 1 10
3.2 Settimana 2 11
3.3 Settimana 3 12
3.4 Settimana 4 13
3.5 Settimana 5 14
3.6 Settimana 6 14
3.7 Settimana 7 15
3.8 Settimana 8 16
3.9 Ripartizione delle ore per ogni settimana 17
3.10 Obiettivi richiesti 21

4.1 Tabella del tracciamento dei requisiti funzionali. 49
4.2 Tabella del tracciamento dei requisiti qualitativi. 50

4.3 Tabella del tracciamento dei requisiti di vincolo. 50

Elenco dei codici sorgenti

5.1 Funzione di estrazione dei dati dal testo 56
5.2 Prossima domanda all'utente 58

Capitolo 1

Il lavoro in Zucchetti S.p.a.

Figura 1.1: Logo Zucchetti S.p.A.

1.1 L'azienda

Zucchetti S.p.A. è una realtà di spicco nel panorama tecnologico italiano, considerata oggi la prima software house del Paese per fatturato e capacità di innovazione. Nata nel 1978 a Lodi, l'azienda ha saputo crescere nel tempo, trasformandosi da una piccola impresa locale a un gruppo riconosciuto anche a livello internazionale. La sua forza sta nella continua evoluzione e nell'affidabilità dimostrata nel fornire soluzioni gestionali sempre più avanzate.

Zucchetti si distingue per un'offerta molto ampia, che comprende strumenti per la gestione amministrativa, delle risorse umane, della sicurezza e per l'automazione dei processi aziendali. Un aspetto centrale della sua strategia è l'attenzione verso le tecnologie emergenti. In particolare, l'intelligenza artificiale sta assumendo un ruolo sempre più rilevante: viene utilizzata per rendere i software più intelligenti, capaci di apprendere dai dati, ridurre i tempi di lavoro e aumentare l'efficienza operativa.

Questa capacità di innovare costantemente e di adattarsi ai cambiamenti del mercato rende Zucchetti un esempio concreto di come la tecnologia possa guidare la trasformazione digitale delle imprese. Proprio per questo approccio orientato al futuro ho deciso di collaborare con loro per il mio tirocinio, cercando un punto di incontro tra ciò che l'azienda ricerca e le competenze che ho sviluppato nel mio percorso universitario.

1.1.1 Sede di Lavoro

Durante il mio periodo di tirocinio presso la sede di Vigonza di Zucchetti, ho avuto l'opportunità di immergermi in un ambiente dinamico e stimolante. Questa esperienza mi ha permesso di osservare da vicino e comprendere le complesse modalità operative di una grande azienda.

La comunicazione interna e la gestione delle attività sono ottimizzate dall'uso di *Microsoft Teams*$_G$ per la condivisione di informazioni e aggiornamenti. La gestione del codice e l'assegnazione dei compiti avvengono tramite *GitHub*$_G$, garantendo una collaborazione ordinata e tracciabile nello sviluppo software. L'integrazione di questi strumenti, insieme al calendario interno aziendale, offre una chiara visione dello stato di avanzamento delle attività, della disponibilità dei colleghi e delle priorità, semplificando la pianificazione e l'ottimizzazione delle risorse e dei tempi.

Ho avuto modo di confrontarmi con professionisti esperti, il cui supporto e la cui guida si sono rivelati preziosi per il mio percorso formativo. Grazie a loro, ho potuto approfondire il *Way of working*$_G$ aziendale e con i principali prodotti

in fase di sviluppo e manutenzione. La possibilità di implementare tecnologie al di fuori di quelle inizialmente previste, come l'*Embedding*$_G$, e di discutere di queste con colleghi specializzati, ha ulteriormente arricchito la mia esperienza, orientando le mie ricerche e fornendo indicazioni concrete per l'integrazione di approcci innovativi.

1.2 PROGETTO

1.2.1 Interesse e Obiettivi del Progetto

Il cuore di questo progetto di tesi è uno studio di fattibilità: vogliamo capire fino a che punto possiamo espandere l'uso degli LLM in diversi settori, sfruttando le tecnologie attuali. La frase che ha accompagnato questo progetto è: *l'intelligenza artificiale deve sempre essere un mezzo per aumentare la produttività delle persone e migliorare la qualità di prodotti e servizi.*[1]

In particolare, ci siamo concentrati sull'esplorazione di quanto un LLM, eseguito localmente e con risorse *Hardware*$_G$ limitate, possa essere efficace nell'interpretare il linguaggio naturale umano e generare risultati di qualità eccellente e professionale.

Abbiamo quindi scelto di testare le capacità degli LLM nel settore turistico, valutandone la competitività. L'applicazione specifica che abbiamo esplorato è la creazione automatica di una fattura o comanda basata su una conversazione testuale con un cliente.

1.2.2 Percorso e risultati ottenuti

Architettura e Integrazione del Sistema

La fase iniziale si è concentrata sulla progettazione di un'architettura software modulare. La comunicazione tra i diversi componenti è stata realizzata tramite *API*$_G$, assicurando un flusso di dati stabile e standardizzato. Per l'interfaccia utente, è stato sviluppato un frontend seguendo i principi del *responsi-*

[1] Gregorio Piccoli. «TriplaAAA». in: *Zucchetti S.P.A.* (2025).

ve design, per garantirne la piena compatibilità con diversi dispositivi (desktop, tablet e mobile) e la capacità di gestire eventi in tempo reale provenienti dal backend.

Ottimizzazione dei Dati e Analisi

Una volta che i sistemi comunicavano correttamente, il lavoro si è spostato sull'ottimizzazione del trattamento dei dati. Un'analisi specifica dei dati estratti dalla piattaforma NuExtract ha permesso di definire la dimensione ottimale per i template di risposta. Si è osservato che, fornendo il contesto completo della conversazione per ogni template, questi venivano compilati in modo molto più preciso, migliorando in modo significativo la qualità e l'affidabilità dei dati salvati. In questa fase, l'utilizzo di tecniche di embedding è stato fondamentale per validare e migliorare la coerenza dei dati estratti.

Sviluppo della Logica di Business

Con a disposizione dati puliti e ben strutturati, è stato possibile procedere con lo sviluppo della logica applicativa. Per questo scopo è stato impiegato il motore di regole Drools, che ha permesso di gestire in modo flessibile le regole di business del progetto, come le diverse tipologie di vacanze e l'applicazione degli sconti corrispondenti.

Risultati e Conclusioni

I risultati ottenuti nella fase di test sono stati molto positivi. L'osservazione più importante è che un *Large Language Model (LLM)* addestrato per un compito specifico (*domain-specific*) ha mostrato *performance* migliori rispetto a modelli concorrenti molto più grandi ma di natura generalista. Questo suggerisce che, almeno per il caso d'uso analizzato, la specializzazione del modello è un fattore più critico per il successo rispetto al numero totale di parametri.

1.3 Organizzazione del testo

Il secondo capitolo descrive l'idea del progetto di stage e le motivazioni che hanno portato a scegliere questo progetto.

Il terzo capitolo Spiega il piano di lavoro, analisi dei rischi e approfondisce le tecnologie scelte.

Il quarto capitolo approfondisce l'analisi dei requisiti, casi d'uso del progetto.

Il quinto capitolo descrive l'implementazione del *ChatBot*, con particolare attenzione all'utilizzo di algoritmi e scelte archittetturali.

Il sesto capitolo approfondisce i risultati ottenuti, le competenze acquisite e difficoltà affrontate.

Nel settimo capitolo conclusione con retrospettiva e proposte di possibili sviluppi futuri.

Riguardo la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

- gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
- per la prima occorrenza dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: api;
- i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*.

Capitolo 2

Processi e metodologie

2.1 Introduzione al progetto

Il progetto è stato definito con un colloquio preliminare con l'azienda, durante il quale è stato delineato come uno studio di fattibilità volto alla realizzazione di un *Chatbot* basato su *Large Language Models (LLM)*. L'obiettivo primario era la conversione di dati non strutturati in linguaggio naturale in un formato strutturato.

La pianificazione delle attività è avvenuta su base settimanale, con ciascuna settimana dedicata a specifiche *task*. Durante il periodo di tirocinio, il piano iniziale è stato integrato con nuove "mini sfide". Questo approccio dinamico ha permesso di approfondire le competenze acquisite e di sfruttare le opportunità di sviluppo di questo progetto per sperimentare ulteriormente, contribuendo a un significativo miglioramento dell'esperienza utente e a un maggiore coinvolgimento.

2.2 Scopo del progetto

Il presente progetto di tesi nasce dall'esigenza di ottimizzare e accelerare i processi di acquisizione di informazioni scritte in linguaggio naturale. L'obiettivo primario è stato quello di sperimentare un sistema automatico in grado di convertire i dati non strutturati di una conversazione testuale in un formato

utile per la loro elaborazione.

Inizialmente, il contesto applicativo immaginato riguardava il settore della ferramenta. L'idea era di snellire la creazione di commesse o fatture a partire da conversazioni testuali con i clienti, dove questi richiedevano specifici articoli come chiodi, viti e altri componenti. Questo approccio mirava a ridurre i tempi di gestione e minimizzare gli errori associati all'interpretazione manuale delle richieste.

Tuttavia, il progetto ha subito un'interessante evoluzione. Su suggerimento ricevuto in fase iniziale di tirocinio, l'ambito di applicazione è stato reindirizzato verso il settore delle agenzie di viaggio. Questa transizione si è rivelata una sfida più complessa e stimolante, offrendo un terreno fertile per esplorare un approccio più dinamico all'estrazione di informazioni. La ricchezza e la variabilità delle richieste relative all'organizzazione di vacanze che spaziano da destinazioni e date, a preferenze di alloggio e attività, hanno permesso di testare il sistema in un ambiente linguistico più articolato e variegato.

Per l'estrazione dei dati, ci siamo affidati a NuExtract 2.0, un *Large Language Model (LLM)* sviluppato da Numind e specificamente addestrato per compiti di *information extraction*. La scelta di NuExtract è stata strategica perché ci ha permesso di implementare un processo di estrazione basato su un meccanismo di *input-output* definito. In pratica, fornivamo al modello un testo in linguaggio naturale insieme a un template schema predefinito (strutturato in *JSON*$_G$) e NuExtract ci restituiva lo stesso template, ma popolato con i dati estratti direttamente dal testo fornito.
Sono stati progettati i template per identificare e catturare con precisione le entità e relazioni più rilevanti per la nostra analisi. La forza di questo approccio risiedeva nella capacità di NuExtract di trasformare e strutturare le informazioni estratte direttamente in formato JSON. Essendo il modello ottimizzato per riconoscere i pattern di interesse e restituire i dati già organizzati in una struttura JSON coerente, abbiamo ottenuto un flusso di lavoro estremamente

efficiente, che ha velocizzato la conversione del linguaggio naturale in un dataset strutturato e pronto all'uso.

Il passaggio del progetto da un chatbot per un'azienda di ferramenta al settore turistico ha rappresentato una sfida decisamente più stimolante. La maggiore complessità delle richieste relative alle vacanze, che spesso implicano una varietà di parametri e preferenze implicite, ha richiesto lo sviluppo di un sistema più robusto e versatile. Tale evoluzione ha permesso di gestire un elevato numero di scenari conversazionali, facilitando la generazione automatica di preventivi o schede di prenotazione.

2.3 Motivazioni alla Base della Scelta Progettuale

L'interesse verso questo progetto nasce da una profonda affinità con il campo dell'intelligenza artificiale. Sin dall'avvento dell'AI, con *ChatGPT*$_G$, nell'uso quotidiano, ho manifestato un forte desiderio di approfondire le tecnologie LLM, un settore dell'informatica in rapidissima evoluzione e con orizzonti ancora da esplorare. Questa dinamicità è proprio ciò che mi attrae.

Il progetto di tesi rappresenta per me un'occasione cruciale per ampliare le mie conoscenze in materia e per apprendere i principi fondamentali per gestire questa nuova e potente risorsa tecnologica. Credo fermamente che l'intelligenza artificiale debba essere considerata un mezzo per raggiungere un fine, e non il fine stesso; uno strumento capace di aumentare la produttività e migliorare la qualità di prodotti e servizi.

L'opportunità di "cavalcare" questa tecnologia emergente, contribuendo seppur in minima parte al suo miglioramento e alla sua comprensione, è per me fonte di grande entusiasmo.

Capitolo 3

Descrizione dello stage

3.1 Pianificazione del lavoro

Il piano di lavoro ha coperto un arco temporale di otto settimane, cosi dipartite:

3.1.1 Settimana 1

Studio preliminare e definizione del sistema	
Obiettivi	• Approfondire il funzionamento di NuExtract • Definire i requisiti del chatbot e del motore di regole • Identificare le fonti dati e strutturare dataset di esempio
Descrizione	Studio di NuExtract e delle sue capacità di estrazione dati. Definizione della struttura del sistema e dei dati necessari per l'integrazione con Drools o libreria simile.
	Continua nella prossima pagina...

Tabella 3.1 – Continuo della tabella

Studio preliminare e definizione del sistema	
Risultati attesi	• Chiarezza sulle funzionalità di NuExtract • Struttura iniziale del sistema definita • Dataset di test pronto per le fasi successive

Tabella 3.1: Settimana 1

3.1.2 Settimana 2

Modellazione della richiesta cliente e recupero dati	
Obiettivi	• Integrare NuExtract per il recupero dati dalle chat • Definire gli intenti principali del chatbot • Strutturare un output JSON per il motore di regole
Descrizione	Sviluppo dell'interazione tra chatbot e NuExtract, con focus sulla comprensione delle richieste dei clienti. Generazione di un JSON strutturato con i dati estratti.
	Continua nella prossima pagina...

Tabella 3.2 – Continuo della tabella

Modellazione della richiesta cliente e recupero dati	
Risultati attesi	• Chatbot capace di estrarre dati utili dalle richieste clienti • JSON strutturato e pronto per Drools o libreria simile • Test preliminari sulla precisione di NuExtract

Tabella 3.2: Settimana 2

3.1.3 Settimana 3

Creazione della struttura per la gestione delle commesse	
Obiettivi	• Definire un modello JSON standard per le commesse • Sviluppare il modulo di conversione dei dati da NuExtract • Integrare la struttura delle commesse con il motore di regole
Descrizione	Progettazione e test della struttura dati JSON per la gestione delle commesse, con verifica della compatibilità con Drools o libreria simile.
	Continua nella prossima pagina...

Tabella 3.3 – Continuo della tabella

Creazione della struttura per la gestione delle commesse	
Risultati attesi	• JSON definitivo per la gestione delle commesse • Integrazione iniziale con Drools • Conversione automatizzata dei dati testata

Tabella 3.3: Settimana 3

3.1.4 Settimana 4

Test e documentazione intermedia	
Obiettivi	• Eseguire test sui moduli implementati nelle prime tre settimane • Documentare l'integrazione tra chatbot, NuExtract e libreria regole • Identificare eventuali problemi e miglioramenti necessari
Descrizione	Durante questa settimana, verranno eseguiti test per verificare il corretto recupero dei dati dalle chat e la generazione del JSON per il motore di regole. Verranno analizzati i primi risultati dei preventivi generati dalla libreria di regole e documentate le problematiche riscontrate.

Continua nella prossima pagina...

Tabella 3.4 – Continuo della tabella

Test e documentazione intermedia	
Risultati attesi	• Verifica del funzionamento base del sistema. • Identificazione di errori o aree da migliorare. • Documentazione aggiornata sui progressi del progetto.

Tabella 3.4: Settimana 4

3.1.5 Settimana 5

Analisi ABC e personalizzazione offerte	
Obiettivi	• Implementare la classificazione clienti con Analisi ABC • Integrare l'analisi storica degli acquisti nel motore di regole • Creare strategie di scontistiche personalizzate
Descrizione	Applicazione dell'Analisi ABC per segmentare i clienti in base al valore economico e personalizzare le offerte nei preventivi.
	Continua nella prossima pagina...

Tabella 3.5 – Continuo della tabella

Analisi ABC e personalizzazione offerte	
Risultati attesi	• Classificazione clienti integrata con il sistema • Personalizzazione delle offerte in base alla classe ABC • Test iniziali sulla coerenza dei risultati

Tabella 3.5: Settimana 5

3.1.6 Settimana 6

Ottimizzazione e test su scenari reali	
Obiettivi	• Eseguire test avanzati su scenari diversi di clienti e ordini • Verificare la qualità delle risposte del chatbot
Descrizione	Test approfonditi del sistema con diversi tipi di richieste per valutare precisione, tempi di risposta e coerenza delle decisioni.
Risultati attesi	• Sistema più stabile e performante • Test superati con dati variabili • Identificazione di miglioramenti da implementare

Tabella 3.6: Settimana 6

3.1.7 Settimana 7

Validazione del sistema	
Obiettivi	• Validare il sistema con test avanzati • Risolvere eventuali bug • Assicurare che l'integrazione tra chatbot, NuExtract e Drools sia stabile
Descrizione	Si eseguiranno test esaustivi per verificare l'affidabilità dell'intero sistema. Saranno raccolti feedback sui risultati dei preventivi generati e si apporteranno le ultime correzioni prima della fase di documentazione finale.
Risultati attesi	• Sistema validato e funzionante • Bug principali risolti

Tabella 3.7: Settimana 7

3.1.8 Settimana 8

Documentazione e conclusioni	
Obiettivi	• Redigere la documentazione tecnica del progetto • Scrivere il report finale con risultati, criticità e miglioramenti possibili • Effettuare un ultimo controllo di qualità sul sistema
Descrizione	Produzione della documentazione tecnica e del report finale.
Risultati attesi	• Documentazione completa e ben organizzata • Report finale pronto per la presentazione • Progetto rifinito e pronto per la consegna

Tabella 3.8: Settimana 8

3.1.9 Ripartizione delle ore

Durata (ore)	Date	Descrizione dell'attività
40	23/04/2025 - 30/04/2025	Studio preliminare di NuExtract e definizione dei requisiti del chatbot e del motore di regole.
40	02/05/2025 - 08/05/2025	Modellazione della richiesta cliente e sviluppo dell'integrazione tra chatbot e NuExtract.
40	09/05/2025 - 15/05/2025	Creazione della struttura JSON per la gestione delle commesse e sviluppo del modulo di conversione dati.
40	16/05/2025 - 22/05/2025	Test e documentazione dei moduli implementati nelle settimane precedenti.
40	23/05/2025 - 29/05/2025	Applicazione dell'Analisi ABC per la classificazione dei clienti e personalizzazione delle offerte.
40	30/05/2025 - 6/06/2025	Test avanzati del sistema su diversi scenari e ottimizzazione delle prestazioni.
40	9/06/2025 - 13/06/2025	Validazione del sistema, risoluzione di bug e verifica dell'integrazione tra chatbot, NuExtract e Drools.
40	16/05/2025 - 20/06/2025	Test finali e documentazione completa del progetto per la consegna.

Tabella 3.9: Ripartizione delle ore per ogni settimana

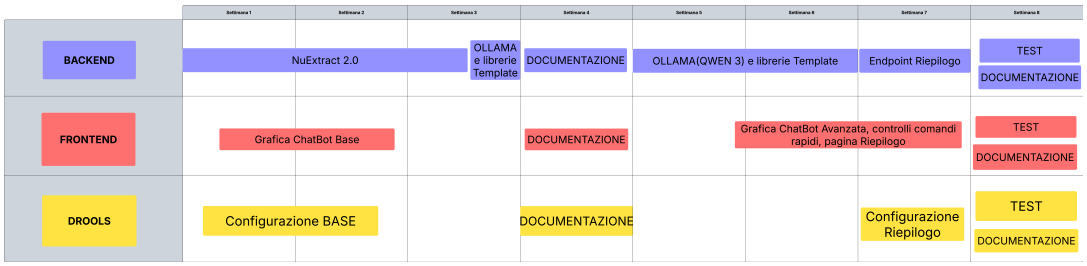

Figura 3.1: Diagramma di GANTT

3.2 Analisi preventiva dei rischi

Durante la fase di analisi iniziale sono stati individuati alcuni possibili rischi a cui si potrà andare incontro. Si è quindi proceduto a elaborare delle possibili soluzioni per far fronte a tali rischi.

1. Performance hardware insufficienti

Descrizione: le performance hardware del portatile possono risultare insufficienti o causare rallentamenti nello sviluppo del software richiesto.

Soluzione: sviluppare il software con LLM di basse dimensioni.

2. Performance del LLM scelto

Descrizione: le performance di un LLM di basse dimensioni possono incidere negativamente nel risultato finale.

Soluzione: studio e test di LLM di piccole dimensioni, con focus su una singola task per migliorare la qualità della risposta ottenuta.

3. Ritardi dovuti alla poca esperienza

Descrizione: Possibili ritardi dovuti alla poca esperienza accumulata per rispettare le scadenze settimanali.

Soluzione: studio, ricerca di soluzioni su forum specifici e richieste ad AI per ottimizzare il tempo e la qualità finale.

4. Problemi di compatibilità con software di terze parti

Descrizione: Possibili problemi riscontrabili quando si integrano più servizi in più linguaggi informatici o LLM.

Soluzione: utilizzo di api per la separazione netta dei processi in più linguaggi; organizzazione del progetto attraverso test separati.

5. Problemi di compatibilità con software o librerie di terze parti
Descrizione: L'utilizzo di software o librerie esterne può comportare malfunzionamenti, incompatibilità e difficoltà di integrazione.
Soluzione: L'adozione di un sistema di versionamento dei file, "backup" regolari e la replica dei dati.

3.3 Obiettivi richiesti

Si farà riferimento ai requisiti secondo le seguenti notazioni:

- **OB** per i requisiti obbligatori, vincolanti in quanto obiettivo primario richiesto dal committente;
- **DE** per i requisiti desiderabili, non vincolanti o strettamente necessari, ma dal riconoscibile valore aggiunto;
- **OP** per i requisiti opzionali, rappresentanti valore aggiunto non strettamente competitivo

Le sigle precedentemente indicate saranno seguite da un numero, identificativo del requisito. Si prevede lo svolgimento dei seguenti obiettivi:

Obbligatorio		Esito
OB 1	Analisi dettagliata della richiesta del cliente	Soddisfatto
OB 2	Definizione e implementazione della struttura di supporto per la gestione delle commesse in formato JSON	Soddisfatto
OB 3	Studio e applicazione dell'algoritmo RETE per l'elaborazione delle regole	Soddisfatto
OB 4	Utilizzo di LLM(NuExtract) per il recupero di dati dalla chat e l'interpretazione delle richieste	Soddisfatto
OB 5	Implementazione di una interfaccia grafica base	Soddisfatto
OB 6	Implementazione di vari template in json schema per la gestione di vacanze personalizzate	Soddisfatto
OB 7	utlizzo del LLM NuExtract nella versione 2.0	Soddisfatto
OB 8	Controlli di sicurezza minimi per raggiungere la pagina di riepilogo senza errori	Soddisfatto
OB 9	Implementazioni di risposte personalizzate per le varie chiavi dei template	Soddisfatto
OB 10	passare tutta la conversazione generata durante la compilazione del template attivo	Soddisfatto
OB 11	la *WebApp*$_G$ deve funzionare per la versione stabile di Chrome: 108.0.5359	Soddisfatto
Desiderabile		**Esito**
DE 1	Implementazione dell'Analisi ABC per la classificazione dei clienti in base agli acquisti	Soddisfatto

Continua nella prossima pagina...

Tabella 3.10 – Continuo della tabella

DE 2	Implementazioni di risposte personalizzate con l'aiuto di un LLM	Soddisfatto
DE 3	Implementazione di una interfaccia grafica avanzata	Soddisfatto
DE 4	Implementazione di sistemi di uscita dalla chat prima di raggiungere la fine della conversazione	Soddisfatto
DE 5	Controlli di sicurezza di buon livello per raggiungere la pagina di riepilogo senza errori	Soddisfatto
Opzionale		**Esito**
OP 1	Valutazione delle prestazioni del sistema con diversi modelli LLM	Soddisfatto
OP 2	Implementazione di un comando rapido (Ctrl + R) per navigare alla pagina riepilogo	Soddisfatto
OP 3	Salvare la sessione tra una pagina e l'altra	Soddisfatto
OP 4	Salvare la sessione anche dopo aver ricaricato la pagina	Soddisfatto

Tabella 3.10: Obiettivi richiesti

3.4 Tecnologie Scelte

In questa sezione vengono analizzate le tecnologie scelte durante lo stage, motivandone la scelta.

3.4.1 Backend

Python

Figura 3.2: Logo Python

Per la gestione degli LLM è stato adottato Python 3.10. Questa scelta è stata motivata dalla sua posizione di standard di settore per lo sviluppo in Intelligenza Artificiale e *Machine Learning*, garantendo accesso a un ecosistema vasto e consolidato di librerie e *framework*, essenziale per la gestione di modelli complessi come gli LLM.

In particolare, la versione 3.10 è la versione stabile di questo progetto, in grado di soddisfare tutte le dipendenze e criticità delle librerie installate.

FastApi e Uvicorn

Figura 3.3: Logo FastApi+Unicorn

FastAPI è un *framework web* in Python davvero in gamba, perfetto per creare API REST in modo rapido ed efficiente. La sua forza sta nel fatto che si basa su standard aperti come OpenAPI, il che significa che ti genera in automatico una documentazione interattiva super utile. Il tutto gira poi su Uvicorn, un server *Asynchronous Server Gateway Interface*$_G$(ASGI) velocissimo, ottimizzato per gestire tante richieste contemporaneamente.

NuExtract 2.0

Figura 3.4: Logo NuExtract

LLM NuExtract è un modello di linguaggio di grandi dimensioni *(Large Language Model*) specializzato, prodotto da NuMind. La sua scelta è stata motivata dall'eccezionale capacità di estrarre dati strutturati, in particolare in formato JSON, a partire da input testuali in linguaggio naturale.

Un fattore determinante nella selezione di NuExtract è la sua architettura e il suo addestramento mirato. Essendo stato specificamente pre-addestrato per questo scopo di *text-to-JSON*, esso riduce drasticamente la necessità di complesse e dispendiose tecniche di *prompt engineering* per ottenere risultati di alta qualità. Questo rappresenta un notevole vantaggio in termini di efficienza dello sviluppo e affidabilità dei risultati. Abbiamo optato per la versione 2.0 di NuExtract, la quale introduce la capacità di elaborare e conformarsi a template definiti da *JSON Schema*. Questa funzionalità non solo standardizza l'output, ma ottimizza ulteriormente la qualità e la coerenza dei dati estratti, riducendo la necessità di post elaborazione.

Tra le sue caratteristiche distintive, spicca la combinazione di dimensioni contenute e notevole precisione. Questa peculiarità lo rende particolarmente

adatto per essere eseguito in ambiente locale, un requisito fondamentale per la gestione di dati sensibili. La possibilità di mantenere l'elaborazione *on-premise* garantisce un elevatissimo livello di *privacy* e sicurezza dei dati, aspetto critico per l'applicazione sviluppata.

Le *performance* di NuExtract, anche in confronto a LLM di ben maggiori dimensioni e complessità, i quali avrebbero richiesto un'ingegnerizzazione dei prompt ben più sofisticata e laboriosa, si sono dimostrate superiori, già dalla sua versione 1.5, NuExtract ha superato le aspettative, confermandosi come la scelta ottimale per le nostre esigenze di estrazione testo-JSON, bilanciando efficacia, sicurezza e scalabilità.

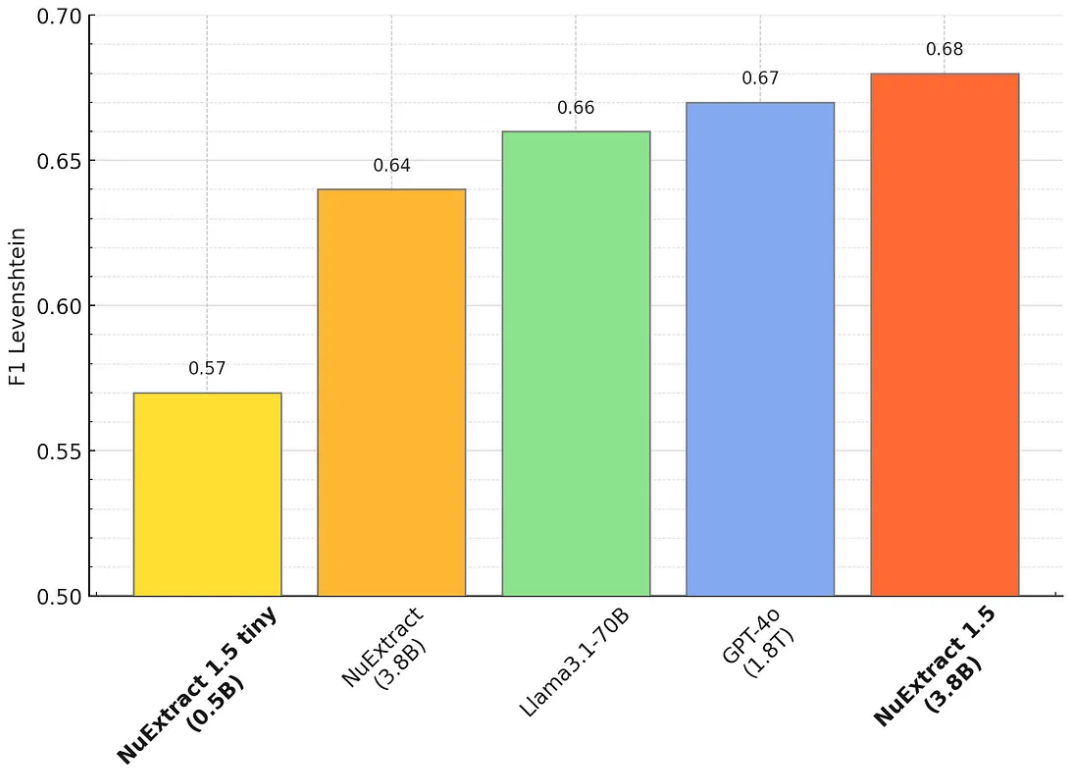

Figura 3.5: Tabella ZeroShot inglese

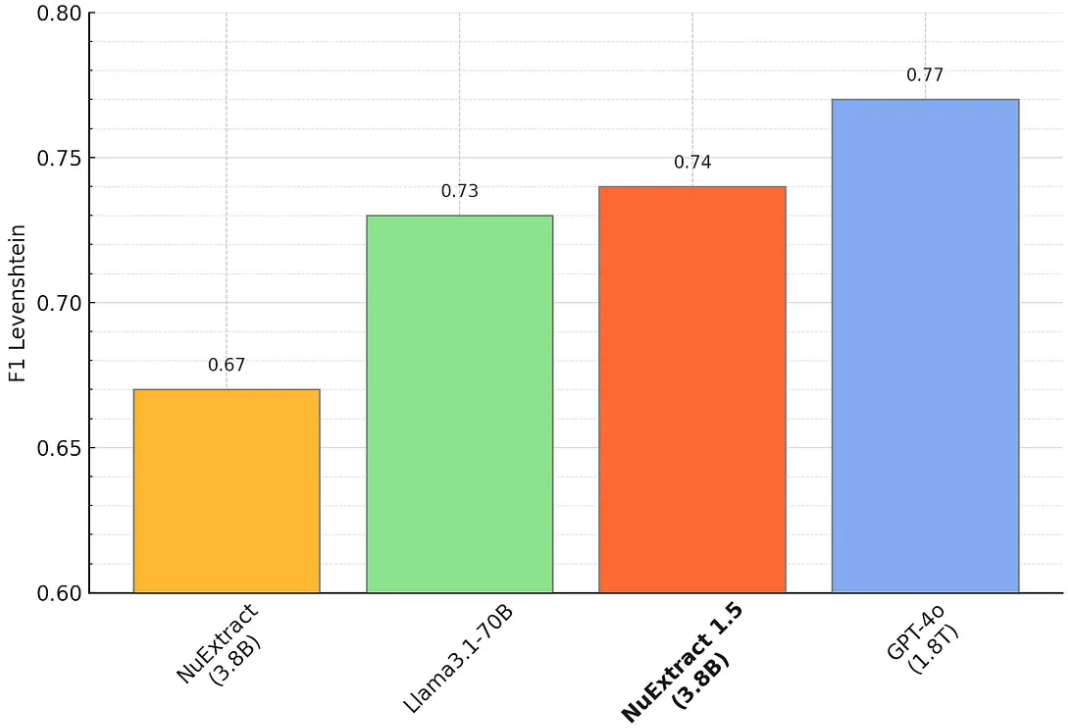

Figura 3.6: Tabella ZeroShot multilingua

Ollama

Figura 3.7: Logo Ollama

Ho scelto Ollama per la gestione e l'esecuzione di *Large Language Models (LLM)* in ambiente locale. Questa piattaforma si è rivelata la soluzione più adatta alle esigenze del progetto, principalmente per la sua straordinaria semplicità d'uso e la sua interfaccia intuitiva, che ne hanno reso l'adozione e l'integrazione nel *workflow* di sviluppo particolarmente agevoli.

Ollama eccelle nel fornire un ambiente "*plug and play*" per scaricare, eseguire e interagire con una vasta gamma di LLM direttamente sulla propria macchina. La sua configurazione minimale e la facilità nel richiamare i modelli via API locale hanno rappresentato un fattore decisivo, permettendo di concentrarsi di più sulla logica applicativa.

Confronti con Alternative Considerate Durante la fase di valutazione, sono state considerate altre piattaforme e framework:

vLLM: Questo framework è stato valutato per le sue notevoli capacità in termini di performance e produttività. vLLM è specificamente progettato per ottimizzare l'inferenza di LLM in scenari che prevedono un elevato numero di richieste in parallelo. Tuttavia, per gli scopi di questa tesi, che prevedono principalmente inferenze sequenziali e gestite singolarmente piuttosto che un'intensa concorrenza di richieste, le sue ottimizzazioni non avrebbero portato un vantaggio significativo. La complessità aggiuntiva nella configurazione e nella gestione di vLLM, rispetto alla semplicità richiesta dal nostro caso d'uso specifico di estrazione dati singola, ha portato a scartarlo in favore di una soluzione più snella.

LM Studio: Anche LM Studio è stato considerato per la sua natura *user-friendly* e la sua capacità di far girare LLM in locale con un'interfaccia grafica intuitiva. Offre un'esperienza simile a Ollama in termini di facilità d'uso per l'esecuzione di modelli. Tuttavia, avendo più famigliarità con Ollama, ho deciso di scartare questa piattaforma.

QWEN 3

Figura 3.8: Logo QWEN 3

Qwen 3:1.7 è un LLM sviluppato da Alibaba Cloud, progettato per generare risposte in linguaggio naturale. La scelta di utilizzare un modello di dimensioni "contenute", invece di uno con miliardi di parametri, nasce da una strategia ben precisa: trovare il giusto equilibrio tra la qualità delle risposte generate e le prestazioni computazionali, fondamentali per garantire un sistema reattivo e veloce.

Il ruolo di Qwen 3:1.7B all'interno del progetto è stato quello di elaborare un prompt strutturato.Questo prompt è stato progettato per essere il più dinamico possibile: chiedo al modello di comportarsi come un agente turistico, gli fornisco la lingua in modo che possa rispondere nello stesso idioma, e gli passo il primo campo libero del template insieme a una frase guida personalizzata. L'obiettivo è supportare al massimo il modello nella generazione della risposta.

Analisi delle Prestazioni e Confronti

Le performance di Qwen 3:1.7B si sono rivelate sorprendentemente sufficienti e all'altezza delle aspettative, soprattutto se si considera le sue piccole dimensioni. La qualità delle risposte generate, in termini di coerenza, pertinenza e fluidità, ha dimostrato che anche un LLM di questa dimensione può gestire compiti complessi di generazione di linguaggio naturale in modo efficace.

Questa scelta non è stata casuale, ma il risultato di un'attenta fase di confronto e selezione. Abbiamo valutato attentamente altri LLM con un range di parametri simile, in particolare Gemma e Llama 3, entrambi nelle versioni comprese tra 1 e 2 miliardi di parametri. Nonostante l'elevata reputazione e le indubbie capacità di questi modelli, i test preliminari hanno evidenziato che le risposte prodotte da Qwen 3:1.7B erano qualitativamente superiori per le nostre specifiche esigenze. Questa superiorità si è manifestata nella capacità di Qwen 3:1.7B di interpretare con maggiore precisione le sfumature del prompt composito e di generare output che richiedevano minimi aggiustamenti, confermando la sua idoneità come motore di generazione per il nostro sistema.

In definitiva, l'adozione di Qwen 3:1.7B ha rappresentato un compromesso ottimale, dimostrando che non sempre un maggior numero di parametri si traduce in una migliore performance per ogni specifica applicazione, specialmente quando l'efficienza e la precisione nel contesto dato sono prioritarie.

Java

Figura 3.9: Logo Java

Java è un linguaggio orientato agli oggetti, il che significa che il codice è organizzato attorno al concetto di "oggetti" che incapsulano dati e comportamenti. Questa struttura promuove la modularità, la riusabilità del codice e una maggiore facilità di manutenzione, aspetti cruciali per lo sviluppo di applicazioni complesse e scalabili. La sua forte tipizzazione e la gestione automatica della memoria (tramite il *Garbage Collector*$_G$) contribuiscono a ridurre gli errori comuni e ad aumentare l'affidabilità del software.

La scelta di Java per alcune componenti del nostro backend è stata fondamentale per l'integrazione di una delle tecnologie chiave che intendevamo esplorare e testare: Drools.
Drools, essendo un potente sistema di gestione delle regole aziendali, è nativamente basato su Java e progettato per operare al meglio nell'ambiente *Java Virtual Machine*$_G$(JVM). L'adozione di Java ci ha quindi permesso di interfacciarci direttamente con questa libreria, sfruttandone appieno le funzionalità senza frizioni di compatibilità.
Per facilitare lo sviluppo e la gestione del progetto, ci siamo avvalsi di strumenti e framework che complementano l'ecosistema Java:
Maven è stato impiegato come sistema di build e gestione delle dipendenze, automatizzando il processo di compilazione, test e *packaging* del codice, oltre a semplificare l'inclusione delle librerie necessarie.
Spring Boot fornisce la struttura dell'applicazione, la gestione delle richieste *HyperText Transfer Protocol*$_G$(HTTP) e l'integrazione con Drools.

Maven

Figura 3.10: Logo Maven

Maven si configura come un potente strumento di *build automation* e gestione delle dipendenze per i progetti basati su Java. La sua funzione è assimilabile a quella di un gestore di pacchetti avanzato, il quale, oltre a scaricare e organizzare automaticamente le librerie necessarie, è preposto anche alle fasi di compilazione, testing e packaging dell'applicazione. Un suo punto di forza è la sua efficacia di standardizzazione, che, attraverso il file "pom.xml", garantisce la

coerenza e la riproducibilità del processo di build su diverse piattaforme. Questa scelta ha fornito una base solida e affidabile per l'intero sviluppo. Ha inoltre facilitato l'integrazione fluida con Spring Boot, per la strutturazione dell'applicazione, e con Drools, per la gestione della logica di business, automatizzando la risoluzione delle dipendenze complesse tra queste tecnologie.

Considerando le alternative, *Gradle* offre maggiore flessibilità e una sintassi più concisa per gli script di build. Tuttavia, risulta molto più complesso da apprendere. In questo progetto, si è preferita la robustezza e la standardizzazione rispetto a un'eccessiva personalizzazione.

Spring Boot e Drools

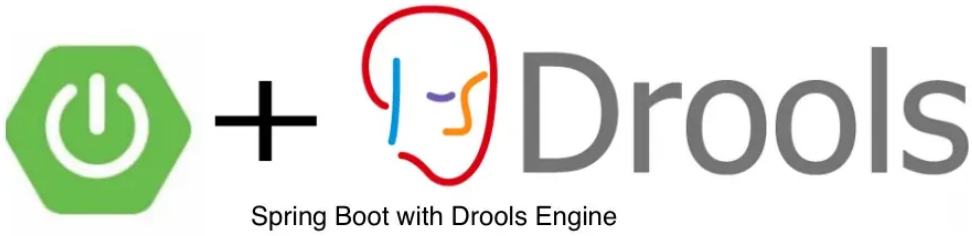

Figura 3.11: Logo Spring Boot e Drools

Spring Boot: Il Framework per lo Sviluppo di Applicazioni *Enterprise* Spring Boot è un framework Java che semplifica e accelera la creazione di applicazioni pronte per l'uso in contesti aziendali *(enterprise-ready)*. Questo approccio riduce notevolmente le configurazioni manuali complesse, permettendo agli sviluppatori di focalizzarsi sulla logica di business.

Nel progetto, Spring Boot si è dimostrato essenziale per gestire vari aspetti, come il server web che espone le API REST del drools-service e la configurazione di Drools.

I principali vantaggi di Spring Boot includono:

Sviluppo Rapido: Grazie all'auto-configurazione intelligente, ai server embedded (come Tomcat) e alle "starter dependencies" preconfigurate, Spring Boot accelera significativamente il ciclo di sviluppo.
Pronto per la Produzione: Fornisce funzionalità integrate per ambienti di produzione, quali metriche, *monitoring* e la capacità di gestire configurazioni esterne e profili specifici per diversi contesti (sviluppo, test, produzione).

Drools è un *Business Rules Management System (BRMS)*, un sistema che consente di separare la logica di business dal codice applicativo. Questa separazione è cruciale per migliorare la manutenibilità e la flessibilità del software, permettendo modifiche alle regole senza dover intervenire sul codice base dell'applicazione.

All'interno di questo progetto, Drools è stato impiegato per gestire la logica specifica relativa al routing turistico, la logica decisionale per i suggerimenti e la validazione dei dati.

I pregi di Drools sono significativi:

Le regole del sistema sono scritte in modo semplice e leggibile e possono essere modificate indipendentemente dal codice.
Performance Elevate: Grazie all'utilizzo di algoritmi efficienti come quello RETE per il *pattern matching*, Drools garantisce prestazioni elevate nell'esecuzione delle regole. Supporta inoltre il *caching* delle regole e l'esecuzione parallela.

3.4.2 Frontend

Node.js

Figura 3.12: Logo Node.js

Node.js è un ambiente di *runtime* JavaScript. È stato utilizzato per l'esecuzione del server locale, per la gestione delle dipendenze tramite *Node Package Manager*$_G$(NPM), e per automatizzare vari processi di build. I suoi principali vantaggi risiedono nell'ampio ecosistema di pacchetti disponibili, che garantisce un eccellente *development workflow*, e nelle elevate *performance* in fase di sviluppo.

Vite

Figura 3.13: Logo Vite

Vite è un moderno strumento di *build e development server* progettato per lo sviluppo di applicazioni web. Si distingue per offrire un'esperienza di sviluppo

estremamente rapida e ottimizzata. Uno dei suoi principali punti di forza è *l'Hot Module Replacement (HMR)*, che permette aggiornamenti quasi istantanei nel browser durante lo sviluppo senza richiedere un ricaricamento completo della pagina. Questo, insieme a tempi di avvio del server di sviluppo quasi immediati e una configurazione minima, lo rende altamente efficiente. Nel progetto, Vite è utilizzato per gestire l'intero processo di sviluppo e *build* dell'applicazione, ottimizzando il codice per la produzione e assicurando un ambiente di sviluppo fluido e reattivo.

React

Figura 3.14: Logo React

Per lo sviluppo dell'interfaccia utente di RouteToWonderland, è stata adottata React (v19), una delle librerie JavaScript più diffuse per la creazione di interfacce utente interattive e dinamiche. React si distingue per l'uso del Virtual *Document Object Model*$_G$(DOM), una rappresentazione leggera del DOM reale che ottimizza le performance, e per l'architettura basata su componenti riutilizzabili. Questo approccio favorisce la modularità del codice e semplifica la manutenzione. Altri vantaggi includono un flusso dati unidirezionale, che rende l'applicazione più prevedibile, e il pieno supporto per gli hooks e i *functional components*, che migliorano la logica dei componenti. Nel progetto, React è il framework principale per la costruzione dell'intera interfaccia, dalla visualizzazione delle mappe ai form di input e all'interazione generale con l'utente.

Per gestire la navigazione all'interno dell'applicazione, è stata integrata *React Router DOM (v7)*. Questa libreria di *routing* permette di definire percorsi dichiarativi e gestire la storia del browser, facilitando la creazione di *Single Page Applications (SPA)* con navigazione dinamica. Supporta inoltre i nested routes e il *code splitting*, contribuendo a migliorare le prestazioni. React Router DOM è impiegato per gestire le transizioni tra le diverse sezioni dell'applicazione, come la pagina principale, quella dedicata alla creazione degli itinerari e la visualizzazione dei percorsi salvati.

3.4.3 Strumenti di sviluppo e supporto

GitHub

Figura 3.15: Logo GitHub

GITHUB è una piattaforma basata sul web che estende le funzionalità di Git, un sistema di controllo versione distribuito. Permette a sviluppatori e team di collaborare allo sviluppo di software in modo efficiente, facilitando la gestione del codice sorgente, il tracciamento delle modifiche e la risoluzione dei conflitti.

Cursor

Figura 3.16: Logo Cursor

Cursor IDE rappresenta un ambiente di sviluppo integrato (IDE) di nuova generazione, specificamente progettato per ottimizzare la produttività dei programmatori attraverso l'integrazione nativa dell'intelligenza artificiale. Basato su Visual Studio Code, Cursor ne estende le funzionalità tradizionali introducendo capacità avanzate di assistenza alla scrittura del codice, debugging e navigazione, direttamente alimentate da modelli linguistici di grandi dimensioni (LLM).

Lucidchart

Figura 3.17: Logo Lucidchart

Lucidchart è una piattaforma di diagrammazione basata sul cloud che consente agli utenti di creare e condividere facilmente diagrammi professionali, come flowchart, diagrammi UML, mappe mentali e schemi di rete. Lucidchart è particolarmente apprezzato per le sue funzionalità collaborative in tempo reale, permettendo a più utenti di lavorare contemporaneamente sullo stesso diagramma e facilitando così la comunicazione e il *brainstorming* visivo all'interno dei team. È ampiamente utilizzato in ambito aziendale, educativo e tecnico per visualizzare processi complessi e migliorare la comprensione.

Microsoft Teams

Figura 3.18: Logo Microsoft Teams

Microsoft Teams è una piattaforma di comunicazione e collaborazione unificata, sviluppata da Microsoft, che integra chat, riunioni video, chiamate e condivisione di file in un'unica interfaccia. Progettata per favorire il lavoro di squadra, permette ai membri di un team di interagire in tempo reale, condividere documenti e gestire progetti in spazi di lavoro dedicati ("team" e "canali").

Capitolo 4

Analisi dei requisiti

4.1 Casi d'uso

Per lo studio dei casi di utilizzo del prodotto sono stati creati dei diagrammi. I diagrammi dei casi d'uso (in inglese *Use Case Diagram*) sono diagrammi di tipo *Unified Modeling Language*$_G$(UML) dedicati alla descrizione delle funzioni o servizi offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono col sistema stesso. Ogni caso d'uso rappresenta uno scenario di utilizzo del programma da parte di un cliente e, oltre alla rappresentazione grafica, sarà correlato dalle seguenti informazioni:

- Descrizione: breve definizione del caso d'uso;
- Attori principali: clienti che eseguono le azioni descritte;
- Precondizioni: serie di condizioni che devono avvenire prima che il cliente svolga l'azione;
- Postcondizioni: stato del programma dopo il caso d'uso;
- Scenario alternativo: caso d'uso alternativo alla postcondizione principale;

4.1.1 Attori

É presente un solo attore, cliente.

- Cliente: utente che interagisce con il chatbot e organizza la vacanza aggiun-

gendo dettagli ad ogni iterazione;

4.1.2 Elenco casi d'uso

UC1: Invio Dati

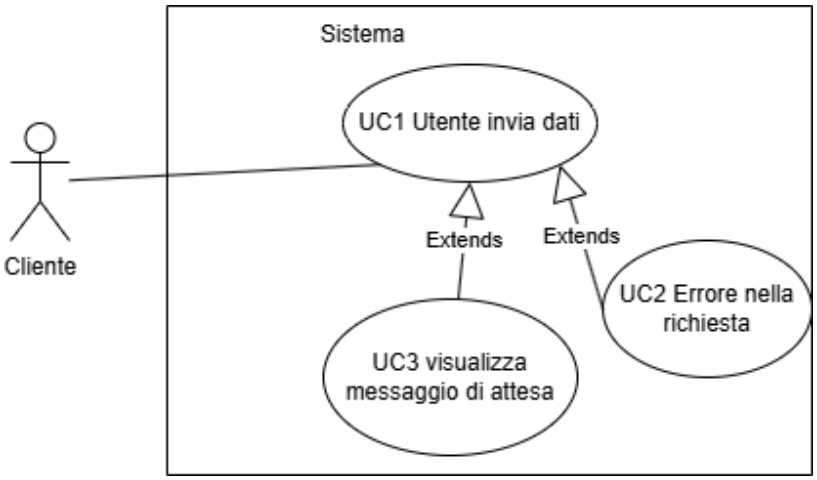

Figura 4.1: Use Case 1: Invio Dati - Errore nella richiesta

Attori Principali: Cliente

Precondizioni: L'utente ha scritto un messaggio nell'input box.

Descrizione: L'utente attraverso l'input box ha inviato al chatbot.

Postcondizioni: UC3 Attesa dell'elaborazione dei dati del chatbot.

Scenario Alternativo:

- UC2 Errore nell'invio dei dati

UC2: Errore nella richiesta

Attori Principali: Cliente

Precondizioni: L'utente ha inviato i dati scritti nell'input box.

Descrizione: L'utente ha ricevuto un messaggio di errore perchè il backend non era ancora pronto a ricevere informazioni.

Postcondizioni: L'utente attendere qualche istante e poi rinvierà la richiesta.

UC3: Attesa risposta

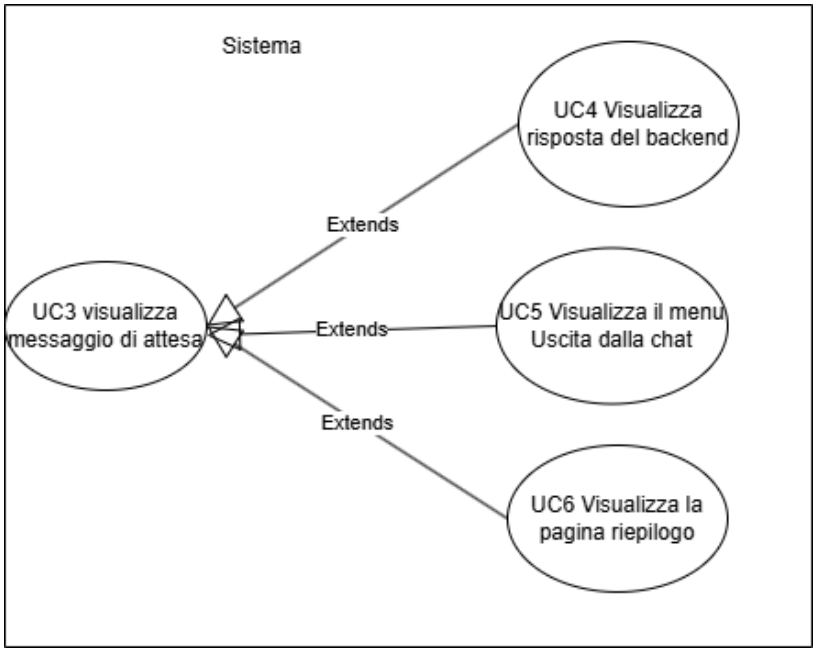

Figura 4.2: Use Case 3: attesa risposta- vari scenari

Attori Principali: Cliente

Precondizioni: L'utente invia dei dati al chatbot.

Descrizione: L'utente riceve un messaggio dal chatbot, che varia in base alla situazione in cui ci si trova.

Postcondizioni: UC4 visualizza risposta del backend.

Scenario Alternativo:

- UC5 visualizza il menu di uscita.
- UC6 visualizza pagina riepilogo.

UC4: Risposta del Backend

Attori Principali: Cliente

Precondizioni: L'utente ha inviato i dati scritti nell'input box.

Descrizione: L'utente ha ricevuto un messaggio dal chatbot con una richiesta di informazioni per riempire il prossimo campo del template.

Postcondizioni: L'utente invierà delle informazioni al chatbot.

UC5: Menu di uscita

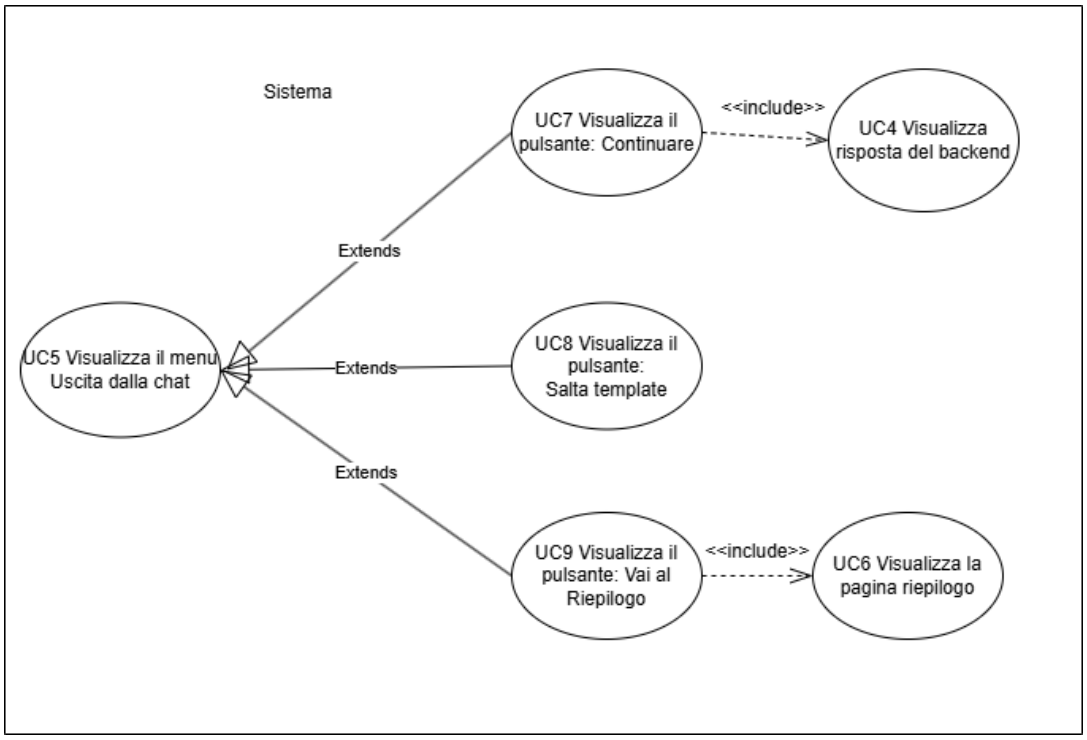

Figura 4.3: Use Case 5: menu di uscita

Attori Principali: Cliente

Precondizioni: il chatbot non ha riempito nessun campo nelle precedenti due iterazioni col cliente o c'è una richiesta di uscita dalla chat. I template obbligatori sono soddisfatti e completi.

Descrizione: Appare un menu con dei bottoni per scegliere le prossime azioni da compiere.

Postcondizioni: L'utente sceglierà l'azione più adeguata.

Scenario Alternativo:

- UC7 Pulsante continua.
- UC8 Pulsante salta Template.
- UC9 Pulsante riepilogo.

UC7: Pulsante Continua

Attori Principali: Cliente

Precondizioni: il chatbot ha visualizzato il menu di uscita e l'utente ha selezionato il pulsante continua.

Descrizione: il chatbot invia una domanda per ricevere informazioni su un campo vuoto del template.

Postcondizioni: L'utente invierà delle informazioni, inclusione del UC4.

UC8: Salta Template

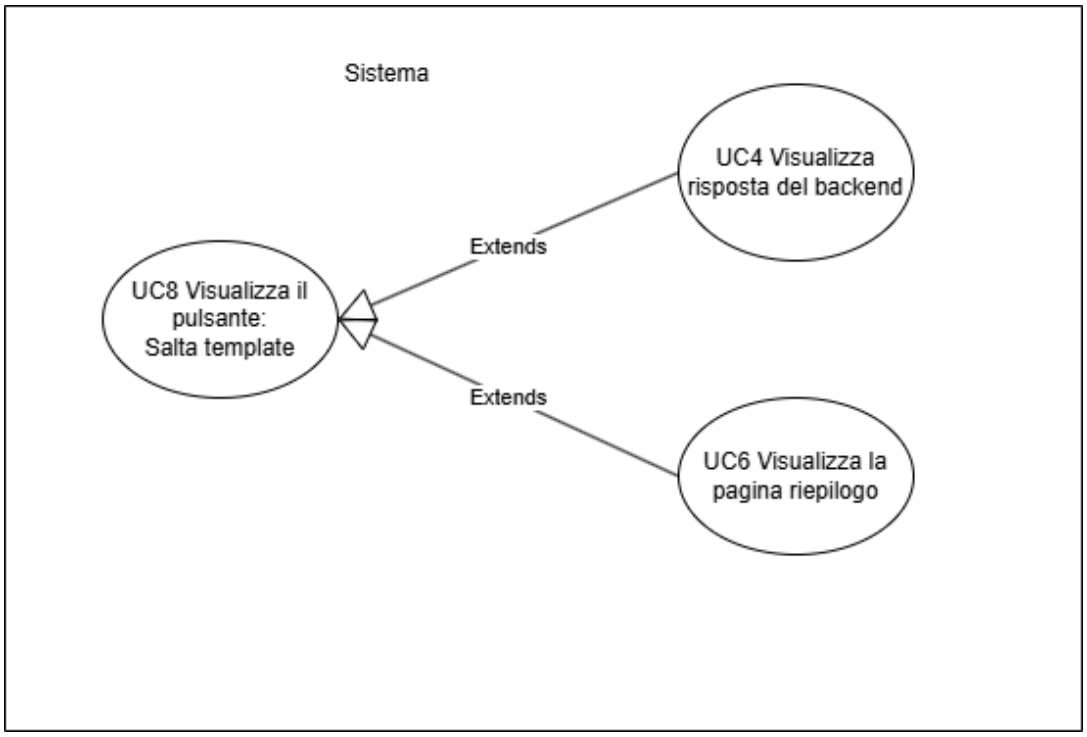

Figura 4.4: Use Case 8: Salta Template

Attori Principali: Cliente

Precondizioni: il chatbot non ha riempito nessun campo nelle precedenti due iterazioni col cliente o c'è una richiesta di uscita dalla chat.

Descrizione: Appare un menu con dei bottoni per scegliere le prossime azioni da compiere.

Postcondizioni: L'utente sceglierà l'azione più adeguata.

Scenario Alternativo:

- UC4 riceverà una risposta dal backend.
- UC6 reindirizzato alla pagina Riepilogo.

UC9: Pulsante Riepilogo

Attori Principali: Cliente

Precondizioni: il chatbot ha visualizzato il menu di uscita e l'utente ha selezionato il pulsante riepilogo.

Descrizione: l'utente verrà rinderizzato alla pagina riepilogo.

Postcondizioni: L'utente si ritroverà nel use case UC6.

UC10: Scorciatoia

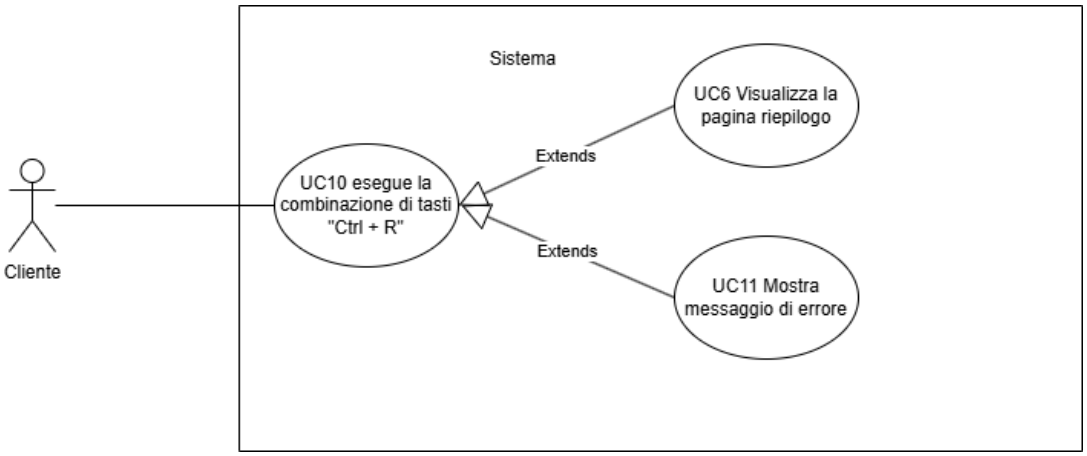

Figura 4.5: Use Case 10: Scorciatoia: Ctrl + R

Attori Principali: Cliente

Precondizioni: L'utente ha effettuato una pressione simultanea dei tasti 'Ctrl' e 'R'.

Descrizione: combinazione di tasti per eseguire una scorciatoia per raggiungere la pagina Riepilogo.

Postcondizioni: l'utente sarà reindirizzato alla pagina Riepilogo UC6.

Scenario Alternativo:

- UC11 errore scorciatoia.

UC11: Errore Scorciatoia

Attori Principali: Cliente

Precondizioni: L'utente preme la combinazione di tasti per la scoriciatoia, prima di aver concluso i template obbligatori.

Descrizione: L'utente riceve un messaggio di errore, nel quale viene specificato che deve prima concludere i template obbligatori.

Postcondizioni: L'utente potrà inserire altre informazioni nella conversazione.

UC6: Pagina Riepilogo

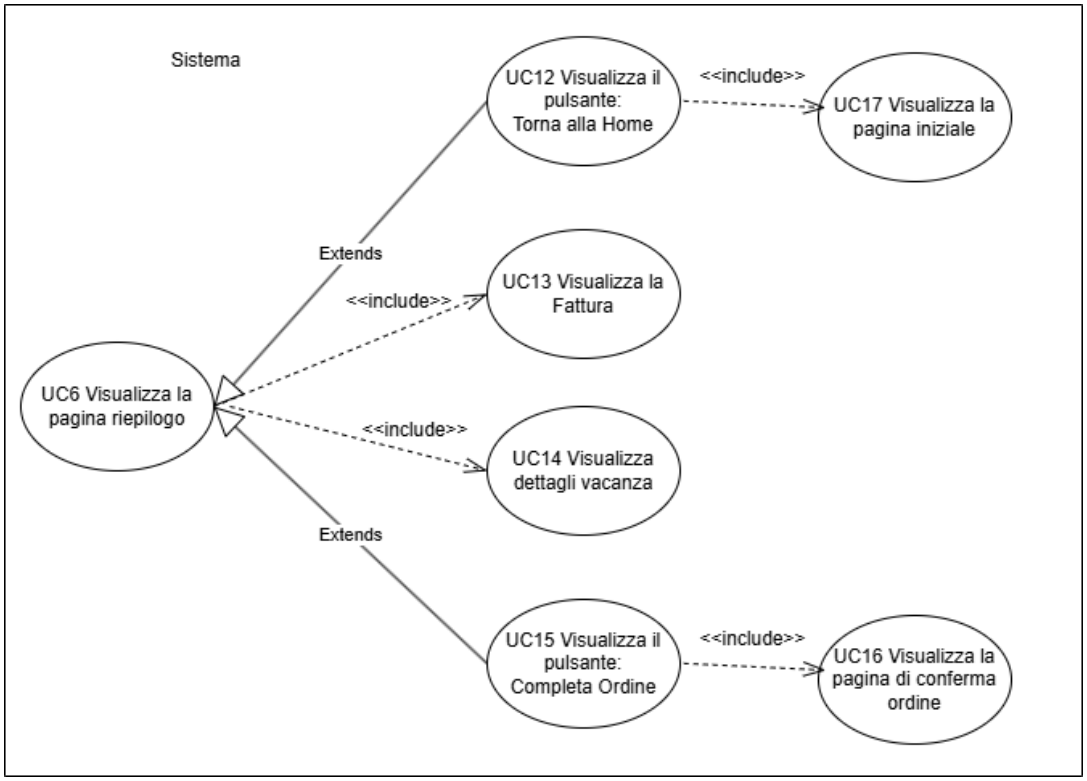

Figura 4.6: Use Case 6: Pagina Riepilogo

Attori Principali: Cliente

Precondizioni: L'utente soddisfa i template obbligatori.

Descrizione: visualizza i dettagli della sua vacanza, la fattura.

Postcondizioni: l'utente confermerà l'ordine UC16.

Scenario Alternativo: UC17 ritorna alla pagina principale.

UC16: Conferma ordine

Attori Principali: Cliente
Precondizioni: L'utente preme il tasto "conferma dell'ordine".
Descrizione: L'utente viene reindirizzato alla pagina di conferma ordine e ringraziamenti.

UC17: Torna alla Home

Attori Principali: Cliente
Precondizioni: L'utente preme il tasto "Torna alla Home".
Descrizione: L'utente viene reindirizzato alla pagina iniziale.
Postcondizioni: L'utente può visualizzare la chat.

4.2 Tracciamento dei requisiti

Da un'attenta analisi dei requisiti e degli *use case* effettuata sul progetto è stata stilata la tabella che traccia i requisiti in rapporto agli *use case*.
Sono stati individuati diversi tipi di requisiti e si è quindi fatto utilizzo di un codice identificativo per distinguerli.
Il codice dei requisiti, dove ogni requisito è identificato con il carattere **R**, è così strutturato:

R[Tipo].[Importanza].[Codice]

dove:

- **Codice:** un numero progressivo a tre cifre che parte dal numero 001.
- **Tipo:** la tipologia di requisito che può essere tra le seguenti:
  - **F (funzionale):** Una funzione di sistema descrive il modo in cui un sistema utilizza determinati ingressi per generare specifiche uscite,

seguendo una logica o una regola prestabilita che definisce il suo comportamento.

  - **Q (qualità):** definisce le caratteristiche di qualità del prodotto software.
  - **V (vincolo):** specifica i limiti e le restrizioni imposte dal capitolato, che il prodotto software deve rispettare.

- **Importanza:** la priorità viene assegnata al requisito con:
  - **O**: requisito obbligatorio che deve essere soddisfatto per la realizzazione del prodotto software;
  - **D**: requisito desiderabile il cui soddisfacimento è apprezzato dal committente;
  - **P**: requisito facoltativo, la cui realizzazione è totalmente a discrezione del team in base all'andamento del progetto.

Nelle tabelle 4.1, 4.2 e 4.3 sono riassunti i requisiti e il loro tracciamento con gli use case delineati in fase di analisi.

4.3 Tabelle dei requisiti

Requisito	Descrizione	Fonte
RF.O.01	L'utente deve poter scrivere nella chat	UC1
RF.O.02	L'utente deve poter inviare al sistema le informazioni scritte	UC1
RF.O.03	L'utente deve poter visualizzare un messaggio di errore nel caso in cui i dati non sono stati ricevuti correttamente dal sistema	UC2
RF.O.04	L'utente deve poter visualizzare un messaggio di attesa della risposta	UC3

Continua nella prossima pagina...

Tabella 4.1 – Continuo della tabella

Requisito	Descrizione	Fonte
RF.O.05	L'utente deve poter visualizzare un messaggio/richiesta dal sistema dopo l'attesa	UC3
RF.O.06	Il messaggio del sistema deve essere una richiesta di informazioni per un campo specifico del template	UC4
RF.O.07	il sistema deve far visualizzare un menu con tre richieste se l'utente non inserisce informazioni in due iterazioni di fila	UC5
RF.D.08	il sistema deve far visualizzare un menu con tre richieste se l'utente fa una richiesta di uscita dalla chat	UC5
RF.O.09	il sistema deve portare alla pagina di riepilogo se tutti i template sono completi	UC6
RF.O.10	il sistema deve far visualizzare il pulsante "Continuare"	UC5
RF.O.11	il sistema deve far visualizzare il pulsante "Salta Template"	UC5
RF.O.12	il sistema deve far visualizzare il pulsante "Vai al Riepilogo"	UC5
RF.O.13	L'utente deve poter prendere il pulsante "Continuare" e visualizzare una nuova richiesta di informazioni del campo da riepire	UC7
RF.O.14	L'utente deve poter prendere il pulsante "Salta Template" e visualizzare una nuova richiesta di informazioni del primo campo disponibile del template successivo	UC8

Continua nella prossima pagina...

Tabella 4.1 – Continuo della tabella

Requisito	Descrizione	Fonte
RF.O.15	L'utente deve poter prendere il pulsante "Salta Template" e visualizzare la pagina riepilogo se non ci sono altri template da completare	UC8
RF.O.16	L'utente deve poter prendere il pulsante "Vai al Riepilogo" e visualizzare la pagina riepilogo	UC9
RF.O.17	L'utente deve aver completato i template obbligatori per usare questo comando	UC10
RF.O.18	Il Sistema mostra un errore se non sono stati completati i template obbligatori	UC11
RF.O.19	L'utente deve poter visualizzare la pagina riepilogo se ha completato i template obbligatori e ha premuto la scorciatoia	UC10
RF.O.20	L'utente nella pagina Riepilogo deve poter visualizzare la fattura	UC13
RF.O.21	L'utente nella pagina Riepilogo deve poter visualizzare i dettagli della vacanza	UC14
RF.O.22	L'utente nella pagina Riepilogo deve poter visualizzare e cliccare il pulsante "Torna alla home"	UC12
RF.O.23	L'utente dopo aver cliccato il pulsante "Torna alla home" deve tornare alla pagina principale	UC17
RF.O.24	L'utente nella pagina Riepilogo deve poter visualizzare e cliccare il pulsante "Completa Ordine"	UC15

Continua nella prossima pagina...

Tabella 4.1 – Continuo della tabella

Requisito	Descrizione	Fonte
RF.O.25	L'utente dopo aver cliccato il pulsante "Completa Ordine" deve andare alla pagina di conferma ordine	UC18
RF.O.26	Il sistema deve dare delle domande al cliente per raccogliere informazioni	PdL
RF.D.27	Il sistema deve dare delle domande personalizzate tramite un LLM al cliente per raccogliere informazioni	Interno
RF.O.28	Il sistema deve raccogliere le informazioni del cliente attraverso LLM "NuExtract"	PdL
RF.P.29	Il sistema deve raccogliere le informazioni del cliente attraverso altri LLM	PdL
RF.O.30	Il sistema deve elaborare le informazioni attraverso il motore di regole DROOLS e algoritmi di machine learning RETE	PdL
RF.O.31	Il sistema deve elaborare una fattura dai dati ricevuti dal cliente	PdL

Tabella 4.1: Tabella del tracciamento dei requisiti funzionali.

Requisito	Descrizione	Fonte
RQ.O.01	Il programma deve avere una interfaccia grafica base	interno
RQ.D.02	Il programma deve avere una interfaccia grafica avanzata	interno
RQ.O.03	Il programma deve avere una interfaccia responsive	interno
		Continua nella prossima pagina...

Tabella 4.2 – Continuo della tabella

Requisito	Descrizione	Fonte
RQ.O.04	Il sistema deve funzionare offline	PdL
RQ.O.05	Il sistema deve funzionare in locale	PdL
RQ.O.06	Il sistema deve garantire la sicurezza dei dati	PdL
RQ.O.07	Il sistema deve scrivere in italiano	interno
RQ.D.08	Il sistema deve supportare più lingue	interno

Tabella 4.2: Tabella del tracciamento dei requisiti qualitativi.

Requisito	Descrizione	Fonte
RV.O.01	Il codice sorgente del backend deve essere presente in una repository GitHub	Interno
RV.O.02	Il codice sorgente del frontend deve essere presente in una repository GitHub	Interno
RV.D.03	Il modello AI (LLM) deve essere Open Source	Interno
RV.O.04	La licenza per i sorgenti dovrà essere Open Source	Interno
RV.O.05	L'applicazione deve essere compatibile con il browser Edge dalla versione 94.0.992.31.	Interno
RV.O.06	L'applicazione deve essere compatibile con il browser Chrome dalla versione 108	Interno
RV.O.07	L'applicazione deve essere compatibile con il browser Firefox dalla versione 109	Interno

Tabella 4.3: Tabella del tracciamento dei requisiti di vincolo.

Capitolo 5

Progettazione e codifica

In questo capitolo vengono analizzate le decisioni architetturali e tecnologiche che hanno guidato lo sviluppo del sistema. La scelta è ricaduta su un'architettura a tre livelli, nota come *3-Tier Architecture*, che permette di separare in modo efficace i diversi ambiti funzionali dell'applicazione: la logica di presentazione, la logica di business e la gestione delle regole complesse.

5.1 Struttura a Tre Livelli

L'architettura è suddivisa nei seguenti tre componenti principali, ognuno con un proprio *stack* tecnologico e un perimetro di responsabilità ben definito.

5.1.1 Livello di Presentazione: Frontend

Figura 5.1: inizio di Chat

Il frontend, ovvero l'interfaccia con cui l'utente interagisce, è stato sviluppato utilizzando la libreria React. La sua responsabilità è limitata alla visualizzazione dei dati e alla cattura degli input utente. Per la gestione della navigazione

tra le diverse sezioni dell'applicazione si è utilizzato *React Router*, mentre la gestione dello stato globale è stata affidata alla *Context API*, una soluzione integrata in React che consente di evitare l'introduzione di dipendenze esterne più complesse. L'interazione con il backend avviene tramite richieste asincrone a un'API RESTful.

5.1.2 Livello Applicativo: Backend

Il backend è il nucleo del sistema, dove risiede la logica applicativa principale. È stato implementato in Python utilizzando FastAPI, un framework moderno e performante per la creazione di API. Questo livello gestisce le richieste provenienti dal frontend, interagisce con il database per la persistenza dei dati e implementa la logica di business. Internamente, il backend è organizzato secondo un approccio monolitico modulare, che consente di mantenere il codice ordinato e coeso pur rimanendo all'interno di un'unica base di codice. Questa scelta rappresenta un compromesso strategico tra la semplicità di gestione di un sistema monolitico classico e l'organizzazione del codice tipica delle architetture a microservizi.

Caratteristiche Principali L'approccio del monolite modulare si distingue per le seguenti caratteristiche:

- Singolo *Deployment*: L'intera applicazione backend viene distribuita come un unico servizio. Non esistono processi indipendenti per le diverse funzionalità, come avverrebbe in un'architettura a microservizi. Questo semplifica notevolmente le operazioni di rilascio e gestione dell'infrastruttura.

- Separazione Logica delle Responsabilità: Nonostante sia un'unica unità, il codice è strutturato in *directory* e moduli distinti. Ogni modulo è specializzato in un compito preciso. Ad esempio, un modulo gestisce gli *endpoint* API, un altro la comunicazione con servizi esterni, e un altro ancora la logica di business specifica.

- Comunicazione Interna Diretta: La comunicazione tra i vari moduli avviene tramite meccanismi interni al linguaggio di programmazione, come chiamate di funzione dirette o *import* di classi e metodi. Non è necessaria alcuna comunicazione di rete (es. chiamate HTTP), riducendo la latenza e la complessità.

- Scalabilità Verticale: Per far fronte a un aumento del carico di lavoro, l'approccio principale consiste nel potenziare le risorse della macchina su cui il servizio è in esecuzione (aumentando CPU, RAM, ecc.). Questo modello è noto come scalabilità verticale e si contrappone alla scalabilità orizzontale dei microservizi, dove si aggiungono istanze multiple di un singolo servizio.

Vantaggi e Svantaggi

La scelta di un monolite modulare comporta un'analisi dei seguenti pro e contro:

Vantaggi

- Manutenibilità Elevata: L'organizzazione logica del codice rende il sistema più leggibile, facile da comprendere e modificare. Ogni modulo ha una responsabilità chiara, riducendo il rischio di introdurre effetti collaterali indesiderati.

- Sviluppo Rapido: L'assenza di comunicazione di rete tra i componenti interni elimina la complessità legata alla gestione di API, serializzazione dei dati e fallimenti di rete, accelerando il ciclo di sviluppo.

- Test Semplificati: È possibile eseguire test unitari e di integrazione sui singoli moduli in modo isolato, senza la necessità di avviare e orchestrare un ambiente con più servizi distribuiti.

Svantaggi

- Scalabilità Limitata: Se una specifica funzionalità (ad esempio, l'elaborazione di immagini) diventa un collo di bottiglia, non è possibile scalare orizzontalmente solo quel modulo. È necessario scalare l'intera applicazione, con un potenziale spreco di risorse.

- *Deployment* "Tutto o Niente": Poiché l'applicazione è un blocco unico, un errore critico in un modulo minore può potenzialmente causare il fallimento dell'intero servizio, impattando tutte le funzionalità.

Il monolite modulare è una soluzione architetturale che offre un eccellente equilibrio per progetti di piccole e medie dimensioni, garantendo un codice ben organizzato e manutenibile senza introdurre la complessità operativa tipica dei sistemi a microservizi.

Livello di Business Logic: Drools Service

Per la gestione delle regole di business più complesse e dinamiche, si è optato per l'integrazione di un servizio esterno basato su Drools. Questo motore di regole (*Business Rule Engine*) è un componente specializzato, tipicamente basato su Java, che ha il solo compito di valutare un set di fatti (dati) rispetto a un insieme di regole predefinite. Il backend comunica con questo servizio tramite API REST, delegando le decisioni complesse e mantenendo così il proprio codice più snello e focalizzato sulla logica principale.

5.2 Algoritmi e Metodi rilevanti

5.2.1 Estrazione di Dati Strutturati con NuExtract 2.0

La trasformazione dell'input utente da linguaggio naturale a un formato strutturato e processabile è un compito affidato al modello LLM specializzato NuExtract 2.0. Questo modello è stato selezionato per la sua alta efficienza nel compito specifico di *Information Extraction*, ovvero la capacità di identificare e convertire dati testuali (o presenti in immagini) in un formato JSON predefinito.

Per massimizzare l'accuratezza, al modello non viene fornita solo l'ultima interazione, ma l'intera cronologia della conversazione avvenuta tra l'utente e il bot, a partire dal momento in cui un template di raccolta dati viene attivato. Questo approccio permette a NuExtract 2.0 di avere una visione completa del dialogo, consentendogli di gestire correttamente eventuali correzioni o ripensamenti da parte dell'utente. In questo modo, il JSON finale riflette fedelmente le

aspettative consolidate dell'utente, garantendo la coerenza con la struttura del template.

Funzione di estrazione dei dati dal testo:

```
def extract_entities(text, template):
        # prepare the user message content
    messages = [{"role": "user", "content": text}]
    text = processor.tokenizer.apply_chat_template(
        messages,
        template=template,
        tokenize=False,
        add_generation_prompt=True,
    )
    image_inputs = process_all_vision_info(messages)
    inputs = processor(
        text=[text],
        images=image_inputs,
        padding=True,
        return_tensors="pt",
    ).to("cpu")
    generation_config = {"do_sample": False, "num_beams": 1,
     ↪  "max_new_tokens": 2048}
    output_text = processor.batch_decode(
        generated_ids_trimmed, skip_special_tokens=True,
         ↪  clean_up_tokenization_spaces=False
    )
    print(output_text)
        return output_text
```

Codice 5.1: Funzione di estrazione dei dati dal testo

5.2.2 Generazione di Risposte Dinamiche con Qwen 3

Per la componente di generazione del linguaggio naturale (NLG), ovvero la formulazione delle risposte del bot, è stato impiegato il modello Qwen. Anche in questo caso, si tratta di un modello di dimensioni contenute, la cui efficacia è stata massimizzata attraverso un'attenta ingegnerizzazione del prompt (*prompt engineering*).

Per guidare il modello a produrre una risposta pertinente e contestualmente adeguata, il prompt fornito a Qwen è stato strutturato per includere tre elementi chiave:

- Una frase guida: Un'istruzione che suggerisce come sollecitare l'informazione successiva, tipicamente legata al primo campo del template non ancora compilato.
- Il contesto linguistico: L'indicazione esplicita della lingua della conversazione (es. italiano), per assicurare coerenza stilistica e grammaticale.
- Lo stato della conversazione: Il template JSON, compilato con i dati raccolti fino a quel momento, che fornisce al modello un quadro completo delle informazioni già acquisite.

Questa strategia di *prompting* consente a Qwen di generare risposte che non sono solo naturali e fluide, ma anche strategicamente mirate a proseguire la raccolta dati in modo efficace, migliorando l'esperienza utente complessiva.

Funzione per la prossima domanda all'utente:

```
prompt = f"""Sei un assistente di viaggio esperto e cordiale. Il
↪  tuo compito è comunicare all'utente che ci sono ancora
↪  domande obbligatorie da completare.

IMPORTANTE:
- La risposta deve essere: "{messaggio}"
- La risposta deve essere in {lingua_sistema}
- Non modificare la frase fornita
- Non aggiungere altro testo

Ora, formula la tua risposta:"""

            response = requests.post(
                f"{self.base_url}/generate",
                json={
                    "model": self.model_name,
                    "prompt": prompt,
                    "stream": False,
                    "options": {
                        "temperature": 0.7,
                        "top_p": 0.9,
                        "top_k": 40,
                        "system": f"Sei un assistente di viaggio
                        ↪  esperto e cordiale. Rispondi sempre
                        ↪  in {lingua_sistema}."
                    }
                }
            )
    return response
```

Codice 5.2: Prossima domanda all'utente

5.2.3 Correzione e Normalizzazione dei Dati tramite Embedding Semantico

Per garantire l'integrità dei dati raccolti è la loro normalizzazione. I modelli linguistici di piccole dimensioni, pur essendo computazionalmente efficienti, possono manifestare limitazioni nell'interpretazione di termini specifici, generando talvolta varianti o errori ortografici. Tale fenomeno compromette l'efficacia delle interrogazioni esatte verso un database, dove la corrispondenza letterale è un requisito fondamentale.

Per ovviare a questa criticità, è stato implementato un meccanismo di correzione basato su *embedding*. Questa tecnica consiste nel proiettare le parole in uno spazio vettoriale multidimensionale, dove la distanza tra i vettori rappresenta la similarità semantica tra i termini. Il processo operativo è il seguente:

- Un termine estratto dal modello viene convertito nel suo rispettivo vettore di embedding.
- Questo vettore viene confrontato con un dizionario pre-calcolato di vettori corrispondenti a un insieme di "parole valide" e canoniche.
- Viene ricercata la parola valida il cui vettore ha la distanza minore rispetto al vettore del termine estratto.

Per evitare associazioni errate, è stata definita una soglia di distanza massima, pari a 0.4. Questo garantisce che la sostituzione avvenga solo in caso di alta confidenza e similarità, preservando così l'accuratezza delle informazioni e rendendo le ricerche successive affidabili.

Capitolo 6

Competenze Maturate e Criticità Affrontate nel Progetto

6.1 Risultati ottenuti

I risultati conseguiti al termine del tirocinio sono pienamente allineati con gli obiettivi prefissati. È stato implementato con successo un sistema conversazionale avanzato, capace di guidare un utente dalla creazione di un pacchetto vacanze fino alla generazione della relativa fattura. In corso d'opera, il progetto è stato arricchito dall'integrazione di un secondo *Large Language Model*, una scelta non prevista inizialmente ma rivelatasi di grande valore. Questo LLM è stato dedicato specificamente alla gestione di risposte dinamiche e naturali, elevando notevolmente la qualità dell'interazione. Tale miglioramento si è dimostrato particolarmente strategico per il contesto simulato di un'agenzia di viaggi, dove la fidelizzazione del cliente attraverso un dialogo efficace rappresenta un fattore cruciale di successo.

Figura 6.1: riepilogo costi

Figura 6.2: riepilogo della vacanza

Figura 6.3: riepilogo della vacanza

Figura 6.4: riepilogo della vacanza

Figura 6.5: riepilogo della vacanza

Figura 6.6: riepilogo della vacanza

6.2 Competenze acquisite

Questa esperienza di tirocinio ha rappresentato un'importante opportunità di crescita, permettendomi di consolidare competenze trasversali e tecniche. Ho affinato in modo significativo le mie capacità di problem solving analitico e le metodologie di ricerca della documentazione, soprattutto nell'affrontare le complesse sfide legate alla persistenza, coerenza e validazione dei dati. Ho inoltre acquisito una comprensione pratica e approfondita delle moderne tecnologie di Intelligenza Artificiale, imparando a configurare e interagire con i LLM. L'esperienza mi ha permesso anche di gestire in autonomia l'intero ciclo di vita di un problema tecnico: dalla sua identificazione e analisi, fino all'implementazione di una soluzione funzionale ed efficiente in un contesto innovativo.

6.3 Difficoltà affrontate

Durante il percorso sono state affrontate diverse sfide tecniche che hanno richiesto un approccio metodico. La difficoltà principale ha riguardato la configurazione dell'ambiente di esecuzione per i *Large Language Models*. Si sono presentate significative incompatibilità di versione tra le librerie software più recenti e le dipendenze di alcuni modelli, con metodi e funzioni deprecati che hanno richiesto tempo per il debugging. Un'ulteriore sfida, essendo un primo approccio a un'architettura complessa, è stata quella di stabilire una comunicazione API stabile e affidabile tra i diversi servizi. Infine, la gestione delle inconsistenze e la validazione dei dati estratti dai modelli si sono rivelate un ostacolo significativo, che ha reso necessaria l'implementazione di procedure di controllo per assicurare l'integrità dell'output finale.

Capitolo 7

Conclusioni

7.1 Possibili sviluppi futuri dell'applicazione

Il lavoro svolto pone solide basi per numerose evoluzioni del progetto. Durante la fase di sviluppo sono state identificate diverse aree di potenziale miglioramento che potrebbero aumentare significativamente le funzionalità e l'intelligenza dell'applicazione. Di seguito vengono delineate le direzioni più promettenti.

- **Estrazione di Entità con Schemi Dinamici** Un'evoluzione consisterebbe nel superare l'attuale sistema basato su template statici per l'acquisizione delle informazioni. L'obiettivo è impiegare un Large Language Model per generare e gestire dinamicamente lo schema dei dati da estrarre in base alla conversazione. Ad esempio, il modello potrebbe decidere autonomamente quali campi sono pertinenti (es. "numero di bambini", "età animali") a seconda dell'intento specifico dell'utente. Questo approccio renderebbe l'interazione più fluida e personalizzata, eliminando la rigidità dei moduli predefiniti e migliorando l'efficienza del processo di raccolta dati.

- **Sincronizzazione nel Calendario** Il sistema potrebbe essere potenziato per consentire la sincronizzazione della prenotazione con i calendari personali dell'utente (es. Google Calendar, Outlook), automatizzando l'aggiunta dell'evento e dei relativi promemoria tramite l'utilizzo delle opportune API.

7.2 Retrospettiva

L'esperienza di tirocinio in Zucchetti S.P.A. ha rappresentato un'opportunità di crescita di grande valore, sia a livello professionale che personale. Questo percorso mi ha permesso non solo di acquisire nuove competenze pratiche, ma anche di approfondire ambiti tecnologici all'avanguardia e di grande interesse, come quello dei *Large Language Models (LLM)*.

Uno degli aspetti più significativi è stato poter osservare da vicino il ciclo di vita di un prodotto software all'interno di un contesto aziendale strutturato. Ho compreso l'importanza cruciale della gestione del tempo e la necessità di trovare un equilibrio costante tra la qualità del lavoro e le scadenze prefissate. Questa consapevolezza mi ha spinto a ottimizzare le strategie di sviluppo per realizzare un prodotto di alto livello, nel pieno rispetto delle ore concordate.

Dal punto di vista tecnico, il progetto si è rivelato estremamente stimolante. La sfida di integrare e far comunicare tra loro più LLM di piccole dimensioni, sfruttando al massimo le tecnologie a mia disposizione, è stata particolarmente avvincente. Questa scelta architetturale ha richiesto un approccio creativo al problem solving e mi ha permesso di ottenere risultati che andavano oltre le aspettative iniziali.

Sono molto soddisfatto del prodotto finale, nel quale ho potuto applicare e consolidare attivamente molte delle conoscenze teoriche acquisite durante gli anni di università. Vedere i concetti studiati trasformarsi in una soluzione software funzionante è stata la conclusione perfetta di un percorso formativo che considero fondamentale per il mio futuro professionale.

Glossario

Agile Metodo di gestione progetti che enfatizza iterazioni rapide, collaborazione continua e adattamento flessibile ai cambiamenti durante lo sviluppo del prodotto. iv

API In informatica, un'API è un insieme di procedure a disposizione dei programmatori, tipicamente raggruppate per formare un toolkit per un compito specifico all'interno di un programma. Il suo scopo è quello di fornire un'astrazione, solitamente tra l'hardware e il programmatore o tra software di basso e alto livello, semplificando il processo di programmazione. 3

Asynchronous Server Gateway Interface è uno standard per la comunicazione tra server web e applicazioni Python asincrone. È l'evoluzione di WSGI, pensato per supportare WebSocket, richieste asincrone e applicazioni in tempo reale. Consente prestazioni migliori in ambienti moderni. 23

ChatGPT è un assistente virtuale basato su intelligenza artificiale, sviluppato da OpenAI, che utilizza modelli di linguaggio avanzati per comprendere e generare testo in linguaggio naturale. È in grado di rispondere a domande, svolgere compiti di scrittura, programmazione, traduzione e molto altro. Viene usato in ambito educativo, lavorativo e quotidiano per semplificare l'interazione uomo-macchina. 8

Document Object Model è una rappresentazione ad albero di una pagina web, che consente a JavaScript di accedere e modificare contenuti, strut-

tura e stile. Ogni elemento HTML è un nodo del DOM. Permette l'interazione dinamica con la pagina. È fondamentale nello sviluppo web lato client. 33

Embedding è una rappresentazione numerica di dati, come parole o immagini, che permette ai computer di comprendere e analizzare il contenuto in modo più efficace. Viene usato soprattutto nell'intelligenza artificiale per trasformare informazioni complesse in vettori utilizzabili da algoritmi. 3

Framework è una struttura software predefinita che fornisce un insieme di componenti, strumenti e regole per facilitare lo sviluppo di applicazioni. Offre un'architettura di base su cui gli sviluppatori possono costruire, evitando di scrivere da zero le funzionalità comuni (es. gestione degli eventi, routing, accesso ai dati). iv

Garbage Collector componente della gestione automatica della memoria che rileva e rimuove oggetti non più utilizzati da un programma. Libera spazio nell'heap evitando sprechi di memoria. È comune in linguaggi come Java, C# e Python. Migliora la sicurezza e l'efficienza, ma può introdurre pause durante l'esecuzione. 28

GitHub è una piattaforma online per ospitare e gestire codice sorgente usando Git, un sistema di controllo versione. Permette a sviluppatori di collaborare, tracciare modifiche e gestire progetti software. Supporta repository pubblici e privati e facilita la condivisione e revisione del codice. È molto usato per sviluppare software sia open source che privato. 2

Hardware è l'insieme delle componenti fisiche di un computer o di un dispositivo elettronico, come CPU, RAM, disco rigido, scheda madre e periferiche. È la parte tangibile, opposta al software, che invece riguarda i programmi e le istruzioni. Il funzionamento del sistema dipende dall'interazione tra hardware e software. Può essere interno (es. processore) o esterno (es. tastiera, monitor). 3

HyperText Transfer Protocol L'uso del protocollo HTTP, in locale permette di comunicare tra componenti software sullo stesso dispositivo, spesso durante lo sviluppo. Le API locali espongono funzionalità accessibili tramite richieste HTTP (es. localhost:3000). Facilitano il test, la modularità e l'integrazione tra servizi. 29

Java Virtual Machine è la macchina virtuale che esegue i programmi Java, traducendo il bytecode (codice intermedio) in istruzioni comprensibili dal sistema operativo. Rende Java portabile su diverse piattaforme senza modifiche al codice. Gestisce memoria, sicurezza ed esecuzione del programma. 29

JSON è un formato leggero per lo scambio di dati, facile da leggere e scrivere per gli esseri umani e semplice da interpretare per le macchine. Utilizza una sintassi basata su coppie chiave-valore e strutture annidate come oggetti e array. È indipendente dal linguaggio ma spesso usato con JavaScript. È molto comune in applicazioni web per la comunicazione tra client e server. 7

Large Language Model Modello di intelligenza artificiale di grandi dimensioni, addestrato su enormi quantità di testo, capace di comprendere e generare linguaggio naturale in modo avanzato. iv

Microsoft Teams è una piattaforma di comunicazione e collaborazione che integra chat, videochiamate, condivisione di file e strumenti di lavoro. È usata principalmente da aziende e scuole per facilitare il lavoro di gruppo e le riunioni online. Supporta integrazioni con altre app Microsoft come Office 365. Favorisce la collaborazione in tempo reale tra team distribuiti. 2

Node Package Manager è il gestore di pacchetti predefinito per Node.js. Permette di installare, aggiornare e condividere librerie JavaScript. Facilita la gestione delle dipendenze nei progetti. 32

NuExtract è una famiglia di modelli linguistici leggeri (LLM) sviluppata da NuMind, progettata specificamente per l'estrazione strutturata di dati da testo non strutturato, con output in formato JSON. È particolarmente adatta per applicazioni che richiedono efficienza computazionale e prestazioni elevate. iv

Scrum Framework Agile per gestire progetti complessi, basato su sprint brevi, ruoli definiti (come Product Owner e Scrum Master) e riunioni regolari per migliorare la collaborazione e il progresso. iv

Unified Modeling Language è un linguaggio di modellazione e specifica basato sul paradigma orientato agli oggetti. UML funge da "lingua franca" nella comunità della progettazione e programmazione orientata agli oggetti. Gran parte della letteratura di settore utilizza UML per descrivere soluzioni analitiche e progettuali in modo conciso e comprensibile per un vasto pubblico. 37

Way of working indica il modo in cui un team o un'organizzazione organizza e gestisce il proprio lavoro, includendo processi, metodologie, strumenti e abitudini adottate per collaborare e raggiungere obiettivi. Definisce come si comunica, si prende decisioni e si coordina il lavoro quotidiano. 2

WebApp è un software accessibile tramite browser, che funziona su internet senza bisogno di essere installato sul dispositivo. Utilizza tecnologie come HTML, CSS e JavaScript per l'interfaccia utente, spesso con un backend che gestisce dati e logica. È multipiattaforma e aggiornabile centralmente. 20

Sitografia

Diagramma dei casi d'uso. URL: https://www.html.it/pag/15664/use-case-diagram/.

Doc 3 Tier. URL: https://www.fs.com/it/blog/cos%C3%A8-larchitettura-threetier-1387.html.

Doc Drools. URL: https://docs.drools.org/7.6.0.Final/drools-docs/html_single/.

Doc Nomic. URL: https://www.nomic.ai/.

Doc TypeScript. URL: https://www.typescriptlang.org/docs/.

Docs Java. URL: https://docs.oracle.com/javase/8/docs/api/.

Docs Python. URL: https://docs.python.org/3.11/library/index.html.

Docs React. URL: https://react.dev/learn.

GitHub. URL: https://github.com/.

HuggingFace Course. URL: https://huggingface.co/learn/llm-course/.

HuggingFace NuExtract2. URL: https://huggingface.co/numind/NuExtract-2-2B-experimental.

Monolite Modulare. URL: https://www.adesso.it/le-architetture-a-microservizi-e-monoliti-modulari.

Node Package Manager. URL: https://www.npmjs.com/.

Node.js. URL: https://nodejs.org/en.

Nomic Huggingface. URL: https://huggingface.co/nomic-ai/nomic-embed-text-v1.5.

Ollama. URL: https://ollama.com/.

Pip. URL: https://pypi.org/project/pip/.

Prompt engineering. URL: https://www.ibm.com/think/topics/prompt-engineering-guide.

Qwen3. URL: https://ollama.com/library/qwen3.

strategia italiana AI. URL: https://www.notizie.ai/pathal/uploads/2024/04/Dpd_Executive_Summary.pdf.

Vite.dev. URL: https://vite.dev/guide/.

Bibliografia

Articoli

Piccoli, Gregorio. «TriplaAAA». In: *Zucchetti S.P.A.* (2025) (cit. a p. 3).