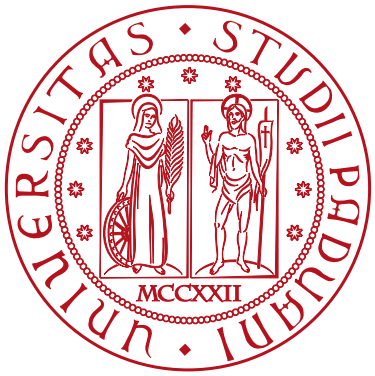
Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica



Sviluppo di un Sistema di Generazione Basato su Recupero delle Informazioni (RAG) per l'Analisi e l'Automazione dei Processi Aziendali

Tesi di laurea

Relatore

Prof. Lamberto Ballan

Laureando

Alessandro Carraro

Matricola 2000548

ANNO ACCADEMICO 2023-2024

Alessandro Carraro: *Sviluppo di un Sistema di Generazione Basato su Recupero delle Informazioni (RAG) per l'Analisi e l'Automazione dei Processi Aziendali*,
Tesi di laurea, © Dicembre 2024

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di circa trecento ore, dal laureando Alessandro Carraro presso l'azienda SpazioDev.
Questo progetto di tirocinio si concentra sullo sviluppo di un sistema di generazione basato sul recupero delle informazioni, *Retrieval-Augmented Generation* (RAG), completamente gestito in locale, realizzato in Python, integrato con Ollama, utilizzando l'ampio modello linguistio, *Large Language Model* (LLM), llama3:70B, e supportato da database.
L'obiettivo principale è fornire un'infrastruttura interna in grado di gestire processi aziendali senza dipendere da fornitori esterni di modelli linguistici.
Il sistema, containerizzato tramite Docker e configurato con DDEV (strumento che offre ambieti di sviluppo preconfigurati), permette di eseguire analisi avanzate sui dati aziendali fornendo risposte precise basate su documentazione specifica e/o di precedenti progetti aziendali.
Il risultato finale è un sistema utilizzabile per ricerche interne, analisi del codice e automazione di processi, favorendo il miglioramento continuo delle attività aziendali.
Il lavoro si basa su una metodologia che sfrutta tecniche di containerizzazione per garantire un sistema flessibile e scalabile, in grado di adattarsi alle esigenze dei diversi progetti aziendali.
Grazie all'integrazione di strumenti di apprendimento automatizzato, *Machine Learning* (ML), e all'uso di modelli linguistici su server locali, il sistema è progettato per offrire un'architettura modulare e personalizzabile, facilitando la gestione efficiente delle risorse informative interne.

Indice

1. L'Azienda **1.**
1.1. Dominio Applicativo ..... 1.
1.1.1. Storia dell'Azienda ..... 1.
1.1.2. Il Personale ..... 1.
1.1.3. Ambiti di Lavoro ..... 2.
1.1.4. Progetti ..... 2.
1.2. Processi di Sviluppo ..... 3.
1.2.1. Ricerche di Mercato ..... 3.
1.2.2. Organizzazione Settimanale ..... 3.
1.2.3. Organizzazione giornaliera ..... 4.
1.2.4. Progettazione ..... 4.
1.2.5. Sviluppo ..... 5.
1.2.6. Relazioni Interpersonali ..... 5.
1.3. Tecnologie Usate ..... 6.
1.3.1. Comunicazione Interna ..... 6.
1.3.2. VCS ..... 6.
1.3.3. ITS ..... 7.
1.3.4. Hardware Aziendale ..... 7.

2. La Proposta aziendale **8.**
2.1. Progetto di stage ..... 8.
2.1.1. Proposta StageIt ..... 8.
2.1.2. Cambio di Progetto ..... 8.
2.1.3. Proposta Finale ..... 9.
2.1.4. Variazione al Piano di Progetto ..... 9.
2.2. Obiettivi di Progetto ..... 10.
2.2.1. Analisi dei Requisiti ..... 10.
2.2.2. Interfaccia Utenti ..... 10.
2.2.3. Documentazione ..... 10.
2.2.4. Metodologia ..... 10.
2.3. Obiettivi Personali ..... 11.
2.3.1. Obiettivi Tecnici ..... 11.
2.3.2. Obiettivi Sociali ..... 11.

3. Il Tirocinio **12.**
3.1. Prime Settimane ..... 12.
3.1.1. Configurazione Ambiente ..... 12.
3.1.2. Norme Aziendali ..... 14.
3.2. Tecnologie Usate ..... 16.
3.2.1. Python ..... 16.

3.2.2. MySQL ..... 16.
3.2.3. Chroma DB ..... 16.
3.2.4. Laravel e Filament ..... 17.
3.2.5. Docker e DDEV ..... 17.
3.3. Sviluppo del progetto ..... 18.
3.3.1. RAG - Ingesting ..... 18.
3.3.2. RAG - Embedding ..... 19.
3.3.3. RAG - Indexing ..... 20.
3.3.4. RAG - Retrieving ..... 21.
3.3.5. RAG - Generating ..... 22.
3.3.6. RAG - Database ..... 23.
3.3.7. RAG - Contenirizzazione ..... 26.
3.3.8. RAG - Integrazione ..... 28.

4. Conclusioni **30.**
4.1. Consuntivo finale ..... 30.
4.1.1. Obiettivi Aziendali ..... 30.
4.1.2. Obiettivi Personali ..... 31.
4.1.3. Valutazione Finale ..... 31.

Capitolo 1.

L'Azienda

1.1. Dominio Applicativo

Questa sezione parla della storia dell'azienda, del personale, degli ambiti di lavoro e dei progetti.

1.1.1. Storia dell'Azienda

L'azienda è stata fondata da Matteo Forzan, mio Tutor interno, e da Nicolò Zuanon nel settembre 2023.
L'unione delle conoscenze commerciali di Nicolò si fondava perfettamente con le conoscenza informatiche di Matteo.
L'esperienza nella gestione di locali di ristorazione ha suscitato in Nicolò il desiderio di sviluppare un servizio di fidalizzazione per i clienti, l'incontro con Matteo è avvenuto pressochè per caso, ma fin da subito entrambi hanno capito la potenzialità del progetto, investendoci creando una software house pronta per lo sviluppo. Da qui nasce SpazioDev.

1.1.2. Il Personale

L'azienda ha deciso di suddividere il personale in reparti:

- **Reparto Commerciale**: Questo reparto è composto da tre persone, coordinate da Nicolò (un fondatore dell'azienda), e si occupa di cercare potenziali clienti, proporre i servizi dell'azienda tramite incontri digitali o in presenza;
- **Reparto Automazione**: Il reparto è composto da due persone e si occupa di tutti quei progetti che vanno ad implementare automazioni in processi industriali.
- **Reparto Gestionali/CRM**: Questo reparto è composto da tre persone (quattro quando aumenta il carico di lavoro) e si occupa delle implementazioni o delle assistenze lato back-end di software gestionali o *Customer Relationship Management* (CRM).
- **Reparto Web**: Questo reparto è composto da tre persone e si occupa della progettazione e dello sviluppo di siti web, principalmente vetrina e, se necessario, dell'immagine digitale di un'azienda o di un privato;
- **Reparto Modellazione**: Questo reparto è composto da una persona e si occupa della modellazione e della stampa 3D di prototipi plastici per aziende, per privati o per progetti interni. Inoltre offre rendering 3D per qualsiasi progetto;
- **Reparto AI**: Questo reparto è composto da tre persone, coordinate da Matteo (un fondatore dell'azienda), e si occupa di studio e applicazione di soluzioni con Intelligenza Artificiale, *Artificial Intelligence* (AI), in progetti aziendali o futuri. Questo è il reparto in cui ho lavorato.

1.1.3. Ambiti di Lavoro

L'azienda non ha un settore specifico in cui lavora, sta cercando di svilupparsi più orizzontalmente per esplorare il più possibile il mercato e le sue richieste. Per questo motivo l'azienda ha diversi reparti che lavorano su tecnologie diverse e, come si vedrà in seguito, anche i progetti aziendali sono eterogenei.

1.1.4. Progetti

- **Fault Prevention**: Questo progetto ha come obiettivo la creazione di un software in grado di prevenire fallimenti in macchinari industriali.
  Grazie a questo progetto l'azienda ha acquisito conoscenze tecniche nell'ambito elettronico e nei protocolli di comunicazione, oltre a creare un reparto specializzato in applicazioni o web app sviluppate in React e React Native;
- **RelAI**: Questo progetto è il principale prodotto dell'azienda, consiste in un CRM a cui si affidano locali di ristorazione o centri estetici o organizzatori di eventi per gestire la fidelizzazione dei loro clienti.
  Grazie a questo progetto l'azienda ha acquisito conoscenze di comunicazione e marketing, oltre a creare un reparto specializzato nel lato backend di un web service tramite php;
- **Mugalab**: Questo è il progetto creativo attraverso il quale l'azienda propone i propri servizi di modellazione o stampa 3D o rendering.
  Grazie a questo progetto l'azienda ha creato un solido bagaglio tecnico per la prototipazione plastica;
- **WebScraper**: Questo è il progetto più sperimentale dell'azienda, grazie all'utilizzo di AI si cerca di valutare possibili clienti e proporre mirati servizi offerti dall'azienda.
  Grazie a questo progetto l'azienda ha acquisito conoscenze nel campo dell'AI, oltre a creare un reparto specializzato nello sviluppo di applicazioni con Python e l'integrazione dell'AI nei progetti aziendali;
- **Progetti Didattici**: Questi progetti hanno lo scopo di arricchire le conoscenze del personale aziendale, principalmente i programmatori.
  Sono progetti che permettono di esplorare e imparare nuove tecnologie.
  I progetti didatti principali sono due:
  - **TowerGame**: Si pone l'obiettivo di creare un gioco della tipologia Tower Defense multigiocatore, quindi con la gestione di una sessione di gioco, in un server, tra due giocatori;
  - **OpenHands**: Si pone l'obiettivo di sviluppare degli agenti che tramite LLM riescano a codificare specifiche funzionalità, come inizializzare database o operazioni di creazione, lettura, modifica o eliminazione, *Create Read Update Delete* (CRUD), con php.

1.2. Processi di Sviluppo

Questa sezione parla dei metodi di lavoro dell'azienda: le ricerche di mercato, l'organizzazione settimanale e giornaliera, le fasi di progettazione e sviluppo. Inoltre ha un approfondimento dedicato alla formazione.

1.2.1. Ricerche di Mercato

Fondamentali per un'azienda in crescita sono le ricerche di mercato, espandere il proprio network e far conoscere i propri prodotti; questo aspetto è molto sentito all'interno dell'azienda e tutti quanti cercano e si impegnano per far crescere l'azienda.
In questo momento i canali principali per la ricerca di mercato sono due:

- **Reparto Commerciale**: Come detto in precedenza sono ragazzi che propongono i nostri servizi ad aziende o privati;
- **Reparto AI**: Questo è il reparto in cui ho lavorato io e sta cercando di ottimizzare le possibilità di successo nel trovare clienti a cui vendere i servizi. Come vedremo successivamente questo reparto sta portando grandi innovazioni, sia lato hardware, con computer e server più potenti per gestire calcoli più onerosi, sia lato del software con un grande studio delle tecnologie e delle strategie.

1.2.2. Organizzazione Settimanale

Un altro aspetto fondamentale per un'azienda in crescita è uno buona organizzazione del lavoro, questi processi richiedono tempo, ma sono investimenti che un'azienda deve fare; una buona pianificazione riduce possibili difficoltà future.
In particolare i momenti organizzativi settimanali sono:

- **Allineamento Inizio Settimana**: La settimana inizia con una riunione di un paio di ore in cui ogni reparto ed ogni respondabile di progetto, *Project Master* (PM), illustrano gli obiettivi raggiunti e la pianificazione della settimana.
  Questo momento è molto importante per allineare tutta l'azienda sul lavoro svolto e da svolgere, oltre ad essere un ottimo momento di brainstorming e problem solving.
  Questo momento di confronto permette ad ogni reparto di portare il proprio contributo e la propria conoscenza in ogni progetto aziendale;
- **Test Infrasettimanali**: Il mercoledì o il giovedì pomeriggio sono dedicati al controllo del lavoro svolto, all'interno di ogni reparto viene analizzato il lavoro svolto e vengono segnalate problematiche di sviluppo.
  Questo momento è indispensabile per garantire il corretto svolgimento del progetto, per allineare tutto il team e per assicurare di star procedendo nella giusta direzione;
- **Allineamento Fine Settimana**: La settimana finisce con un resoconto dei team ai fondatori dell'azienda, soprattuto a Matteo, direttore tecnico, e un confronto fra PM sulle difficoltá incontrate.
  Questo momento permette di valorizzare il lavoro svolto e sopratutto preparare il lavoro per la settimana a venire.

Una nota alla **Formazione del Venerdì**, un ritaglio temporale di due o tre ore al venerdì pomeriggio per sperimentare con nuove tecnolgie o per esplorare soluzioni innovative nei progetti aziendali.
Un ottimo momento collettivo in cui tutto il personale lavora per il continuo miglioramento.
Oltre ad essere un'attività didattica è anche un'attivita pensata per spronare a svolgere un buon lavoro e rispettare le scadenze della settimana, infatti, solo nel caso in cui tutte le scadenze fossero rispettate si ha la possibilita di organizzare questi incontri.

1.2.3. Organizzazione giornaliera

Analogamente all'organizzazione settimanale anche quella giornaliera è fondamentale, permettendo di non perdere il focus nel lavoro da svolgere.
I punti principali per garantire un'ottima organizzazione giornaliera sono:

- **Orari Flessibili**: Per quanto possa risultare poco produttivo, permettere alle persone di lavorare quando sono concentrate e al massimo delle loro capacità risulta una scelta vincente. Inoltre questo approccio rafforza molto le capacità di lavoro asincrono dell'azienda, forzando un'ottima programmazione e una piena conoscenza dei progetti;
- **Riunione Mattiniera**: La mattina risulta un perfetto momento per i programmatori per confrontarsi su difficoltà e soluzioni dei problemi, infatti spesso esponendo il problema ad un occhio esterno si riesce ad avere una visione che porta velocemente ad una soluzione, questo momento rafforza molto il legame e la fiducia fra colleghi;
- **Aggiornamenti Pomeridiani**: Il momento successivo alla pausa pranzo è un'altra ottima occasione per i programmatori per allinearsi sul confronto mattutino, un ottimo modo per motivarsi a vicenda e proseguire con il lavoro nel pomeriggio;
- **Aggiornamenti Fine Giornata**: La giornata si conclude con cinque minuti di allineamento per riportare i risultati giornalieri e allineare tutti i programmatori.

1.2.4. Progettazione

Tanto importante è l'organizzazione, quanto lo è la progettazione.
Una buona progettazione permette di ottimizzare i tempi di lavoro e le risorse, oltre che ottenere un prodotto più completo.
Alla base di una progettazione risiedono:

- **Analisi del progetto**: Una prima analisi iniziale del progetto, con studio di fattibilità, permette di incanalare le risorse aziendali al meglio.
  Questo processo viene svolto principalemte dal PM, ma il contributo di tutto il team è essenziale per evitare fraintendimenti o in fase di sviluppo;
- **Analisi delle tecnologie**: L'analisi delle tecnologie è un altro aspetto fondamentale, l'azienda riesce a spaziare nelle conoscenze tecnologiche grazie ai vari reparti, questo garantisce veloci progressi in fase di sviluppo;

- **White Board**: Nonstante la crescita della tecnologie, l'analogico è ancora radicato nell'organizzazione e nella progettazione basica di alcune fasi.
  Spesso una veloce delucidazione con carta e penna oppure lavagna bianca e pennarello garantiscono un veloce superamente degli ostacoli e una veloce progettazione senza fraintendimenti;
- **POC**: Un'altro aspetto fondamentale è la realizzazione di una dimostrazione di fattibilità, *Proof of Concept* (POC), il prima possibile per valutare sin da subito le scelte prese e la direzione scelta.
  Un ottimo modo per evitare futuri disagi in fase di sviluppo.

1.2.5. Sviluppo

Terminata la progettazione si passa allo sviluppo, ovvero la fase centrale e più corposa nella vita di un progetto.
La fase di sviluppo prevede delle azioni preliminari:

- **Definizione Team e Ruoli**: La scelta del team è fondamentale, così come l'assegnazione dei compiti. Queste sono due scelte cruciali per la fase di sviluppo, evitano perdite di tempo future;
- **Consuntivi**: Valutare costantemente il lavoro è fondamentale, permette di capire se le risorse sono applicate correttamente e se il progetto rispetta i vincoli.
  Pianificare in anticipo la verifica di più aspetti possibili è molto importante;
- **Interazioni con Cliente**: Sono essenziali le interazioni con il cliente, in caso di dubbio, chiarimenti sull'analisi dei requisiti, su criticità di sviluppo e, se necessario, su valutazioni errate.

1.2.6. Relazioni Interpersonali

In questa sezione vorrei sottolineare quanto quello scritto in precedenza, tutte le fasi di organizzazione, le fasi di sviluppo e tutte le scelte prese dall'azienda, contribuiscano molto a creare un forte legame interpersonale fra i colleghi. Inoltre questi momenti di scambio e confronto permettono di conoscersi meglio, conoscere i punti di forza e le debolezze di ogni persona.

1.3. Tecnologie Usate

Questa sezione parla delle tecnologie pratiche per lo sviluppo usate dall'azienda: dalla comunicazione giornaliera asincrona, al tracciamento delle issue, dei task e al versionamento.
In questa sezione infine si illustra velocemente l'hardware aziendale.

1.3.1. Comunicazione Interna

Avere un buon canale di comunicazione è fondamentale per lo sviluppo di un progetto.
In azienda si sono scelte due tecnologie:

- **Telegram**: Offre la naturalezza di una chat fra colleghi e una facile interazione nei gruppi, le abitudini di comunicazione delle vita privata si riportano facilmente in Telegram vista la sua semplicità.
  Allo stesso tempo risulta difficile differenziare molto le conversazioni con il rischio di mischiare discussioni nello stesso gruppo;
- **Discord**: Offre una maggior gestione delle comunicazioni, con la presenza di canali completamente adattabili alle necessità specifiche della conversazione. Una facile condivisione di documenti, link o codice lo ha reso un ottima soluzione per la comunicazione interna asincrona.
  Inoltre, al contrario di Telegram, non c'è un limite alle chat, quindi non si perdono messaggi con il tempo.

1.3.2. VCS

Importantissimo per una azienda di sviluppo software, *software house*, è avere un sistema di controllo delle versioni del software, *Version Control System* (VCS).
L'azienda ha scelto di esplorare una soluzione gestita in locale: **Gitea** nello specifico.
Gitea permette di creare e gestire repository basandosi su Git, garantendo un veloce curva di apprendimendo.
Basandosi su Git garantisce tutti quei servizi base: creazione di branch, tags, milestone ed assegnazione delle issue.
Inoltre Gitea, come GitHub, incorpora le actions, garantendo un workflow facile e familiare.
Tutto questo garantendo un livello altissimo di sicurezza e, essendo dipendente dall'hardware aziendale, permette un maggior controllo e una maggior velocità.

1.3.3. ITS

Importantissimo per la fase di sviluppo è un sistema di tracciamento delle problematiche, *Issue Tracking System* (ITS).
L'azienda ha scelto di esplorare, anche in questo caso, una soluzione gestita in locale: **Plane**.
Il punto forte di Plane è il totale adattamento al progetto, dalla creazione delle issue, alla gestione delle issue tramite assegnazione di ruoli, date, tag e sopratutto legami bloccanti con altre issue.
Plane, infatti, permette di creare dei cicli del progetto, questi cicli richiedono la definizioni di una sequenza di issue da completare, garantendo che ogni fase del progetto venga completata al meglio.
Plane, inoltre, permette la visualizzazione grafica dello svolgimento del lavoro e tracciamento del tempo di sviluppo.
Plane non richiede una grande curva di apprendimento, però richiede impegno e dedizione nell'utilizzo per essere sfruttato al massimo delle sue potenzialità.

1.3.4. Hardware Aziendale

Importante per una software house è possedere del buon hardware, maggiori sono le potenzialità di calcolo e di archiviazione maggiore sono le possibilità di crescita per l'azienda e per i suoi progetti.
In particolare l'azienda ha deciso di investire in un potente computer con due schede video per disporre di abbastanza VRAM per i modelli di AI e per la modellazione e rendering 3D, inoltre l'azienda ha acquistato un server in grado di gestire e smaltire lunghi lavori su più processori, suddividendo il lavoro nei vari core a disposizione.
Infine l'azienda ha predisposto una connessione Gigabit su tutti i computer, sul server e sul NAS aziendale per l'archiviazione e condivisione di grandi file.

Capitolo 2.

La Proposta aziendale

2.1. Progetto di stage

Questa sezione parla del progetto proposto dall'azienda, le problematiche riscontrare e il cambio avvenuto in conclusione.

2.1.1. Proposta StageIt

Ho conosciuto l'azienda tramite l'evento StageIt organizzato dall'università, in questa occasione l'azienda aveva portato tre progetti, uno dei quali, quello che più mi aveva colpito si chiamava *Smart Agriculture*, consisteva nell'utilizzare le immagini di campi agricoli catturate da un drone, farle elaborare da un'AI e tramite AI analizzarle per valutare lo stato di salute delle piante.
Come mai nasce il progetto?
I fondatori dell'azienda erano a conoscenza delle tecniche di valutazione dello stato di salute di piantagioni di palme da cocco in paesi africani.
In queste vastissime piantagioni il lavoro di analisi viene svolto a mano valutando su una cartina lo stato della pianta, per poi intervenire in ampie zone, tramite l'uso di veicoli aerei, con prodotti chimici disinfestanti.
Ovviamente questo processo richiede molto tempo e molto denaro, da qui la proposta di automatizzare il tutto con droni e AI.

2.1.2. Cambio di Progetto

Sfortunatamente il progetto non si è avviato come previsto, pertanto l'azienda mi richiedeva uno studio teorico e la preparazione di un POC per gli investitori.
Il progetto è stato sottovalutato dall'azienda e non mi conveniva iniziarlo perchè sarei stato da solo e poco seguito.
In accordo con l'azienda abbiamo quindi valutato altre proposte che rimanessero sempre in ambito AI.

2.1.3. Proposta Finale

La proposta finale dell'azienda consisteva nello sviluppare un sistema di generazione automatica della risposta tramite ricerca aumentata da documenti, *Retrieval-Augmented Generation* (RAG), da integrare con *RelAI*, il loro prodotto principale.
Il sistema RAG permette la generazione di risposta partendo da documenti selezionati e una volta sviluppato, il RAG deve ricevere lo storico delle campagne marketing fatte dall'azienda e fornire in risposta una profilazione degli utenti e delle strategie per le campagne future.
Questa proposta si inseriva in un progetto già avviato, che mi permetteva di essere molto più seguito e sostenuto, inoltre il reparto era ben formato, quindi gli obiettivi erano molto più chiari, cosi come le buone pratiche di sviluppo.
Un contesto molto più realistico per una figura giovane come la mia.

2.1.4. Variazione al Piano di Progetto

Un'altro cambio è avvenuto durante il periodo di stage.
L'azienda aveva fatto partire il nuovo reparto AI e aveva bisogno di un'aiuto sul progetto WebScraper, quindi, in accordo con il tutor interno ed il tutor universitario ho modificato le ultime settimane del piano di progetto.
Terminate la fase di implementazione del RAG in *RelAI*, non ancora completamente ultimata l'interfaccia utente, *User Interface* (UI), mi sono spostato verso il nuovo progetto aziendale.
Questo progetto prevedeva lo sviluppo di modelli AI per la classificazione grafica dei siti web.

2.2. Obiettivi di Progetto

Questa sezione parla degli obiettivi del progetto, le aspettative del mio responsabile interno: l'analisi dei requisiti, l'interfaccia utente, la documentazione ed infine la metodologia.

2.2.1. Analisi dei Requisiti

I requisiti fondamentali del progetto erano:
- Creare un'interfaccia di input per il caricamento dei documenti;
- Creare un'interfaccia di input per la richiesta al RAG;
- Creare un'interfaccia di ouput per visualizzare la risposta del RAG;

In questo progetto di stage non esisteva un vero e proprio cliente, ma delle esigenze interne all'azienda che andavano esplorate.
Dopo la variazione al piano di progetto i requisiti si sono aggiornati:
- Clusterizzazione con apprendimento non supervisionato, *unsupervised learning,* degli screenshot delle pagine web di un'ampio database di siti;
- Creazione del modello di classificazione tramite apprendimento supervisionato, *supervised learning*;
- Classificazione dei siti tramite il modello sopra citato;
- Valutazione della classificazione e successivo invio automatico di email.

2.2.2. Interfaccia Utenti

L'interfaccia utenti doveva integrarsi con RelAI, il CRM offerto dall'azienda, sviluppato in Laravel e Filament, framework php, e contenerizzato con DDEV, strumento che sfrutta Docker per fornire container preconfigurati.
Successivo al cambio del piano di progetto è seguito anche il cambio dell'interfaccia utenti richiesta: sempre utilizzando Laravel e Filament andava implementeta un'interfaccia per la schedulazione e l'invio delle email.

2.2.3. Documentazione

Il progetto richiedeva la documentazione necessaria per garantire ai futuri sviluppatori un facile inserimento all'interno del progetto, inoltre, vista la natura didattica del tirocinio, era un ottimo modo per familiarizzare con pratiche aziendali.
L'azienda utilizza Plane oltre che come ITS, anche come piattaforma per la documentazione interna.
Questo strumento integrato nelle pratiche aziendali è perfetto per la documentazione interna.
Su Plane è possibile creare documenti ed associarli ai progetti, non è dotato di versionamento, ma la documentazione richiesta era abbastanza semplice.

2.2.4. Metodologia

Un'altra richiesta dell'azienda era la produzione di un software altamente manutenibile e evolvibile.
Vista la natura del progetto, il forte utilizzo di AI, richiede la possibilità di evolvere facilmente alcune funzionalità.

2.3. Obiettivi Personali

Questa sezione parla delle motivazioni che mi ha portato a scegliere l'azienda e il progetto.
Premetto che non ero interessato a fare il tirocinio in una grande azienda per poter inserire il loro nome nel curriculum, non era neanche interessato a cercare un'azienda che mi garantisse un tirocinio rimunerato, come non lo era trovare un'azienda comoda o che offrisse smart-working.
Il mio obiettivo principale era andare in un'azienda dove potessi imparare il più possibile.

2.3.1. Obiettivi Tecnici

Era per me molto importate trovare un progetto che mi permettesse di esplorare il mondo AI, volevo conoscere più a fondo questo mondo sia dal lato teorico che dal lato pratico.
La proposta di SpazioDev era per me la più convincente.

2.3.2. Obiettivi Sociali

Un'altra caratteristica aziendale che mi piaceva esplorare era la grandezza e la complessità dell'azienda: ero molto curioso di provare una realtà piccola, ancora in fase di crescita, dove l'impegno individuale contava molto, rispetto a grandi aziende dove c'è il rischio di essere visti solo come un altro programmatore.
Anche in questo caso, l'ambiete di SpazioDev era molto attraente.

Capitolo 3.

Il Tirocinio

3.1. Prime Settimane

Questa sezione parla delle prime settimane di tirocinio, in cui ho configurato l'ambiente di lavoro e ho appreso le norme di sviluppo aziendali.

3.1.1. Configurazione Ambiente

L'azienda mi ha fornito una postazione su cui lavorare, ho quindi dovuto configurare il computer ed inserire le mie credenziali.

Google Chrome
In azienda viene usata una web app per la timbratura, siccome utilizza delle credenziali di accesso, ho deciso di accedere al mio profilo google in modo da poter salvare le password.
Inoltre l'accesso su chrome mi permetteva di salvare nei preferiti alcune pagine, come quelle di documentazione e quelle di strumenti aziendali, come Plane o Plesk.
Salvare queste pagine garantiva una facile e veloce consultazione in caso di dubbi.

Telegram e Discord
Sono i due canali di comunicazione interna, ho quindi provveduto subito ad effettuare l'accesso ed entrare nei canali aziendali.

Git
Fondamentale per il versionamento ho installato Git e GitHub Desktop, effettuato l'accesso e clonato in locale i repositori su cui dovevo lavorare.
Inoltre, come menzionato in precedenza, l'azienda utilizzava Gitea per il versionamento di alcuni progetti, ho provveduto a richiedere che il mio account Git fosse collegato con l'account aziendale per visualizzare i progetti interni.

Docker e DDEV
Fondamentale per un ambiente di lavoro condiviso è contenizzare il progetto.
Ho quindi provveduto a scaricare Docker Desktop ed installare DDEV nella mia macchina.
I progetti aziendali sono tutti contenerizzati con DDEV, quindi ho letto la documentazione e testato il funzionamento per capirlo appieno.

Visual Studio Code
L'azienda non impone un ambiente di sviluppo comune, *integrated development environment* (IDE), ogni programmatore è libero di usare quello che preferisce, nel mio caso per abitudine e semplicità di adattamenteo, ho utilizzato *Visual Studio Code* (VS Code), ho effettuato l'accesso con il mio profilo per la sincronizzazione, ho poi provveduto ad installare le estensioni necessarie per i progetti aziendali:

- Pacchetto estensioni per Python;
- Pacchetto estensioni per Laravel;
- Pacchetto estensioni per Php;
- Pacchetto estensioni Git Lens.

WSL
RelAI, il progetto su cui ho lavorato, viene eseguito sull'ambiente virtuale Linux in Windows, *Windows Subsystem for Linux* (WSL), per garantire maggior velocità, ho quindi installato nel computer WSL 2 e spostato nell'apposita cartella di sistema i progetti aziendali.

Python e Venv
Ho ovviamente installato una versione di Python per lo sviluppo del RAG e ho attivato un ambinete virtuale *venv* per sviluppare il RAG.

Ollama
Per il mio progetto dovevo sviluppare un RAG, quindi mi serviva un modello AI per generare le risposte, l'azienda mi ha proposto di utilizzare un modello in locale, ci siamo affidati ad Ollama.
In azienda c'è un computer in cui è installato il modello *llama3:70b*, ma in locale, per effettuare test, ho installato un modello più leggero: *codellame:7b*.

TensorFlow 3.10
Nella seconda parte di progetto, quella definita dal cambio, ho utilizzato la potenza della scheda video NVidia per analizzare le immagini, questo ha portato all'installazione di TensorFlow 3.10, ultima versione su windows compatibile con scheda grafica, *graphics processing unit* (GPU).
TensorFlow è un framework open source sviluppato da Google che consente di creare e allenare modelli di machine learning e deep learning.
Questa installazione è stata veramente faticosa poichè richiedeva specifiche dipendenze di sistema e un'attenta installazione, ha richiesto quasi un giorno, ma con il sostegno in azienda l'installazione è avvenuta con successo.

Account Aziendali
Infine mi sono stati forniti gli account aziendali per l'accesso ad alcune risorse:

- ChatGPT pro;
- NAS locale;
- Plesk;
- Plane;

3.1.2. Norme Aziendali

Questa sezione parla delle convenzioni aziendali per i progetti di gruppo.

Issue
Le norme riguardati le issue sono:
- **Titolo Parlante**: Fondamentale è creare le issue con titoli parlanti, così da garantire una rapida comprensione per tutto il gruppo;
- **Assegnare Stato**: Assegnare ed aggiornare lo stato di una issue permette di evitare che più persone inizino a lavorare sulla stesso issue, oltre che dare un'indicazione sull'evoluzione del progetto;
- **Assegnare Programmatore**: Fondamentale assegnare la issue a un programmatore, così da garantire che venga chiusa;
- **Assegnare Date**: Quando si inizia a lavorare su una issue è fondamentale inserire la data di inizio, analogamente quando si termina il lavoro, così da poter completare correttamente i consultivi;
- **Assegnare Dipendenze**: Assegnare una gerarchia all'interno delle issue, questo permette di svolgere il lavoro con la giusta progressione.
  Stabilire quale issue ne blocca un'altra permettere di non perdere traccia del lavoro da svolgere e semplifica la pianificazione;
- **Assegnare Label**: Assegnare un'etichetta alla issue permette di filtrare facilmente le issue.

Versionamento
Le norme riguardati il versionamento sono:
- **Branch**: Sviluppare il più possibile in branch atomici, aprire tanti branch feature quante funzionalità ha il prodotto;
- **Nome Branch**: Utilizzare la sintassi *feature/funzione-x* per nominare i branch;
- **Merge**: Prima di effettuare il merge su develop o main, assicurarsi di integrare prima il branch principale nel branch secondario, così da prevenire eventuali errori in produzione;
- **Comunicare**: Fondamentale comunciare push o merge quando si sta lavorando alla stessa funzionalità.

Commit
Le norme riguardanti i commit sono:
- **Commit Atomici**: Fondamentale inserire modifiche atomiche in ogni commit, questo permette di individuare facilmente la modifica in caso di errore. Un commit atomico solitamente include un solo file, ma potrebbe includere più file nel caso in cui il commit riguardi una funzionalità gestita in più file.
- **Titolo**: Il titolo del commit deve riferire che operazione si sta facendo, quindi se si tratta di creazione, modifica, eliminazione, update o refactory;
- **Descrizione (opzionale)**: Fortemente consigliata per motivare le modifiche effettuale;
- **Lingua (opzionale)**: Preferibilmente i commit vanno fatti in lingua inglese, ma è opzionale.

Php
Le norme riguardanti PHP sono:

- **Laravel**: Programmando con il framework Laravel la documentazione stabilisce le convenzioni da seguire, tutti i programmatori devono seguire la documentazione ufficiale;
- **Filament**: Analogo alla documentazione Laravel;
- **Creazione File**: Creazione di file tramite comando specifico PHP artisan;
- **Nominazione**:
  - Le **Classi** segnuono il PascalCase, esempio *class UserController{}*;
  - I **Metodi** seguono il camelCase, esempio *public function getUserName(){}*;
  - Le **Variabili** seguono il camelCase, esempio *$userName = "John Doe"*;
- **Lingua**: Usare la lingua inglese;
- **Commenti**: Utilizzare più commenti possibili
  - **Commenti in linea** per specificare assegnazioni o valori;
  - **Commonti multi linea** per invarianti o comportamento di classi e metodi;
- **Incapsulamento**: Utilizzare il principio di incapsulamento per controllare l'accesso ai dati e ai comportamenti di una classe;
- **Principio di Responsabilità Unica**: Applicare questo principio il più possibile su classi e metodi.

Python
Le norme riguardanti python sono:

- **Nome File**: Utilizzare snake_case, esempio *my_script.py*;
- **Nominazione**:
  - Le **Classi** seguono il PascalCase, esempio *class MyClass*;
  - I **Metodi** seguono lo snake_case, esempio *def my_method(self)*;
  - Le **Variabili** seguono lo snake_case, esempio *my_variable = 10*;
  - Le **Costanti** seguono l'UPPER_CASE_WITH_UNDERSCORES, esempio *MAX_RETRIES = 5*;
  - I **Moduli** seguono lo snake_case, esempio *import my_module*;
  - Le **Eccezioni** seguono il PascalCase con il suffisso Error, esempio *class ValidationError(Exception)*;
- **Principio di Responsabilità Unica**: Applicare questo principio il più possibile su classi e metodi.

3.2. Tecnologie Usate

Questa sezione parla delle tecnologie usate durante il progetto.

3.2.1. Python

Per lo sviluppo del RAG ho scelto Python.
Questo linguaggio non è il più veloce, ad esempio C++, linguaggio studiato affondo durante il percorso universitario, garantisce una maggior velocità, inoltre non è il più facile da integrare in *RelAI*, visto che questo prodotto aziendale è basato su framework PHP, PHP sarebbe stato più facile da integrare.
Python, però, offre potenti librerie per l'elaborazione del linguaggio naturale e l'AI, libreria che mancano per gli altri due linguaggi.
Inoltre per Python è presente una grande comunità che condivide tutorial, esempi, e best practice per costruire RAG e altri sistemi AI.
Altro vantaggio è la facilità con cui si riesceno ad integrare database.
Infine, essendo Python un linguaggio ad alto livello, permette di sviluppare rapidamente prototipi e iterare sul design del sistema.

3.2.2. MySQL

Nel progetto, i dati sono stati principalmente archiviati e gestiti utilizzando database relazionali MySQL, che garantiscono efficienza e scalabilità.
Per il RAG era necessaria l'integrazione con *RelAI*, questo prodotto aziendale utilizza un database MySQL, quindi per semplicità ho utilizzato lo stesso database.
In particolare avevo la necessità di salvare e leggere documenti e il loro contenuto.
Come strumenti ho utilizzato *MySQL Workbench* per la modellazione dello schema e *PHPMyAdmin* per la gestione operativa, gli stessi utilizzati in azienda.

3.2.3. Chroma DB

Chroma DB è stato utilizzato come database vettoriale per la memorizzazione e il recupero di rappresentazioni semantiche dei dati.
Analizzerò meglio nelle sezioni successive questo strumento.
La scelta di Chroma su un database relazionale SQL è dovuta al supporto nativo della gestione di embedding, fondamentali per recuperare informazioni basate su similarità semantica.

3.2.4. Laravel e Filament

Anche in questo caso la scelte delle tecnologie è stata legata al prodotto *RelAI*, il CRM offerto dall'azienda.
Laravel è un framework PHP moderno, progettato per creare applicazioni web scalabili e manutenibili.
Questo lo rende perfetto per un CRM, che spesso richiede funzionalità complesse e personalizzabili.
Filament è una libreria per Laravel che permette di creare rapidamente dashboard e pannelli di amministrazione.
Questo è particolarmente utile in un CRM, dove è necessario gestire entità come clienti, vendite, o spese.
Filament utilizza un design moderno e accessibile, basato su *Tailwind CSS*, strumento utilizzato in azienda.

3.2.5. Docker e DDEV

Docker è una piattaforma che permette di sviluppare, ed eseguire applicazioni in contenitori.
Essendoci diversi ambienti di sviluppo in azienda, Docker è stata una scelta forzata per evitare codice funzionante solo su certe macchine.
DDEV è una piattaforma di sviluppo che semplifica l'utilizzo di Docker per gestire ambienti di sviluppo complessi.
Integra Docker con una serie di configurazioni predefinite per applicazioni web, come PHP, MySQL e altro, riducendo la necessità di configurazioni manuali.
DDEV è stato scelto per la sua capacità di automatizzare la creazione e gestione degli ambienti di sviluppo grazie alle sue configurazioni pronte all'uso.

3.3. Sviluppo del progetto

Questa sezione parla dello sviluppo del progetto.
In particolare il progetto ha avuto 3 fasi:
1. Sviluppo del RAG;
2. Sviluppo del modello AI per classificazione siti web;
3. Invio automatico di email.

Poichè questo documento tratta dello sviluppo del RAG le sottosezioni successive analizzarennao i vari passaggi solo della prima fase dello stage, lo sviluppo del RAG appunto.

3.3.1. RAG - Ingesting

La prima fase di un RAG è l'ingesting, letteralmente ingestione, in questa fase il sistema riceve in input dei documenti e ne ritorna una variabile con il contenuto dei documenti.
In particolare nel mio caso i documenti ricevuti in input erano file pdf e l'output erano due liste di stringhe, la prima contente i titoli dei documenti, la seconda contenente i testi dei documenti.
Ho utilizzato due librerie:

```
import os
from PyPDF2 import PdfReader
```

1. La prima per la lettura da disco del documento: `os` cerca un documento nella cartella con i documenti e, se lo trovava, costruisce il percorso completo verso quel documento.

   ```
   path = "/path/to/documents/folder"

   if filename in os.listdir(path):
     full_path = os.path.join(path, filename)
       if full_path.endswith('.pdf'):
         texts.append(_extract_text_from_pdf(full_path))
   ```

2. La seconda libreria per la lettura del contenuto del documento, in particolare la lettura del titolo e del testo pagina per pagina.

   ```
   file = "/path/to/documents/folder/example.pdf"

   reader = PdfReader(file)

   metadata = reader.metadata
   if metadata and '/Title' in metadata:
     title =  metadata['/Title']

   text = ""
   for page in reader.pages:
     text += page.extract_text()
   ```

Considerazioni:

- Ogni tipologia di documenti richiede librerie specifiche, però considerando che per la maggior parte dei documenti è possibile la conversione in pdf, io ho esplorato l'ingesting di file pdf;
- Conoscere il contenuto di un documento potrebbe facilitare l'ingestione, se ad esempio si ha un documento tecnico molto lungo, ma ben stutturato, si potrebbe preferire un'ingestione dei capitoli o delle sezioni piuttosto di tutto il testo per intero, allo stesso modo se si hanno documenti molto eterogenei si possono ingerire anche i metadati dei documenti per maggiori informazioni;
- Bisogna sottolineare che questa fase è la più dispendiosa in termini di tempo, poichè molto meccanica, analizzare l'intero documento e salvarlo richiede più tempo tanto è lungo il documento.
  Come analizzerò in seguito, un'ottima strategia è salvare il contenuto in un database.

3.3.2. RAG - Embedding

La seconda fase di un RAG è l'embedding, la vettorializzazione del documento, in questa fase il contenuto del documento viene vettorializzato all'interno di uno spazio multi dimensionale, in questo modo si riesce a dare un significato matematico al documento.
Questa è la fase più importante, poichè più precisa e complessa sarà la vettorializzazione maggiore sarà la precisione con cui il sistema RAG riuscirà ad estrarre le informazioni.
Ho utilizzato:

```
from transformers import AutoTokenizer, AutoModel
```

Dalla libreria `transformers` ho importato:

1. `AutoTokenizer` per assegnare un token, un valore numerico, al testo dei documenti, usando il modello pre addestrato *all-MiniLM-L6-v2*;
2. `AutoModel` che carica il modello pre addestarto *all-MiniLM-L6-v2*.

```
pre_trained_model = "sentence-transformers/all-MiniLM-L6-v2"
tokenizer = AutoTokenizer.from_pretrained(pre_trained_model)
model = AutoModel.from_pretrained(pre_trained_model)

#texts = list of contents of documents
inputs = tokenizer(texts, padding=True, truncation=True,
return_tensors="pt")

embeddings = model(**inputs).last_hidden_state.mean(dim=1).numpy()
```

La variabile `embedding` è una lista, della stessa dimensione della variabile *texts*, contenente, non più stringhe, ma *numpy.ndarray*.

Considerazioni

- Esistono altri modelli, e verranno sicuramente implementati nuovi modelli in futuro, ma lo scopo del progetto non era sviluppare il RAG più preciso possibile, ma dimostrare che il sistema RAG poteva essere integrato nei progetti aziendali.
  OpenAI, ad esempio, fornisce librerie per embedding, ma voleva esplorare soluzioni meno convenzionali;
- Come per l'ingesting, anche questa fase richiede diverso tempo, quindi sarebbe opportuno salvare il risultato delle operazioni in un database, tratterò di questo argomeno più avanti.

Ottimizzazione

- Si può disabilitare il calcolo dei gradienti durante le operazioni, riducendo l'uso di memoria e aumentando la velocità

  ```
  import torch
  with torch.no_grad()
  ```
- Si possono normalizzare i vettori embeddings in modo che abbiano una norma L2 unitaria, rendendoli di lunghezza unitaria nel loro spazio vettoriale

  ```
  import faiss
  faiss.normalize_L2(embeddings)
  ```

3.3.3. RAG - Indexing

La terza fase di un RAG è l'indexing, l'indicizzazione degli embedding.
Questa fase è fondamentale per organizzare e rendere accessibile i dati.
In particolare io ho usato *FAISS (Facebook AI Similarity Search)*.
L'indice è progettato per permettere ricerche rapide basate sulla somiglianza semantica, spesso calcolata con metriche come il coseno della distanza o la distanza euclidea, come nel mio caso.

```
import faiss

# embedded_texts = list of embeddings of documents
embed_dim = embedded_texts.shape[1]

index = faiss.IndexFlatL2(embed_dim)

index.add(embedded_texts)
```

3.3.4. RAG - Retrieving

La quarta fase di un RAG è il retrieving, ovvero il recupero dei documenti a partire da una domanda.
Questa fase ha bisogno di una query, ovvero la domanda, questa query va a suo volta vettorializzata e passata all'indice calcolato in precedenza.
Io ho utilizzato la libreria *FAISS* che permette facilmente di inserire documenti vettorializzati nell'indice e, altrettanto facilmente, permette di ricercara un numero n documenti rilevanti la query.

```
# embedded_query = embedded query
# documents_names = list of names of documents
# index = index generated with faiss
# n = number of document to retrieve

# Search the FAISS index for the documents closest to the query
embedding
distances, indices = index.search(embedded_query, n)

# Retrieve the most relevant documents and their distances
results = [
  (documents_names[i], distances[0][idx])
  for idx, i in enumerate(indices[0])
]
```

Considerazioni

- Analogamente alla vettorializzazione dei documenti, più precisa sarà la vettorializzazione della query, migliore sarà il recupero dei documenti;
- Sottolineo che la lista dei documenti ha la stessa dimensione della lista con gli embedding, la lista degli embedding viene inserita nell'indice, quindi per corrispondenza l'indice verso gli embedding corrisponde all'indice verso i documenti, grazie a questa corrispondenza posso utilizzare i nomi dei documenti come metrica per valutare i documenti recuperati e la relativa distanza alla query;
- Il numero di documenti recuperati è importante: un numero basso di documenti recuperati potrebbe portare ad una risposta incompleta o povera di dettagli, mentre un numero alto di documenti potrebbe portare ad una risposta incorretta, poco precisa, poichè, trovando molte corrispondenza, il sistema RAG potrebbe generare risposte non autentiche.
- A seconda dei documenti si potrebbero trovare diverse strategie per migliorare il recupero: ad esempio se il numero di documenti è ampio, si potrebbe preferire una ricerca sui titoli e sui sommari dei documenti, prima che su tutto il testo, questo permette un recupero più preciso.
  Oppure si potrebbe assegnare un peso alla corrispondenza di specifiche parole, così da ponderare adeguatamente la distanza fra la query e il documenti.

3.3.5. RAG - Generating

La quinta e ultima fase di un RAG è generating, ovvero la generazione della risposta.
In questa fase il sistema utilizza un modello AI per la generazione, la scelta del modello è fondamentale.
Nel mio caso, viste le richieste del progetto, si è scelto un modello Ollama, *llama3:70b* per la precisione, ostato in un computer aziendale.
Il computer locale faceva da client per la convesazione fra il RAG ed il modello.

```
from ollama import Client

# query = string with the question
# retrieved_documents_contents = list with the content of the
retrieved documents

client = Client(host='http://192.168.x.x:x')

prompt =
f"Question: {query}\nDocuments: {retrieved_documents_contents}"

response = client.chat(model='llama3:70b', messages=[
  {
    'role': 'user',
    'content': prompt,
  }
])

answer = response['message']['content']
```

Considerazioni
La variabile *prompt* è cruciale in questa fase.
Modificando il suo valore si può ottenere un grande miglioramento nella risposta.
Ad esempio se si vuole una risposta in italiano bisogna assicurarsi di scrivere il prompt in italiano e, volendo, anche specificare che si vuole, appunto, la risposta in italiano.
Altre richieste più specifiche al modello fanno fatte tramite il prompt.

3.3.6. RAG - Database

Come anticipato nelle sezioni precedenti, alcune fasi del sistema RAG sono molto dispendiose in termini di tempo, le operazioni che devono eseguire sono complesse e lunghe.
Una soluzione che ho adottato è stata eseguire le operazioni una sola volta e salvare il risultato in un database.
La prima fase che richiede tempo è l'ingesting, poichè deve analizzare l'intero documento e salvarne tutto il testo sotto forma di stringa dentro una variabile.
Per risolvere questo problema, per velocizzare l'utilizzo del sistema ho deciso di salvare i documenti in un database SQL.
La seconda fase che richiede tempo è l'embedding, poichè, anche in questo caso, deve analizzare l'intera stringa di testo e calcolarne la vettorializzazione, per poi salvarla sotto forma di matrice dentro una variabile.
Risolvere questo problema non è stato facile, ho inizialmente provato con il salvataggio in un database SQL, per poi passare all'utilizzo di ChromaDB.
Analizziamo ora le due soluzioni.

MySQL
Il RAG doveva integrarsi con il progetto aziendale *RelAI*, questo progetto utilizza un database SQL, quindi io dovevo utilizzare lo stesso database per le funzionalità del RAG.
Il progetto aziendale era basato su *Laravel* e *Filament* quindi dovevo imparare come utilizzare questo due tecnologie per interagire con un database.
I passi sono i seguenti:

1. **Migrazione**: Il primo passo è la creazione di una migrazione, che in Laravel rappresenta lo schema di una tabella nel database;

```
Schema::create('documents', function (Blueprint $table) {
  $table->id();
  $table->string('name');
  $table->string('path');
  $table->text('description')->nullable(); // opzional
  $table->timestamps();
});
```

2. **Modello**: Il secondo passo è la creazione del modello, ovvero l'oggetto che contiene il documento;

```
class Document extends Model
{
  protected $fillable = [
    'name',
    'path',
    'description'
  ];
}
```

3. **Risorse Filament**: Le risorse Filament facilitano le operazioni CRUD da parte dell'utente.
   Le varie risorse si creano tramite il comando:

   ```
   php artisan make:filament-resource ResourceName
   ```

   1. **Pagina creazione**: La pagina di creazione non è altro che una pagina con un form in cui si inseriscono gli attributi del modello precedentemente creato.
      Ecco il codice:

      ```
      $form->schema([
        FileUpload::make('path')
          ->label('Document')
          ->directory('documents')
          ->required(),

        TextInput::make('name')
          ->label('File Name')
          ->required(),

        Textarea::make('description')
          ->label('Description'),
      ]);
      ```
   2. **Pagina visualizza**: La pagina di visualizzazione non è altro che una pagina con una tabella in cui si visualizzano gli attributi del modello precedentemente creato.
      Ecco il codice:

      ```
      $table->columns([
        TextColumn::make('name')
          ->label('File Name'),
        TextColumn::make('description')
          ->label('Description'),
        TextColumn::make('created_at')
          ->label('Uploaded At')
          ->dateTime(),
      ]);
      ```
   3. **Pagina modifica**: La pagina modifica utilizza lo stesso form della pagina di creazione, andando però a modificare gli attributi del modello precedentemente creato.

Una volta eseguiti tutti questi passaggi si avrà un'interfaccia utente in cui è possibile inserire nuovi documenti, visualizzarli e modificarli.
I documenti verranno salvati nella cartella del progetto, che nel mio caso era contenerizzato, quindi venivano salvati dentro la cartella storage del container Docker.
Come vedremo successivamente la cartella storage del progetto veniva clonata all'interno del container del RAG, avevo quindi accesso ai documenti tramite il RAG.

ChromaDB

Il RAG deve salvare anche gli embedding, ma il salvataggio di questi ultimi risulta molto complicato in un database SQL, poichè si tratta di matrici multidimensionali e SQL non riesce a gestirle, richiede infatti di convertire le matrici in dati binari, salvare i dati binari e poi riconvertirli in matrici per utilizzarli.
Ho impiegato un'intera settimana per implementare il salvataggio degli embedding su database SQL e la lettura da SQL, ma non ho avuto nessun risultato, gli errori e i problemi nella conversione erano troppi.
Sono quindi passato a tecnolgie già ottimizzate: *ChromaDB*.
Questo sistema permette il salvataggio del contenuto dei documenti e degli embedding in maniera molto semplice, i passaggi sono i seguenti:

1. **Creazione della collezione**: Per creare una collezione bisogna prima creare un client, nel mio caso il client era il container docker del RAG, vedremo successivamente questo container

   ```
   client = await chromadb.AsyncHttpClient(
     host='chromadb_service',
     port=8000
   )
   collection = await client.create_collection(
     "collection_name"
   )
   ```

2. **Salvataggio in Chroma**: Il salvataggio in Chroma avviene tramite il metodo *collection.add*, permette di salvare con un'unica operazione il contenuto e gli embedding dei documenti.
   Ecco il codice:

   ```
   # names = list of names of documents - type string[]
   # contents = list of contents of documents - type string[]
   # embeddings = list of embeddings of contents - type np.array[]
   for name, content, embedding in zip(names, contents, embeddings):
     await collection.add(
       ids=[name],
       contents=[content],
       embeddings=[embedding],
     )
   ```

3. **Lettura da Chroma**: La lettura da Chroma avviene tramite il metodo *collection.get*, permette di ottenere con un'unica operazione il contentuo e gli embedding dei documenti. Ecco il codice:

   ```
   results = await collection.get(
     ids=names,
     include=["contents", "embeddings"]
   )
   contents = results['contents']
   embeddings = results['embeddings']
   ```

   In questo modo, quando mi serve il testo di un documento, non devo eseguere delle operazioni per estrarlo dal documento, ma mi basta leggerlo dal database Chroma, operazione molto più veloce ed efficace.

3.3.7. RAG - Contenirizzazione

Come detto in precedenza, il progetto Aziendale era contenerizzato con DDEV, ho quindi provveduto contenerizzare il RAG all'interno del progetto *RelAI*.
In particolare ho creato il container per il codice python del RAG e il container per il database Chroma.

Docker-Compose RAG
Ecco il codice del file *docker-compose.rag.yaml*:

```
services:
  rag:
    build:
      context: ../rag
    container_name: rag_service
    command: >
      sh -c "python main.py && tail -f /dev/null"
    volumes:
      - ../rag:/app
      - ../storage/app/documents:/storage/app/public/documents
    ports:
      - "5000:5000"
    networks:
      - ddev_default

networks:
  ddev_default:
    external: true
```

Annotazioni:

- Il container che si crea associa il proprio storage con lo storage del container del progetto

  ```
  volumes:
    - ../storage/app/documents:/storage/app/public/documents
  ```

  In questo modo ogni modifica veniva simultaneamente esegueta in entrambe i container;
- Il container utilizzava le porte 5000

  ```
  ports:
    - "5000:5000"
  ```

  non essenci conflitti ho utilizzato quelle predefinite per container python;
- Infine il container dichiara di utilizzare DDEV per la gestione della rete

  ```
  networks:
    ddev_default:
      external: true
  ```

Dockerfile RAG
Il Dockerfile permette al container di eseguire il codice python

```
FROM python:3.9-slim

WORKDIR /app

COPY rag_requirements.txt .
RUN pip install -r rag_requirements.txt

COPY . .

CMD ["python", "main.py"]
```

Docker-Compose ChromaDB
Ecco il codice del file *docker-compose.chroma.yaml*:

```
services:
  chromadb:
    container_name: chromadb_service
    image: chromadb/chroma:latest
    volumes:
      - chroma_data:/data
    ports:
      - "8000:8000"
    networks:
      - ddev_default

volumes:
  chroma_data:
    driver: local

networks:
  ddev_default:
    external: true
```

Annotazioni:
- Il container utilizzava le porte 8000

  ```
  ports:
    - "8000:8000"
  ```

  non essenci conflitti ho utilizzato quelle predefinite per container chromadb;
- Infine il container dichiara di utilizzare DDEV per la gestione della rete

  ```
  networks:
    ddev_default:
      external: true
  ```

3.3.8. RAG - Integrazione

La fase finale dello sviluppo del RAG era l'integrazione fra il sistema già esistente e il RAG.
I due sistemi sono su due container indipendenti, dovevo quindi implementare chiamate API per la comunicazione fra *RelAI* ed il RAG.

API lato PHP

- **Caricamento su Chroma**

```
$rag_service_url = 'http://rag_service:5000/load_to_chroma';

$response = Http::get($rag_service_url, [
  'names' => $names
]);

$response = response()->json(['message' => 'Data successfully
sent to RAG service']);
```

- **Risposta dal RAG**

```
$documentsGeneratedNames = Document::where('organization_id',
1)
  ->pluck('generated_name')
  ->toArray();

$query = "Domanda per il rag :)";

$response = Http::retry(5, 100)->timeout(60)
  ->post('http://rag_service:5000/generate_rag_response',
    [
      'names' => $documentsGeneratedNames,
      'query' => $query,
    ]
  );

$response = response()->json($response->json());
```

API lato python

```
from quart import Quart, request, jsonify
```

- **Caricamento su chroma**

```
 @self.app.route('/load_to_chroma', methods=['GET'])
 names = request.args.get('names')

ingester = Ingesting(names)
contents = ingester.ingesting()

embedder = Embedding(contents)
embeddings = embedder.embedding()

chroma_instance = MyChroma()
await chroma_instance.chromaing()
await chroma_instance.save_to_chroma_async([names], contents,
embeddings)

return jsonify({"message": f"Loaded {names} into Chroma"})
```

- **Risposta dal RAG**

```
@self.app.route('/generate_rag_response', methods=['POST'])
data = await request.get_json()
names = data.get('names', [])
query = data.get('query', '')

chroma_instance = MyChroma()
await chroma_instance.chromaing()
contents, embeddings =
await chroma_instance.load_from_chroma(names)

indexer = Indexing(embeddings)
index = indexer.indexing()

retriever = Retrieving(query, names, contents, index)
DOCUMENTS_TO_RETRIEVE = 5
k = min(DOCUMENTS_TO_RETRIEVE, len(names))

retrieved = retriever.retrieving(k)

generator = Generating(query, names, contents, retrieved)
response = generator.generating_response_with_ollama()

return jsonify({'output': response})
```

Con questo si conclude l'integrazione del RAG nel progetto aziendale.
Il RAG è ora integrato nel progetto *RelAI*

Capitolo 4.

Conclusioni

4.1. Consuntivo finale

In questa sezione analizzo i risultati ottenuti a fine stage.

4.1.1. Obiettivi Aziendali

L'obiettivo principale per l'azienda era lo sviluppo e l'integrazione di un RAG. Oltre a questo l'azienda ha richiesto lo sviluppo di un modello AI per la classificazione dei siti web e, infine, lo sviluppo di una risorsa per l'invio automatico di email commerciali.

Sviluppo RAG
Lo sviluppo del RAG si è concluso con successo, il codice prodotto è altamente mantenibile ed evolvibile.
La separazione in classi delle diversa fasi del RAG ha permesso lo sviluppo di ogni strato in maniera indipendente e con una responsabilità ben definita.

Integrazione RAG
L'integrazione del RAG all'interno di *RelAI* non è stata completata al 100%, la funzionalità non è stata messa in produzione.
Il prodotto è comunque funzionante ed è una solida base per sviluppi futuri.
Anche in questo caso posso dire che il lavoro è stato ben fatto.

Documentazione RAG
La documentazione non è completa, molto scarsa anzi.
Con lo sviluppo ho fatto alcune note sulle scelte prese, ma non una documentazione completa.
Il codice risulta comunque ben strutturato e ben commentato, inoltre spero che questo documento possa risultare utile in futuro per ripercorrere le fasi dello sviluppo.

Classificazione Visiva Siti Web
Per il poco tempo impiegato nello sviluppo di questa richiesta aziendale il risultato è stato ottimo. L'approfondimento richiesto rimane comunque molto, ma il risultato finale mostra la potenzialità del progetto.
L'azienda non aveva mai sperimentato tanto con l'AI quanto nello sviluppo di questo progetto.
Il know-how aziendale è cresciuto esponenzialmente in questo ambito, affrontando sfide sul software, ma anche sull'hardware.
L'azienda ha capito la potenzialità di queste nuove tecnologie.

Invio Automatico Email
L'ultima settimana mi ha visto impegnato nello sviluppo di un sistema per inviare automaticamente email commerciali.
Il processo è stato molto meccanico seguendo delle guide, ma il risultato ottenuto è estremamente utile per l'azienda.
Anche in questo caso il know-how aziendale è cresciuto molto e la base di partenza per future implementazioni è estremamente solida e ben strutturata.

4.1.2. Obiettivi Personali

Il mio obiettivo principale era imparare, sotto questo punto di vista sono estremamente soddisfatto.
A livello tecnologico ho approfondito molto le mie conoscenze accademiche, dalla progettazione, allo sviluppo, fino al testare.
Era fondamentale per me partecipare ad un progetto in ambito AI e anche su questo devo ritenermi estremamante soddisfatto.
Ho provato con mano la potenzialità di questo mondo, la sua complessità e lo studio che ci sta dietro.
Questa esperienza mi ha fatto appassionare al rigore della programmazione, allo studio della documentazione e alla ricerca dell'ottimizzazione.

4.1.3. Valutazione Finale

In conclusione sono veramente soddisfatto di questa esperienza.
Ho trovato in SpazioDev un bell'ambiete, molto umano.
Il progetto mi ha molto soddisfatto e spero che quanto prodotto da me possa rivelarsi utile per l'azienda, per i progetti futuri e quelli presenti.
La mia curiosità di testare il mondo lavorativo del programmatore è stata pienamente soddisfatta.
Mi auguro di poter continuare ad accrescere le mie conoscenze e migliorare il più possibile in questo settore.