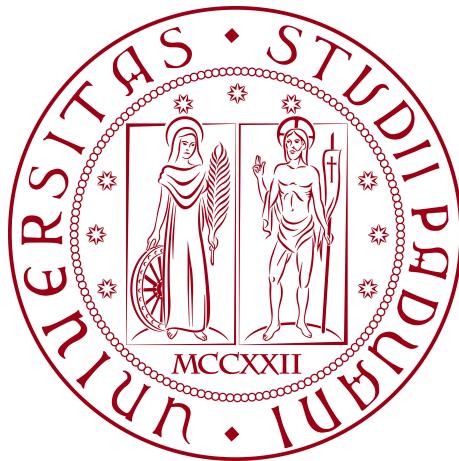


Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA “TULLIO LEVI-CIVITA”

CORSO DI LAUREA IN INFORMATICA



**Integrazione in gestionale ERP di un assistente
virtuale basato su piattaforme di Intelligenza
Artificiale**

Tesi di Laurea

Relatore

Prof. Vardanega Tullio

Laureando

Dal Bianco Riccardo

Matricola 2042385

© Dal Bianco Riccardo, Luglio 2025. Tutti i diritti riservati. Tesi di Laurea: “*Integrazione in gestionale ERP di un assistente virtuale basato su piattaforme di Intelligenza Artificiale*”, Università degli Studi di Padova, Dipartimento di Matematica “Tullio Levi-Civita”, Corso di Laurea in Informatica.

Cacio e Pepe

Ringraziamenti

Innanzitutto, desidero esprimere la mia gratitudine al Professor Tullio Vardanega, relatore della mia tesi, per il sostegno, l'aiuto e la disponibilità dimostrati durante la stesura della relazione finale.

Ringrazio Francesco Turra, amministratore di VisioneImpresa, e tutti i ragazzi che vi lavorano per l'accoglienza e il supporto che mi hanno offerto durante l'intero periodo di stage.

Con infinito affetto, desidero ringraziare i miei genitori per tutto l'appoggio e il sostegno che mi hanno sempre fornito durante questo percorso. Mi hanno permesso di inseguire i miei sogni, nonostante tutti i sacrifici che hanno dovuto fare.

Voglio ringraziare Sara: non era con me all'inizio, ma da quando è arrivata tutto è cambiato. Ha sempre creduto in me più di quanto riuscissi a farlo io.

Infine, un grazie va ai PADLUGHI, Baggio (Peter), Riki e Cri perché siete proprio così come siete e spero che questo non cambi mai.

Padova, Luglio 2025

Dal Bianco Riccardo

Sommario

Il presente documento descrive il lavoro svolto dal laureando Dal Bianco Riccardo durante il periodo di *stage* svolto presso l’azienda VisioneImpresa. La relazione di fine *stage* si divide in quattro capitoli:

1. Presentazione dell’azienda, informazioni generali su VisioneImpresa, i suoi prodotti, tecnologie utilizzate e propensione all’innovazione.
2. Contestualizzazione del progetto di *stage* all’interno della strategia aziendale, perché è stato proposto e aspettative aziendali e personali.
3. Descrizione delle attività svolte durante il periodo di *stage*: l’analisi, la progettazione, parte dello sviluppo, la validazione e i risultati conseguiti.
4. Raggiungimento degli obiettivi prefissati, valutazione e retrospettiva del periodo di *stage* e del percorso universitario.

Convenzioni tipografiche

All’interno del documento vengono utilizzate le seguenti convenzioni tipografiche:

- *Corsivo* indica i termini in lingua non italiana.
- **Grassetto** utilizzato negli elenchi puntati per enfatizzare la parola chiave e utilizzato nei titoli per enfatizzare le porzioni di testo.
- **Monospaziato** utilizzato per indicare nomi di funzioni, *database*, tabelle, *file* e classi.
- Parole del Glossario, rappresentate in corsivo, in blu e da una G a pedice (ad esempio *ERP_G*).

Indice

1	Contesto aziendale	1
1.1	VisioneImpresa	1
1.2	Clienti e servizi	3
1.3	Processi aziendali	5
1.3.1	Metodologie	5
1.3.2	Tecnologie e strumenti utilizzati	9
1.4	Propensione all’innovazione	13
2	Definizione del percorso di stage	15
2.1	Il rapporto tra azienda e <i>stage</i>	15
2.2	Proposte progettuali dell’azienda	16
2.3	Valutazione e presentazione del progetto	18
2.3.1	Valutazione e scelta	18
2.3.2	Presentazione del progetto	18
2.3.3	Vincoli generali del progetto	19
2.4	Vincoli tecnologici e obiettivi aziendali	21
2.4.1	Assistente VisionAI	22
2.4.2	Modulo di pianificazione interventi	24
2.5	Obiettivi personali	26
3	Il progetto di stage	27
3.1	Pianificazione delle attività	27
3.2	Sviluppo di API REST per VisionAI	28
3.2.1	Contesto applicativo e funzionamento di VisionAI	28
3.2.2	Analisi dei requisiti	30

INDICE

3.2.3	Progettazione	33
3.2.4	Difficoltà incontrate durante la codifica	38
3.2.5	Verifica e presentazione dei risultati ottenuti	39
3.3	Sistema di assegnazione automatica per VisionAssistance	41
3.3.1	Analisi dei requisiti	41
3.3.2	progettazione	43
3.3.3	Verifica e risalutati ottenuti	46
4	Valutazione retrospettiva	48
4.1	Valutazione degli obiettivi	48
4.1.1	Obiettivi aziendali	48
4.1.2	Obiettivi personali	49
4.2	Conoscenze e competenze acquisite	50
4.3	Considerazioni finali sul percorso universitario	51
Glossario		i
Sitografia		iv

Elenco delle figure

1.1	Obiettivi di VisioneImpresa in quanto società <i>benefit</i>	2
1.2	Enterprise Resource Planning.	3
1.3	Struttura generale di un processo Agile.	6
1.4	Ciclo di lavoro generale secondo Scrum.	7
1.5	Schema di integrazione degli strumenti nei processi aziendali . .	11
2.1	Pianificazione temporale dello <i>stage</i>	20
2.2	Tecnologie utilizzate per la comunicazione e la documentazione.	21
2.3	Ambienti di sviluppo per il modulo VisionAI.	22
2.4	Tecnologie utilizzate per il modulo VisionAI.	23
2.5	Ambienti di sviluppo per il modulo assegnazione interventi. . . .	25
2.6	Tecnologie utilizzate per il modulo di pianificazione interventi. .	25
3.1	Flusso operativo di VisionAI.	30
3.2	Architettura <i>API_G</i>	34
3.3	Diagramma di funzionamento di un ORM.	38
3.4	Interfaccia finale <i>form</i>	41
3.5	Struttura delle <i>repository</i> della <i>pipeline</i>	44
3.6	Struttura architettonica della <i>pipeline</i> dei dati.	45

Elenco delle tavole

2.1	Proposte di progetto presentate da VisioneImpresa.	17
2.1	Panoramica delle proposte progettuali.	18
2.2	Tabella obiettivi aziendali sezione VisionAI	24
2.3	Tabella obiettivi aziendali per sezione pianificazione interventi .	26
2.4	Obiettivi personali dello <i>stage</i>	26
3.1	Tabella del tracciamento dei requisiti.	33
3.2	API sviluppate e sistemate durante il periodo di stage.	40
3.3	Tabella dei requisiti individuati per il modulo di assegnazione interventi.	43
3.4	Scelte progettuali chiave adottate nella realizzazione della <i>pipeline</i> .	46

Capitolo 1

Contesto aziendale

1.1 VisioneImpresa

VisioneImpresa è un'azienda con oltre quarant'anni di esperienza nel settore dell'*Information Technology*. La sua attività è focalizzata sullo sviluppo di soluzioni *software* per l'automazione e l'organizzazione di processi aziendali. In particolare, offre una gamma di strumenti *ERP G*, sistemi integrati che permettono di gestire e coordinare in modo centralizzato le attività operative, dalla contabilità alla logistica, ottimizzando la gestione delle informazioni in un ambiente condiviso.

La sede si trova a Pernumia (PD), all'interno di un casolare ristrutturato nel cuore del territorio delle Terme Euganee ed ospita un *team* di circa venti persone. VisioneImpresa opera prevalentemente nel Nord Est, ma con contatti che si estendono su tutto il territorio nazionale.

Nel 2016 è entrata a far parte del gruppo OfficeGroup, un *network* di aziende orientate alla progettazione di *software* gestionali avanzati. L'ingresso in questa rete ha favorito la crescita soprattutto grazie allo scambio di conoscenze e competenze con altre aziende andando a rafforzare la posizione di mercato.

Il 5 ottobre 2023, VisioneImpresa diventa una società *benefit*, come viene indicato sul sito aziendale¹, non è stata una scelta imposta ma è arrivata at-

¹ Società Benefit - VisioneImpresa. URL: <https://www.vsh.it/azienda/societa-benefit/>.

traverso un lungo e consapevole percorso. Questa forma giuridica rappresenta le società che affiancano agli obiettivi di profitto anche attività che portano al beneficio comune.

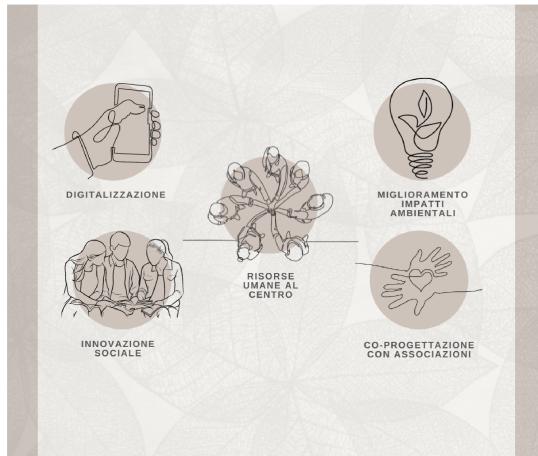


Figura 1.1: obiettivi di VisioneImpresa in quanto società *benefit*.
fonte: <https://www.vsh.it/azienda/societa-benefit/>

In qualità di società *benefit* VisioneImpresa adotta una visione imprenditoriale responsabile, sostenibile e trasparente; assumendo impegni verso la società e l'ambiente.

Come dimostra la figura 1.1 si impegna a perseguire i seguenti obiettivi:

- La promozione della digitalizzazione al fine anche dell'utilizzo di minore quantità di materiali supportando la transizione ecologica.
- Il miglioramento delle condizioni lavorative, mettendo al centro il benessere delle risorse umane, riponendo particolare attenzione a tematiche come la parità di genere, la formazione continua e la conciliazione tra attività lavorativa e vita privata.
- La promozione di progetti orientati all'innovazione e al benessere sociale favorendo la collaborazione con enti del territorio tra cui scuole e università.
- Partecipazione attiva in progetti locali, favorendo la collaborazione con enti del territorio per generare valore alla comunità.

- Mantenere un'alta attenzione all'impatto ambientale, impegnandosi nell'utilizzo di risorse energetiche pulite e riducendo l'impatto attraverso pratiche legate al consumo di risorse e alla gestione dei rifiuti.

1.2 Clienti e servizi

VisioneImpresa si rivolge principalmente a realtà di piccole e medie dimensioni situate nel Nord Est, ma opera anche in regioni del Centro Italia e in Sardegna. La principale offerta si fonda su soluzioni gestionali configurabili in base alle esigenze specifiche del cliente, affiancate da servizi di supporto tecnico e di consulenza. Il prodotto principale dell'azienda è VisionENTERPRISE, un sistema *ERP G* modulare progettato per adattarsi a praticamente tutti i contesti aziendali.



Figura 1.2: Enterprise Resource Planning.

fonte: <https://www.softplaceweb.com/erp-open-source/>

Si tratta di un sistema flessibile, che costituisce una base alla quale vengono affiancate anche soluzioni verticali pensate per settori specifici. Questi prodotti verticali si basano sulla stessa infrastruttura *software*, ma dispongono di funzionalità specifiche che hanno l'obiettivo di soddisfare esigenze particolari di

determinate filiere produttive.

Soluzioni verticali della linea Vision

Le soluzioni verticali offerte da VisioneImpresa includono:

- **VisionENERGY:** Gestionale pensato per aziende del settore petrolifero, include diverse funzionalità, in aggiunta a quelle tipiche di un *ERP*, mirate a soddisfare bisogni di questo settore come la gestione della vendita del carburante, la manutenzione degli impianti, la fatturazione con defiscalizzazione e la manutenzione degli impianti.
- **VisionBLUE:** Gestionale pensato per la filiera ittica, fornisce strumenti per attività legate alla vendita all'ingrosso dei prodotti ittici, come la gestione della tracciabilità dei prodotti, funzioni per l'inventario e organizzazione in contesti di alta intensità.
- **VisionASSISTANCE:** Gestionale dedicato alle aziende che forniscono assistenza su prodotti, consente di gestire richieste di intervento, stipulazione di contratti di assistenza, organizzazione dei tecnici sul territorio e assegnazione di interventi.
- **VisionFRESH:** Prodotto ideato per risolvere le esigenze del settore ortofrutticolo, contiene funzionalità che si integrano con le bilance elettriche, gestione dei lotti, imballaggi e prodotti e movimentazione della merce.
- **VisionANTINCENDI:** Gestionale che va a risolvere le richieste specifiche del settore della sicurezza e manutenzione di sistemi antincendio, offre funzionalità per la gestione degli interventi, gestione dei buoni di manutenzione e gestione dei contratti.
- **VisionTRASPORTI:** Gestionale orientato alle aziende di logistica e trasporti, contiene funzionalità specifiche per facilitare la gestione dei listini, delle anagrafiche oppure funzionalità per la pianificazione dei tragitti.

- **VisionELETTRON**: Gestionale dedicato alle officine elettromeccaniche, contiene funzionalità dedicate come la gestione delle matricole, gestione delle attività e navigazione per consultare lo storico degli interventi.

Servizi di supporto e personalizzazione

Oltre alla linea di prodotti principale VisioneImpresa offre diversi servizi accessori, tra questi ci sono un servizio di formazione rivolto al personale aziendale per istruirli all'utilizzo ottimale del *software*, un servizio di assistenza tecnica in caso di problemi, un servizio di aggiornamento e manutenzione. L'azienda inoltre offre la possibilità di effettuare personalizzazioni *ad-hoc* su richiesta, queste personalizzazioni sono vere e proprie modifiche al *software* di base per andare a risolvere esigenze mirate dei clienti, integrandole direttamente a livello del codice per mantenere stabile il sistema originario e fornire il massimo livello di prestazioni.

La linea MoviDat

In parallelo al gestionale principale, VisioneImpresa ha sviluppato una *suite* di applicazioni mobili, che si chiama MoviDat, creata per facilitare le attività in mobilità. Le applicazioni si integrano in modo nativo con i gestionali Vision e permettono di interagire con esso quando l'utilizzo del *computer* risulterebbe scomodo o impossibile. La linea MoviDat è composta da applicazioni per dispositivi mobili e compatibili con sistemi IOS o Android. L'obiettivo di questi prodotti è migliorare la fruibilità del gestionale semplificando azioni specifiche utilizzando in modo diretto uno *smartphone* o un *tablet*. Fanno parte della linea MoviDat le seguenti applicazioni: *MoviDoc*, *Han-dy*, *MoviSell*, *MoviRep*, *MoviAlert*, *MoviCheck*, *MoviExpenses*, *MoviCheckin*, *MoviOrder* e *MoviShop*.

1.3 Processi aziendali

1.3.1 Metodologie

Per quanto riguarda l'organizzazione delle attività l'azienda adotta metodologie Agile per la gestione dei progetti *software*. Questa tipologia di organizzazione si basa su cicli di lavoro brevi e ben definiti chiamati *sprint* la cui

struttura può essere visualizzata nella figura 1.3. Questi *sprint* consentono al *team* di adattarsi in maniera rapida ai cambiamenti concentrandosi nel migliorare il prodotto in maniera progressiva favorendo la comunicazione continua e riducendo il rischio di disallineamenti interni.

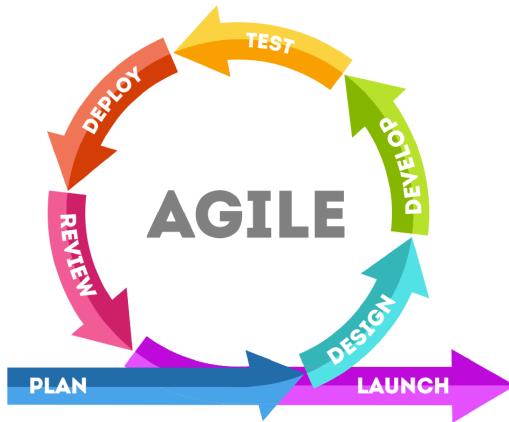


Figura 1.3: Struttura generale di un processo Agile.

fonte: <https://www.josoft.it/sviluppo-software-personalizzato/>

In particolare VisioneImpresa applica il *framework* Scrum, uno dei modelli più utilizzati nell'ambito Agile. Scrum prevede la suddivisione di un periodo lungo in *sprint* dalla durata fissa, di solito di 1-2 settimane, ognuno dei quali ha i propri obiettivi specifici e mira ad apportare un incremento al progetto. L'immagine seguente mostra la struttura tipica del ciclo Scrum 1.4.

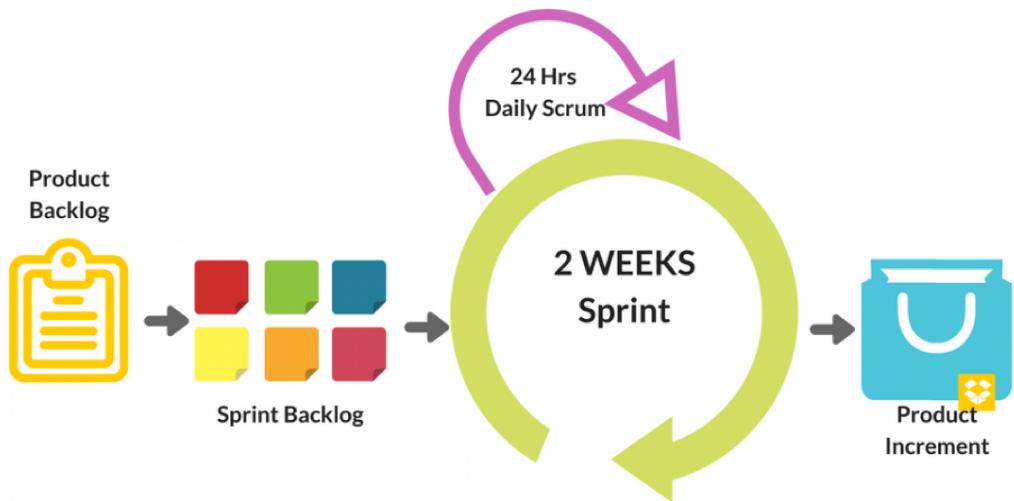


Figura 1.4: Ciclo di lavoro generale secondo Scrum.

fente:<https://vitolavecchia.altervista.org/differenza-tra-sprint-e-sprint-backlog-nella-metodologia-agile/>

Il *framework* sopra presentato definisce un insieme di ruoli, ed eventi, così come descritti nella *Scrum Guide*².

Tuttavia, nel contesto di VisioneImpresa l'approccio Scrum non viene seguito in modo scolastico ma viene adottato con coerenza in alcuni casi specifici. In particolare, ho avuto modo di osservare direttamente la metodologia con il quale questo *framework* viene applicato al progetto pluriennale di riscrittura del sistema gestionale VisionENTERPRISE, di cui fornirò più informazioni nella sotto-sezione 1.4. L'approccio adottato in questo contesto mantiene i principi fondamentali di questo metodo di lavoro, quali collaborazione, *feedback*, trasparenza, ma presenta alcune personalizzazioni legate alle esigenze aziendali.

L'applicazione concreta del metodo si può riassumere nei seguenti punti:

- **Sprint meeting settimanali:** ogni mercoledì viene organizzata una riunione di 2 ore per discutere dei progressi e delle next tasks.

²Ken Schwaber e Jeff Sutherland. *La Guida Definitiva a Scrum: Le Regole del Gioco*. URL: <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Italian.pdf>.

nione tecnica a cui partecipano gli sviluppatori coinvolti nel progetto. Questo incontro svolge sia la funzione di *sprint planning* dove si discute degli obiettivi e delle funzionalità da implementare nel prossimo periodo, ma anche la funzione di *sprint review* dove viene visualizzato il lavoro svolto nel periodo passato valutandone gli obiettivi conseguiti. Questa riunione viene fatta con l'obiettivo di discutere delle problematiche incontrate e fissare gli obiettivi per il prossimo periodo.

- **Daily meeting:** ogni giorno gli sviluppatori partecipano a una breve *call*, posizionata durante la mattinata, durante la quale tutti espongono cosa hanno fatto nel giorno precedente e quello in cui si concentrerà nella giornata corrente. Questo incontro viene svolto nel *software* 3CX, il centralino telefonico usato per le comunicazioni interne, ed è fondamentale per mantenere tutto il *team* allineato.
- **Flessibilità del lavoro:** un'altra caratteristica osservata è che gli sviluppatori non hanno un carico di lavoro uniforme ma vengono destinate alla realizzazione del progetto, per motivi aziendali, risorse limitate, infatti lavorano a questo progetto solo alcuni giorni della settimana. Questa scelta è motivata dall'impossibilità di interrompere le attività e i processi legati al vecchio *software*. Per questo motivo il *team* deve gestire il lavoro con grande flessibilità per evitare blocchi o disallineamenti che rappresentano un rischio per lo sviluppo.
- **Riunione mensile aziendale:** oltre alle attività settimanali, VisioneImpresa organizza un *meeting* mensile a cui partecipano tutti i membri dell'azienda. Durante questo incontro vengono presentati gli avanzamenti conseguiti durante il mese, si fa il punto della situazione riguardante lo stato dei progetti in corso e si presenta quello che verrà fatto nel prossimo mese. Oltre a un compito di allineamento, questa riunione serve anche per favorire il dialogo tra tutti i componenti dell'azienda, i dipendenti infatti sono invitati a proporre soluzioni a eventuali problemi che sono stati individuati, avanzare idee e offrire punti di riflessione utili allo sviluppo dei progetti, sia attuali che futuri. Questa fase può essere vista come una

sessione di *brainstorming* aziendale dove viene stimolato un confronto sui temi trattati.

Per concludere, l'adozione del modello Scrum da parte di VisioneImpresa non segue una struttura rigida, ma si configura secondo le esigenze progettuali. Nonostante tutto questo processo, visto da un occhio esterno, risulta coerente con i principi fondamentali e si dimostra efficace nell'organizzare e coordinare un progetto di grandi dimensioni come questo.

1.3.2 Tecnologie e strumenti utilizzati

Nel contesto di competenza nel quale opera, VisioneImpresa utilizza un ampia gamma di tecnologie a supporto delle attività. Questi strumenti possono essere divisi in base al contesto di utilizzo ovvero: strumenti operativi, strumenti per la comunicazione e la produttività, strumenti per lo sviluppo *software*, strumenti per la gestione del codice e dei progetti. Essendo così grande il numero di tecnologie impiegate e non essendo entrato in contatto con tutti gli ambiti dell'azienda, le tecnologie descritte in seguito sono solo quelle che ho avuto modo di osservare o quelle con cui ho avuto il modo di interagire durante il periodo di *stage*.

Strumenti operativi

- Portatili aziendali: ad ogni dipendente viene fornito un portatile personale, con sistema operativo Windows 10 o 11, oppure macOS.
- Dispositivi mobili: per i dipendenti che si occupano di sviluppo di applicazioni *mobile* vengono dati in dotazione *smartphone* e/o *tablet* con sistemi Android o IOS, utilizzati principalmente per funzioni di *testing* delle applicazioni.

Strumenti per la comunicazione e la produttività

- Microsoft Office 365: *suite* di strumenti Microsoft per la produttività.
- Outlook: client integrato, facente parte della *suite* Office, utilizzato per la posta elettronica e per la comunicazione interna e esterna.

- 3CX: centralino telefonico *PBX_G*, una tecnologia che consente creare una rete di comunicazioni interna, utilizzato per le chiamate vocali, messaggistica istantanea e riunioni online.

Strumenti per lo sviluppo *software*

- Visual Studio: Un *IDE_G* un ambiente di sviluppo che contiene strumenti per scrivere e testare codice progettato da Microsoft. Molto utilizzato nei progetti che coinvolgono il *framework .NET*, data una larga *gamma* di strumenti offerti.
- Visual Studio Code: *editor* di codice sorgente *open-source*, sviluppato da Microsoft, viene contraddistinto per la sua leggerezza, velocità e personalizzazione.
- SQL Server Management Studio (SSMS): Questo strumento è dedicato alla gestione dei *database* relazionali basati su SSMS, ha un interfaccia molto esplicativa e viene utilizzato per la gestione in tempo reale dei dati.
- Swagger: Piattaforma per la progettazione e la documentazione di *API_G REST_G*, ovvero interfacce che permettono la comunicazione tra sistemi *web*, permette di visualizzare in maniera chiara le chiamate, e offre strumenti per testare le richieste e le risposte.

Framework e linguaggi

- .NET: Un *framework* di sviluppo *software* creato da Microsoft che consente lo sviluppo di vari applicazioni sia a livello *desktop* sia a livello *mobile* e *web*.
- FoxPro: Linguaggio di programmazione e sistema per la gestione del *database*, attualmente considerato *legacy_G* a causa della dismissione ufficiale del supporto da parte di Microsoft nel 2015. Viene utilizzato per la manutenzione in attesa della completa migrazione verso tecnologie più moderne.

Gestione del codice e dei progetti

- Bitbucket: Strumento utilizzato per il versionamento del codice e facilitare la collaborazione tra membri del *team*. Offre funzionalità di controllo delle modifiche e revisione del codice.
- Jira: Strumento sviluppato da Atlassian per la gestione dei progetti e la divisione delle *task*. Supporta le metodologie Agile consentendo la gestione del lavoro in *sprint*.
- Excel: Nonostante sia uno strumento che non offre le stesse potenzialità di altri come Jira, viene utilizzato per la pianificazione elementare delle attività.

Integrazione delle tecnologie nei processi aziendali

Durante il periodo di *stage* ho potuto osservare come gli strumenti e le tecnologie vengono integrate all'interno del flusso di lavoro. Nello schema seguente 1.5 viene rappresentato in maniera semplificata l'integrazione dei principali strumenti all'interno dei processi aziendali, come osservato durante la mia esperienza.

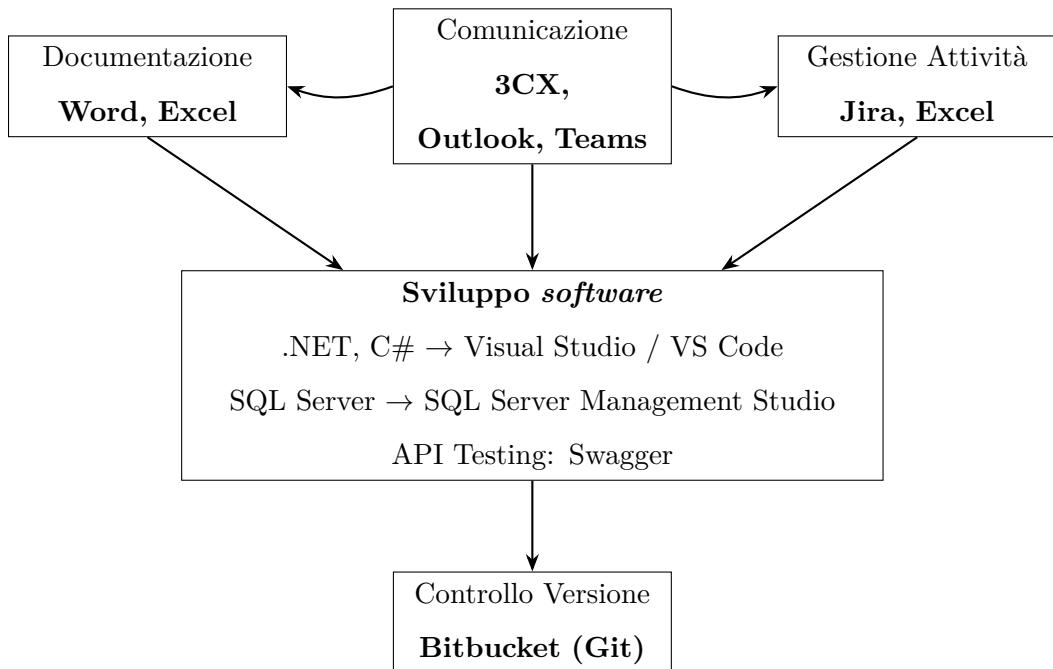


Figura 1.5: Schema semplificato dell'integrazione degli strumenti principali nei processi aziendali.

Di seguito è presentato nel dettaglio come ciascun processo aziendale viene affiancato e supportato dalle tecnologie citate.

- **Documentazione**

La scrittura della documentazione avviene principalmente attraverso Microsoft Word, con il quale vengono redatti i documenti più significativi, come le analisi dei requisiti e i casi d'uso, solitamente questa attività viene svolta dal responsabile di progetto e in seguito i documenti vengono condivisi con il *team* di sviluppo. Nel mio periodo di *stage*, ho preso visione di diversi documenti e ne ho scritti alcuni. Inoltre, vengono prodotti anche i documenti a supporto delle riunioni, che svolgono il ruolo di verbali, riportando i temi trattati. L'utilizzo di Excel si limita ad alcuni casi specifici.

- **Comunicazione**

Lo strumento principale per quanto riguarda la comunicazione interna è 3CX, utilizzato per chiamate, messaggi istantanei e riunioni. È lo strumento preferito per la collaborazione rapida tra i membri dell'azienda. Io non ho avuto un account per l'accesso a 3CX e ho comunicato con Microsoft Outlook, utilizzandolo per mandare e leggere mail e messaggi. In alcune occasioni come la riunione aziendale mensile viene utilizzato Microsoft Teams .

- **Gestione attività**

Le attività inerenti ai progetti vengono tracciate utilizzando principalmente Jira, ma talvolta a livello di supporto viene impiegato anche Microsoft Excel per via della sua semplicità. Durante il mio *stage* i miei compiti erano assegnati via email o tramite riunioni in presenza, non ho utilizzato quindi in prima persona strumentazione per la gestione delle attività.

- **Sviluppo *software***

Per quanto riguarda lo sviluppo *software* le tecnologie e i linguaggi adottati variano in base al progetto di riferimento. Per la mia esperienza il *framework* di riferimento è .NET, utilizzato con il linguaggio C#, viene usato per lo sviluppo di applicazioni *web* e *desktop* in maniera performante. Gli ambienti di sviluppo variano sempre in base al progetto di riferimento ma principalmente vengono utilizzati Visual Studio o Visual

Studio Code. Tutte le applicazioni sviluppate si interfacciano con *database* relazionali gestiti tramite SQL Server Management Studio (SSMS), questo strumento viene utilizzato per la creazione, la manutenzione e l'interrogazione del *database*. Per quanto riguarda le *API_G REST_G*, VisioneImpresa utilizza Swagger come strumento di testing e documentazione, favorendo la visualizzazione dei risultati e migliorando lo sviluppo.

- **Controllo versione**

Per quanto riguarda invece la gestione del codice in azienda viene utilizzato Bitbucket. Anche se non mi sono interfacciato direttamente con i *repository*, ho visualizzato alcuni flussi tipici del codice. Il codice di ogni progetto viene messo in *repository* dedicati, i quali poi sono strutturati in *branch* come *main*, *develop* e *branch* specifici per implementare funzionalità. Questo strumento consente di mantenere il codice stabile e versionato, migliorando lo sviluppo in parallelo.

1.4 Propensione all’innovazione

VisioneImpresa non dispone di un reparto dedicato alla ricerca e sviluppo, ciononostante l’innovazione e l’aggiornamento rappresentano un elemento fondamentale sulla quale si basa la filosofia aziendale. Essendo un’azienda che opera da oltre quarant’anni nel settore dell’*Information Technology*, ha avviato un processo di transizione tecnologica di ampio raggio, tra le iniziative più significative c’è la progressiva migrazione dei sistemi *ERP_G* sviluppati in FoxPro verso ambienti più moderni e performanti. Questo processo per complessità e dimensione richiede una pianificazione pluriennale e il coinvolgimento graduale di un gran numero di risorse. In parallelo a questa transizione, VisioneImpresa considera i progetti di *stage* un’opportunità concreta per il progresso. Durante il mio tirocinio, ho avuto modo di contribuire attivamente allo sviluppo di progetti finalizzati alla modernizzazione. In particolare mi è stato comunicato che uno dei due progetti a cui ho avuto modo di lavorare è destinato a integrarsi con il nuovo gestionale attualmente in fase di sviluppo, pensato per sostituire

CAPITOLO 1. CONTESTO AZIENDALE

l'attuale *ERP_G* con una soluzione più moderna e tecnologicamente avanzata. Inoltre, l'azienda incoraggia gli stagisti a proporre nuove idee e soluzioni tecniche, introducendo anche nuove tecnologie in autonomia. Questa libertà consente di testare nuove soluzioni in un ambiente controllato, favorendone la possibile introduzione nei processi aziendali.

Capitolo 2

Definizione del percorso di stage

2.1 Il rapporto tra azienda e *stage*

Come già accennato nella sotto-sezione 1.4, VisioneImpresa investe molte risorse nei progetti di *stage*, ritenendo che questi siano uno strumento concreto ed efficace per promuovere l’innovazione e la sperimentazione. Durante il mio periodo di tirocinio erano presenti, oltre a me, altri quattro *stagisti*. Quasi tutti erano impegnati in progetti differenti e ritenuti rilevanti dall’azienda. Questo dato testimonia quanto VisioneImpresa ritenga importante questo tipo di rapporto con gli studenti e gli enti del territorio. In quanto società *bene-fit* l’azienda persegue anche obiettivi sociali, tra i quali la collaborazione con scuole e università attraverso progetti orientati all’innovazione. Gli *stagisti* provengono principalmente da corsi di laurea triennali in informatica o ingegneria informatica, portando con loro le conoscenze accademiche acquisite negli anni. Lo *stage* ha una durata di 300-320 ore un tempo che consente allo studente di contribuire in maniera concreta allo sviluppo di soluzioni aziendali, e allo stesso tempo conferisce un’esperienza significativa dal punto di vista professionale. I progetti presentati dall’azienda non hanno un obiettivo prettamente formativo, ma corrispondono a necessità reali dell’azienda, emerse tramite il confronto con clienti o con *team* di sviluppo interni. I tirocinanti si trovano quindi immersi nel contesto aziendale con l’obiettivo di affrontare problematiche concrete, analizzare e riscrivere codice esistente e approfondire tecnologie, situazioni spesso

nuove per uno studente. Alla fine del periodo ciascun tirocinante presenta il lavoro svolto al *team* di sviluppo più appropriato e al referente aziendale, questa fase è cruciale per rendere possibile la valutazione del lavoro svolto e spiegare il progetto da un punto di vista tecnico ponendo le basi per una possibile futura integrazione di esso nei prodotti aziendali. Inoltre VisioneImpresa utilizza il periodo di *stage* anche come momento per valutare il profilo dello studente, sia a livello tecnico che a livello personale. In diversi casi, il tirocinio si è trasformato in un'opportunità concreta di inserimento nel contesto lavorativo, rendendo possibile agli studenti di proseguire il percorso all'interno dell'azienda.

2.2 Proposte progettuali dell'azienda

Il primo contatto con VisioneImpresa è avvenuto grazie l'evento StageIT 2025, organizzato dall'università di Padova per mettere in contatto studenti alla ricerca di un tirocinio e aziende del territorio che propongono offerte di *stage* in ambito informatico. Durante l'evento ho avuto modo di svolgere numerosi colloqui con aziende interessanti, che proponevano una vasta scelta di progetti distribuiti su molti ambiti applicativi. Nonostante durante l'evento no abbia svolto un colloquio diretto con VisioneImpresa, sono stato contattato nei giorni seguenti tramite *email*. In questo messaggio era presente un invito a leggere le proposte progettuali dell'azienda e nel caso ne fossi stato interessato la possibilità di effettuare un colloquio direttamente in sede per approfondire la conoscenza. In seguito a questo primo contatto ho svolto un riunione dove mi è stato possibile conoscere il referente aziendale, approfondire le proposte progettuali e avere un primo approccio con l'ambiente di lavoro. Durante questa visita mi sono presentati numerosi progetti di *stage*, tutti con obiettivi e ambiti di applicazione differenti, la tabella 2.1 contiene una visuale generale delle proposte di progetto dell'azienda.

Tabella 2.1: Proposte di progetto presentate da VisioneImpresa.

Nome progetto	Breve descrizione
VisionAI	Progetto mirato all'espansione di un assistente virtuale esistente in grado di rispondere in linguaggio naturale, interrogando il <i>database</i> aziendale. Il tirocinante lavorerà al fine di ottimizzare le <i>API</i> già esistenti e crearne di nuove, inoltre lavorerà alla creazione di un <i>agent</i> intelligente.
<i>WebApp</i> attivazione istanze	Sviluppo di una <i>webapp</i> interna all'azienda che ha l'obiettivo di automatizzare le configurazioni di nuove installazioni <i>software</i> , generazione cartelle, <i>database</i> , utenti, credenziali, PDF, e invio automatico di mail.
Catalogo multimediale moviSELL	Sviluppo di un modulo integrato per l' <i>app</i> moviSELL. L'obiettivo è la creazione di un catalogo interattivo con immagini e video dove l'utente può svolgere il suddetto catalogo, selezionare i prodotti e inserirli nel carrello.
Versione lite moviREP	Creazione di una versione <i>smartphone</i> dell' <i>app</i> moviREP, che condivide i dati con la versione per tablet ma offre un'interfaccia più essenziale.
Siti <i>web</i> a basso impatto ambientale	Ottimizzazione di siti WordPress e PrestaShop per andare a ridurre l'impatto ambientale, andando a diminuire i consumi energetici e migliorando le prestazioni.
moviEXPENSE con OCR	Aggiornamento dell' <i>app</i> moviEXPENSE, facendo la migrazione a React Native e aggiungendo un sistema OCR per l'estrazione automatica di dati da scontrini.
moviSELL Web	Progettazione e sviluppo dell' <i>app</i> nativa per dispositivi <i>mobile</i> moviSELL. L'obiettivo è andare a replicare le funzionalità principali della versione <i>tablet</i> , connettendola con il <i>database</i> .
<i>Report Designer</i> moviSELL/moviREP	Sviluppo di un modulo per la creazione di <i>report</i> personalizzati in PDF per le <i>app</i> moviSELL e moviREP, utilizzando strumenti integrati nell'applicazione.

Nome progetto	Breve descrizione
Gestione multilingua	Aggiungere la funzionalità di supporto multilingua delle descrizioni dei prodotti e delle tabelle correlate all'interno delle app moviSELL e moviREP, adattando il testo in automatico alla lingua dell'utente.

Tabella 2.1: Panoramica delle proposte progettuali.

Questa offerta progettuale così ampia testimonia in quanti ambiti tecnologici differenti lavora VisioneImpresa e sottolinea il valore conferito ai progetti di *stage*, tutti orientati a fornire risposte a esigenze concrete dell'azienda.

2.3 Valutazione e presentazione del progetto

2.3.1 Valutazione e scelta

Ho scelto il progetto di *stage* in seguito a una attenta valutazione delle proposte disponibili, l'azienda ha presentato un ampio ventaglio di proposte progettuali spaziando in molti contesti di sviluppo differenti. Tra i progetti disponibili ho scelto, in accordo con il *tutor* aziende, il progetto 1, VisionAI, incentrato sullo sviluppo di un sistema intelligente in grado di fornire risposte in linguaggio naturale a domande poste dagli utenti, interrogando in modo diretto il *database* aziendale. Ho deciso di intraprendere questo progetto principalmente per un forte interesse personale per il settore dell'intelligenza artificiale e della sua applicazione. In particolare, sono molto interessato nell'osservare da vicino come queste tecnologie trovino applicazioni in ambienti lavorativi, anche in realtà di medie-piccole dimensione. Mi interessa inoltre il comprendere quali sono i processi decisionali che portano all'adozione di sudette tecnologie in sistemi funzionanti, e di come le persone possono rapportarsi con essi.

2.3.2 Presentazione del progetto

Il progetto selezionato nasce dal concetto base di introdurre soluzioni intelligenti all'interno dei propri prodotti *software*, con l'obiettivo di mantenere le

applicazioni e i servizi offerti competitivi all'interno di un settore in continua evoluzione. L'idea è quella di semplificare l'accesso alle informazioni, integrando tecnologie che permettono l'utilizzo del linguaggio naturale e inoltre automatizzare processi decisionali complessi. Il lavoro è stato concepito come progetto unitario, articolato in due componenti distinte che però condividono le stesse finalità. Il primo componente riguarda VisionAI, un assistente virtuale, del quale alla base c'è già un prototipo funzionante, capace di interagire via chat con l'utente per fornire risposte a domande di base, interrogando il *database* per ottenere le informazioni. Lo scopo di questa sezione è sviluppare e perfezionare le *API G* su cui VisionAI si basa, utilizzate per dialogare con il *database*, aumentando così il raggio di azione dell'assistente e consentendogli di fornire risposte a domande più complesse. La seconda parte riguarda lo sviluppo di un *agent*, questo modulo basandosi sempre su dati presenti nel *database*, ha come scopo quello di dare supporto alle aziende che operano nel campo dell'assistenza tecnica, organizzando in modo efficace l'ordine degli interventi in un determinato periodo, cercando di ottimizzare l'assegnazione dei tecnici agli interventi da svolgere. In conclusione, sebbene il progetto è diviso in due sezioni distinte, mantiene una visione coerente e unitaria, orientata ad integrare nei prodotti aziendali strumenti "intelligenti".

2.3.3 Vincoli generali del progetto

Come in tutti i progetti nel mondo reale, anche il progetto di stage è sottoposto a vincoli esterni, imposti sia dal contesto aziendale sia dalla pianificazione temporale del tirocinio. Ovviamente questi vincoli hanno un influenza nelle scelte tecnologiche e organizzative prese nelle varie fasi di sviluppo.

Vincoli temporali

CAPITOLO 2. DEFINIZIONE DEL PERCORSO DI STAGE

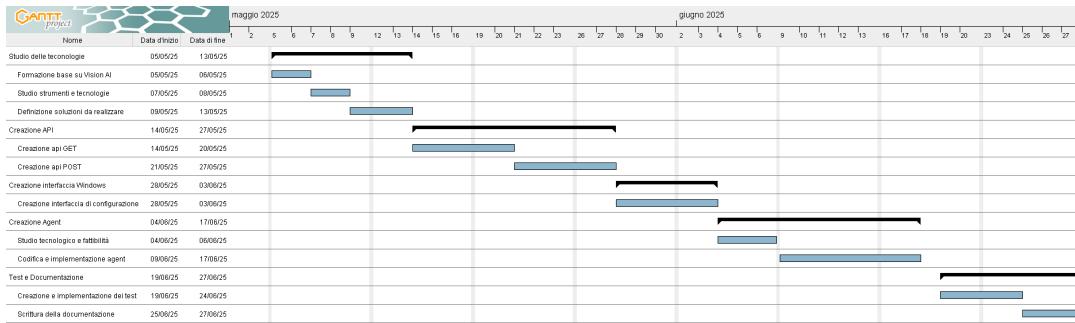


Figura 2.1: Pianificazione temporale dello *stage*

La struttura del lavoro all'interno del periodo di *stage*, che è determinato da un monte ore limitato, è stata definita in collaborazione con il *tutor* aziendale, suddividendo il tempo totale in sotto-periodi come illustrato nell'immagine 2.1. Questa divisione è avvenuta prima dell'inizio del tirocinio e presentata nel piano di lavoro. Lo *stage* è stato diviso in queste sezioni:

- **Studio degli applicativi e del database** (una settimana), l'obiettivo di questo periodo era l'apprendimento delle tecnologie e l'analisi del codice già presente, inoltre una prima formazione sulla struttura del *database*.
- **Nuove API_G** (due settimane), in questo periodo è stata pianificata la creazione delle nuove *API_G* e perfezionare quelle già esistenti.
- **Interfaccia di configurazione** (una settimana), il periodo indicato allo sviluppo dell'interfaccia di connessione a VisionAI, creando un *form* per la connessione al *database* e alla *chat*.
- **Agent** (due settimane), l'obiettivo di questo periodo è la progettazione e lo sviluppo dell'*agent* per la pianificazione automatica degli interventi.
- **Test e documentazione** (una settimana e mezza), questo il periodo dedicato alla stesura della documentazione tecnica e allo svolgimento dei *test*.

Vincoli tecnologici generali

L'azienda non ha imposto vincoli predefiniti a livello di tecnologie per la comunicazione e la documentazione, ma inserendosi in un contesto lavorativo già avviato comporta l'adozione di strumenti e metodologie già in uso.

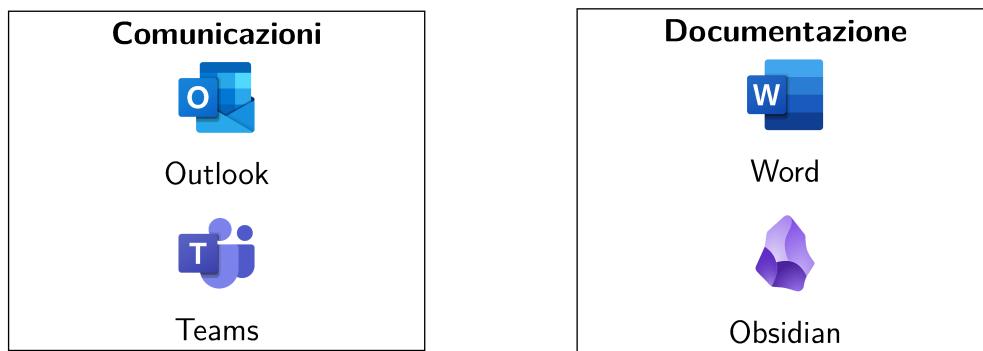


Figura 2.2: Tecnologie utilizzate per la comunicazione e la documentazione.

La figura 2.2 mostra le tecnologie utilizzate per le comunicazioni:

- Outlook: strumento di comunicazione utilizzato per lo scambio di posta elettronica con il referente aziendale o membri dei *team* di sviluppo, per la gestione degli appuntamenti in calendario e per la pianificazione di incontri e riunioni.
- Microsoft Teams: strumento utilizzato in occasione della riunione mensile per rendere possibile la partecipazione di elementi non presenti direttamente in sede e per svolgere incontri di informazione organizzati dal *tutor* aziendale con persone esterne.

Gli strumenti per la documentazione sono i seguenti:

- Microsoft Word: utilizzato principalmente per consultare e integrare la documentazione fornitaci dal *tutor* aziendale e per integrare la seguente documentazione con i risultati dei *test* svolti sul prodotto sviluppato.
- Obsidian: strumento utilizzato per redigere la documentazione tecnica in modo semplice e leggibile, utilizzato per organizzare appunti, annotazioni e idee durante il periodo di *stage* tramite *markdown*.

2.4 Vincoli tecnologici e obiettivi aziendali

Il progetto come ho descritto nella sotto-sezione 2.3.2 è stato diviso in due componenti differenti, ognuno dei quali ha presentato vincoli tecnologici e obiettivi aziendali distinti.

2.4.1 Assistente VisionAI

L’azienda, come già detto, non impone vincoli tecnologici fissi, lasciando piena libertà nella decisione di linguaggi, *framework* e ambienti di sviluppo. Le scelte a livello di tecnologie da utilizzare quindi sono prese in maniera autonoma dallo stagista principalmente in base a efficacia e familiarità. Nel caso specifico del progetto VisionAI, tuttavia, esisteva già una base di codice preesistente, sviluppata in C#, e gli obiettivi concordati prevedevano espressamente l’utilizzo di questo linguaggio. In aggiunta durante lo sviluppo del prototipo iniziale erano già state prese anche altre scelte, è quindi stato opportuno mantenere lo *stack* tecnologico già in uso per coerenza, visualizzabile nelle figure 2.3 e 2.4

Ambienti di sviluppo

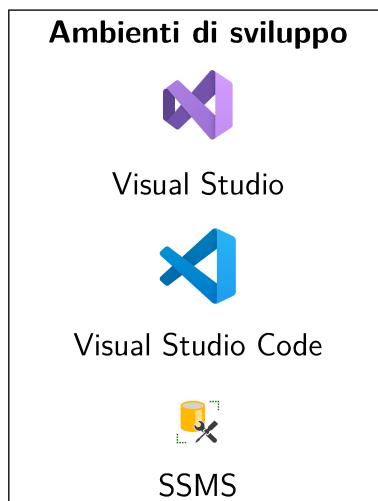


Figura 2.3: Ambienti di sviluppo per il modulo VisionAI.

- Visual Studio: strumento utilizzato per la scrittura e il mantenimento del codice delle *API* tramite il supporto nativo al *framework* .NET. scelto per i numerosi strumenti che fornisce a supporto dello sviluppo come strumenti di *debug* e la gestione dei pacchetti.
- Visual Studio Code: utilizzato anche questo per lo sviluppo delle *API*, scelto principalmente per la familiarità con l’ambiente, per rendere più efficiente la scrittura del codice in determinati momenti.

- SQL Server Management Studio (SSMS): utilizzato per eseguire interrogazioni al *database* relazione, supporta la scrittura di più *query* in contemporanea e utilizzato per svolgere operazioni nel *database* a supporto dello sviluppo.

Framework e software

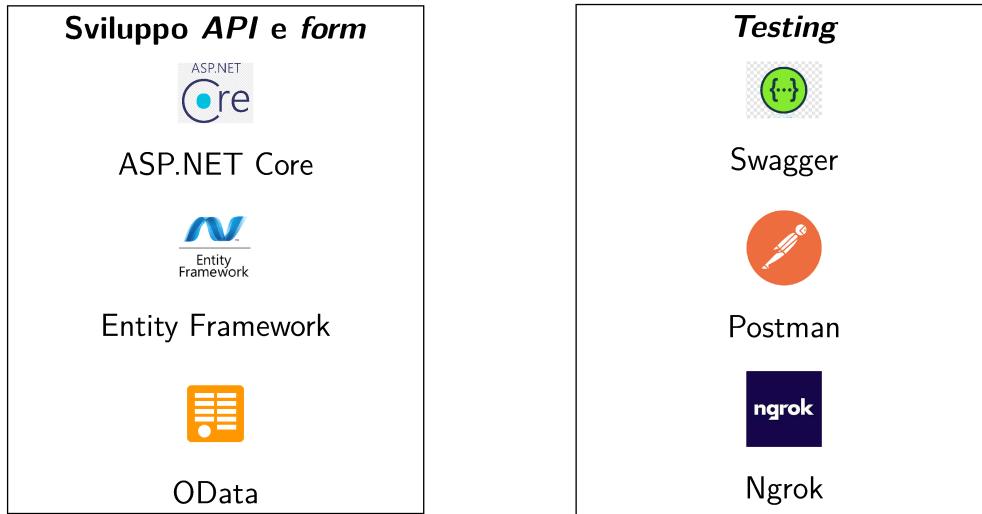


Figura 2.4: Tecnologie utilizzate per il modulo VisionAI.

- ASP.NET Core: *framework* utilizzato per la scrittura delle *API_G RESTful*, permesso di sviluppare *endpoint_G*(url a cui un client può inviare richieste per eseguire operazioni sui dati) e mantenere alte le prestazioni
- Entity Framework Core: strumento *ORM_G*, tecnica che consente di interagire con *database* in linguaggio di programmazione anzi che utilizzare *query*, utilizzato per l'accesso ai dati.
- OData: Utilizzato per integrare funzionalità all'interno della *API_G* direttamente via URL, permettendo all'intelligenza artificiale di formulare *query* con filtri direttamente all'interno dell'URL della chiamata

In merito alle tecnologie a supporto della fase di *testing* delle *API_G* sono state:

- Swagger: utilizzato per la generazione automatica della documentazione delle *API_G*, tramite la sua interfaccia inoltre è possibile esplorare le chiamate ed effettuare *test* facilitando il *debug*.

- Postman: tecnologia utilizzata per effettuare *test* mirati, impiegato principalmente per *testare* la funzionalità delle *API_G* POST.
- Ngrok: utilizzato per esporre in modo temporaneo il *backend* e le *API_G* alla rete esterna, questo strumento ha permesso di *testare* le chiamate all'interno della *chat* senza dover pubblicare il servizio su un *server* remoto.

Obiettivi aziendali

La tabella 2.2 contiene gli obiettivi concordati con il *tutor* aziendale previsti per la sezione corrente. Gli obiettivi sono categorizzati in obbligatori, lettera O, desiderabili, lettera D, e facoltativi, lettera F.

Codice	Descrizione
O01	Sviluppo interfacce Visual Studio con librerie DevExpress
O02	Sviluppo delle <i>API_G</i> di <i>back-end</i> in linguaggio C#
D01	Creazione in autonomia della documentazione e del piano di <i>test</i>

Tabella 2.2: Tabella obiettivi aziendali sezione VisionAI

2.4.2 Modulo di pianificazione interventi

La sezione del progetto incentrata allo sviluppo dell'*agent* per la pianificazione automatica degli interventi nel contesto di VisionAssistance è stato iniziato da zero, quindi ha concesso una libertà assoluta dal punto di vista tecnologico, la figura 2.5 mostra le scelte relative agli ambienti di sviluppo mentre la figura 2.6 mostra le scelte relative ai linguaggi e strumenti *software* prese per questo modulo.

Ambienti di sviluppo



Figura 2.5: Ambienti di sviluppo per il modulo assegnazione interventi.

- Visual Studio Code: ambiente scelto per la scrittura del codice python, utilizzato per la familiarità con l'ambiente di sviluppo e la disponibilità di estensioni utili.
- SQL Server Management Studio (SSMS): impiegato per eseguire interrogazioni e operazioni sul *database* aziendale, utile per la visualizzazione dei dati e verificare l'integrità dei dati utilizzati.

Linguaggi e strumenti *software*

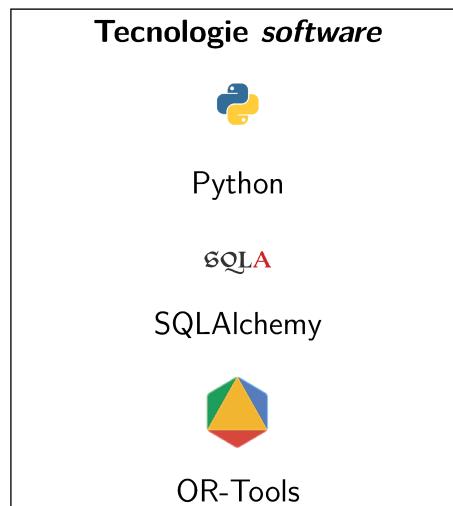


Figura 2.6: Tecnologie utilizzate per il modulo di pianificazione interventi.

- Python: Linguaggio di programmazione principale utilizzato per lo sviluppo di questo progetto, scelto per la sintassi semplice, la familiarità e per la compatibilità con OR-Tools.

- SQLAlchemy: libreria *ORM* utilizzata per gestire l'accesso ai dati. Infatti, permette di scrivere *query* in linguaggio SQL direttamente all'interno del codice per facilitare l'accesso ai dati.
- Google OR-Tools: *solver* per la programmazione vincolata, è stato utilizzato per risolvere il problema dell'assegnazione ottimale degli interventi.

Obiettivi aziendali

La tabella 2.3 contiene gli obiettivi concordati con il *tutor* aziendale previsti per la sezione del progetto di stage in questione. Il codice identificativo è il medesimo della sezione precedente.

Codice	Descrizione
F01	Creazione di un <i>agent</i> per la pianificazione degli interventi in VisionAssistance

Tabella 2.3: Tabella obiettivi aziendali per sezione pianificazione interventi

2.5 Obiettivi personali

La tabella 2.4 mostra invece gli obiettivi personali stipulati prima dell'inizio dello *stage*, la notazione degli obiettivi è la stessa delle tabelle precedenti.

Codice	Descrizione
O01	Imparare nuove tecnologie e <i>framework</i> legati all'ambiente .NET e al linguaggio C#
O02	Collaborare attivamente con il <i>team</i> , mantenendo una comunicazione efficace e continua
O03	Svolgere le attività nel rispetto delle scadenze concordate.
D01	Ampliare il mio bagaglio tecnico e di <i>soft-skill</i> in un contesto strutturato

Tabella 2.4: Obiettivi personali dello *stage*.

Capitolo 3

Il progetto di stage

3.1 Pianificazione delle attività

Come ho già anticipato nella sotto-sezione 2.3.2 abbiamo suddiviso il progetto di *stage* in due sezioni separate, entrambe rivolte a migliorare le soluzioni aziendali fornendo soluzioni "intelligenti". La prima fase del periodo ha riguardato l'estensione delle funzionalità dell'assistente virtuale interno al gestionale chiamato VisionAI mentre la seconda ha riguardato lo sviluppo di un sistema di assegnazione automatica con l'obiettivo di essere integrato all'interno di VisionAssistance. La pianificazione delle attività non ha previsto l'utilizzo di *software* specifici, abbiamo gestito il lavoro in maniera flessibile ponendo alla base la comunicazione con il *tutor* aziendale, che ha avuto un ruolo importante sia per quanto riguarda l'organizzazione delle attività sia nel supporto formativo iniziale. Lo *stage* è stato svolto in coppia con un altro studente, con il quale ho condiviso lo spazio di lavoro e tutte le attività. Abbiamo collaborato costantemente avendo lavorato in maniera combinata a quasi tutte le porzioni del progetto, affrontando in maniera condivisa le problematiche e contribuendo entrambi alla buona riuscita del progetto. L'unica distinzione ha riguardato la fase di sviluppo del modulo di assegnazione automatica, in cui abbiamo seguito lo sviluppo di parti diverse ma complementari, io mi sono occupato della *pipeline* di raccolta e strutturazione dei dati, mentre il mio collega si è occupato dell'algoritmo di assegnazione. Nonostante questa divisione abbiamo man-

tenuto una comunicazione costante che ha permesso l'integrazione tra le due componenti. L'approccio per le attività può essere definito incrementale, con cicli della durata breve. Ogni inizio settimana veniva svolto un breve incontro con il *tutor* aziendale, questi momenti seppur brevi erano molto utili per avere un *feedback* sul lavoro svolto e ricevere linee guida per procedere al passo successivo. Oltre agli incontri di persona, il *tutor* forniva le attività anche tramite documentazione tecnica, contenuti nuove funzionalità da implementare e note di contesto come *query* di esempio inserite per rendere più facile la comprensione della struttura del *database* e le relazioni tra le tabelle. Ognuna delle due fasi è stata preceduta da un periodo di studio, necessario per familiarizzare con le tecnologie da utilizzare, comprendere i sistemi esistenti e l'infrastruttura *software* aziendali. Per entrambi i casi questo periodo di studio è durato circa una settimana è stato fondamentale per poter procedere in maniera organizzata allo sviluppo vero e proprio. Durante queste fasi di molto importante è stata la creazione di appunti che sono stati utili in fase di sviluppo.

3.2 Sviluppo di API REST per VisionAI

In questa sezione descrivo la sezione di progetto finalizzata allo sviluppo delle *API_G REST_G* e l'ampliamento delle funzionalità dell'assistente interno chiamato VisionAI. Il lavoro si è concentrato su piccole migliorie delle *API_G* già presenti e l'integrazione di nuovi *endpoint_G*, rendendo il sistema capace di accedere a nuovi dati del *database* aziendale.

3.2.1 Contesto applicativo e funzionamento di VisionAI

Al fine di comprendere appieno le scelte progettuali e le modalità di sviluppo è importante analizzare l'architettura preesistente e il funzionamento di VisionAI, così da dare contesto al lavoro svolto. L'infrastruttura *software* si basa sull'integrazione tra un motore *LLM_G*, un modello di intelligenza artificiale avanzato che può comprendere e generare linguaggio naturale, gestito tramite

un interfaccia sviluppata da DevLab e un *backend* strutturato a microservizi *REST_G*.

Architettura generale

VisionAI è un assistente intelligente integrato in modo diretto all'interno del gestionale aziendale, ideato per rispondere a domande in linguaggio naturale e restituire informazioni prelevate da uno o più *database* interni. L'utente interagisce tramite una *chat* e il motore sottostante si occupa:

- analizzare e interpretare il significato della domanda,
- visualizzare le *API_G* a disposizione e selezionare quella più adatta,
- costruire la chiamata *HTTP_G* (protocollo di comunicazione per trasferimento dati sul web) con i parametri corretti per l'*API_G* in questione,
- trasformare la risposta ottenuta in testo comprensibile all'utente.

Componenti del sistema

1. Motore realizzato da DevLab

Il punto centrale del sistema è un motore *LLM_G* chiamato "mentiscrm". Questo strumento permette di configurare in modo dettagliato il comportamento dell'assistente con la possibilità di creare il *prompt* principale che viene inviato in ogni iterazione fornendo il contesto necessario, definire le *API_G REST_G* che l'assistente può utilizzare, sarà poi il motore che gestisce la richiesta *HTTP_G*. Altra funzione importante è la possibilità di dare contesto tramite una descrizione alle singole *API_G* per permettere all'*LLM_G* di chiamare quella giusta e nel modo corretto.

2. *Frontend* realizzato da DevLab

Interfaccia dell'assistente, si presenta come una classica *chat LLM_G*

3. *Backend* VisionAI

Il *backend* contenete le *API_G*, espone una serie di *endpoint_G REST_G*ful e contiene il codice sorgente scritto in C# delle *API_G*

4. Interazione con LLM_G

Il motore realizzato da DevLab funziona da interfaccia tra la *chat* dove scrive l'utente e il modello di LLM_G in questo caso GPT-4-mini.

Panoramica del flusso operativo

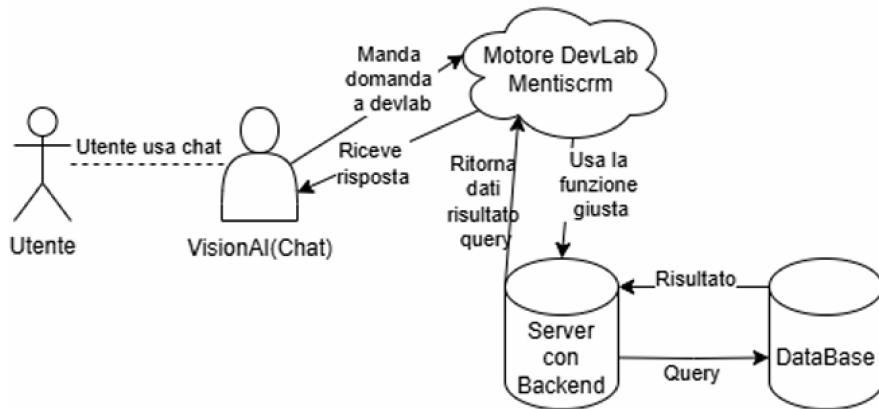


Figura 3.1: Flusso operativo di VisionAI.

L'immagine 3.1 rappresenta il flusso di funzionamento di VisionAI, inizia con l'inserimento di una domanda in linguaggio naturale da parte dell'utente in *chat*. Il motore di DevLab elabora la richiesta e utilizzando il modello GPT trova l' API_G più adatta per la richiesta, successivamente costruisce in modo dinamico l'URL della chiamata. Il *backend* elabora la richiesta, accede al *database* e restituisce i dati in formato JSON. Per concludere GPT interpreta il risultato e crea una risposta in linguaggio naturale.

3.2.2 Analisi dei requisiti

L'analisi rappresenta uno dei momenti più importanti nell'ingegneria del *software*, questa attività costituisce il momento in cui le idee e i bisogni vengono trasformate in specifiche operative che guideranno in seguito la fase di progettazione, sviluppo e validazione. L'analisi dei requisiti in particolare ha come scopo quello di identificare le funzionalità che il sistema deve offrire per rispondere alle necessità individuate dall'azienda. Nel contesto di VisionAI questa attività è servita a individuare quali funzionalità implementare sotto forma

di *API_G REST_G*, quali invece dovessero essere corrette e quali requisiti vadano considerati prioritari e quali invece secondari.

All'interno della seguente tabella 3.1 ho riportato i requisiti individuati nel corso dell'analisi, il codice del requisito viene creato in base al seguente metodo:

$$\mathbf{R}[\mathbf{F}/\mathbf{O}][\mathbf{O}/\mathbf{D}/\mathbf{F}]\text{-NN}$$

dove:

- **R** indica che si tratta di un requisito;
- **F/O** specifica la tipologia del requisito:
 - **F**: Funzionale – descrive un comportamento o una funzione del sistema;
 - **O**: Operativo – descrive vincoli tecnici o ambientali;
- **O/D/F** specifica il grado di priorità:
 - **O**: Obbligatorio – deve essere implementato;
 - **D**: Desiderabile – migliora il sistema ma non è essenziale;
 - **F**: Facoltativo – implementabile solo in caso di tempo o risorse aggiuntive.
- **NN** è un numero progressivo per identificare univocamente ogni requisito.

Codice	Descrizione
RF-O-01	L' <i>API_G</i> deve permettere l'inserimento di nuovi clienti o fornitori, assegnando automaticamente un codice identificativo CodCliFor secondo la strategia di codifica aziendale.
RF-O-02	Sviluppo di un' <i>API_G</i> per controllare lo stato di evasione (completa, parziale o assente) degli ordini clienti o fornitori .

CAPITOLO 3. IL PROGETTO DI STAGE

Codice	Descrizione
RF-O-03	Sviluppo di un' <i>API G</i> per restituire il saldo aggiornato dei conti correnti aziendali per uno specifico esercizio contabile.
RF-O-04	Sviluppo di un' <i>API G</i> per l'elenco degli interventi tecnici previsti ma ancora da svolgere, con filtri per data, cliente, zona, tipo.
RF-F-05	<i>API G</i> per recuperare richieste di assistenza filtrabili per stato, cliente, tecnico e periodo.
RF-O-06	Realizzazione di un'interfaccia grafica Windows (Windows Forms) per configurazione, connessione e avvio dell'applicazione VisionAI.
RF-D-07	Copia strutturata di un documento esistente con aggiornamento automatico di quantità, date e intestazioni.
RF-D-08	Generazione automatica di documenti "evasione" a partire da ordini o bolle precedenti.
RF-F-09	Estensione delle <i>API G</i> di manutenzione per aggregare le attività in base alla sede del cliente (destinazione).
RF-F-10	Visualizzazione, inserimento e modifica di informazioni anagrafiche dei dipendenti interni o collegati a clienti/fornitori.
RF-O-11	Rendere disponibile la funzione di ricerca in rete di dati da parte dell'assistente.
RF-O-12	Le <i>API G</i> POST devono popolare correttamente i campi dati di TimeIns e UserIns
RF-O-13	Miglioramento delle operazioni GET e POST sulla tabella CliFor per la corretta assegnazione dei campi CodPdcPatr e CodPagam , distinguendo tra Cliente e Fornitore.

Codice	Descrizione
RO-O-13	Garantire a tutte le <i>API_G</i> sviluppate una corretta descrizione specificando l'intento dell' <i>API_G</i> e la struttura della chiamata per il corretto funzionamento.

Tabella 3.1: Tabella del tracciamento dei requisiti.

3.2.3 Progettazione

La progettazione del sistema ha rappresentato una fase cruciale per garantire coerenza con la struttura esistente, in quanto, subentrando in un progetto già avviato è stato necessario svolgere un analisi dell'architettura esistente, capendo le logiche interne per poi sviluppare nuove componenti coerenti a i vincoli strutturali.

Struttura a livelli e *pattern*

L'architettura delle *API_G* del sistema VisionAI si basa su un modello a strati, visualizzabile nell'immagine 3.2, progettato per separare le responsabilità funzionali e garantire modularità e scalabilità. La struttura segue un approccio ispirato ai principi della *Clean architecture_G* implementato tramite l'utilizzo combinato di *controller*, servizi, *repository* e *DTO_G*, un *design pattern* utilizzato per trasferire dati in un applicazione *software*. Sebbene non sia una classica architettura MVC (*model-view-controller*), essendo priva di componenti di visualizzazione, la struttura può essere descritta come una *Controller-Service-Repository* con l'integrazione di *model* e *DTO_G*, dove:

- il *controller* gestisce la comunicazione esterna e funge da *entry point* per le richieste *HTTP_G* gestendo il *routing*;
- il *Service/Repository*, in questo progetto appaiono in un'unica struttura, si occupa della logica applicativa e l'accesso ai dati;
- i *Model* rappresentano le strutture dati;
- i *DTO_G* consentono il trasferimento controllato e strutturato delle informazioni.

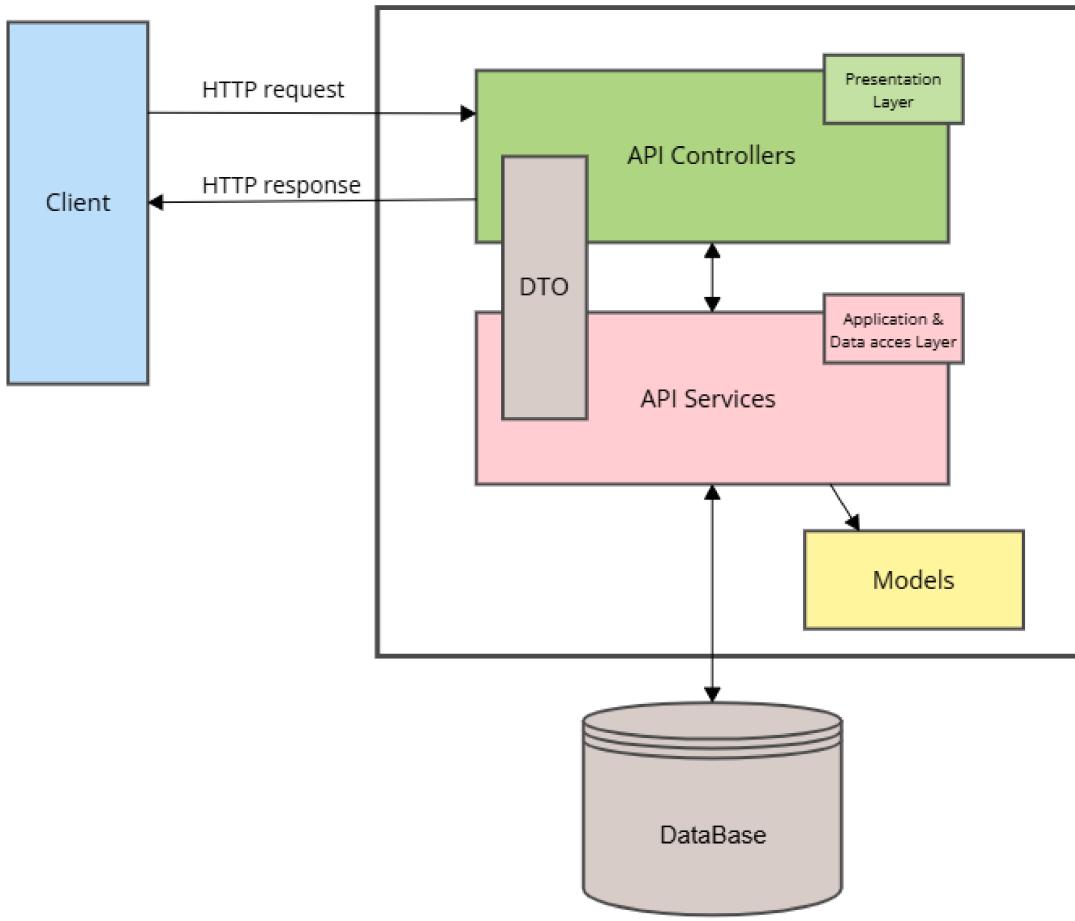


Figura 3.2: Architettura *API G*.

Questa architettura è stata scelta per questo tipo di struttura per vari motivi, per creare *API G REST G* facilmente interrogabili da sistemi *AI*, per garantire una separazione netta delle responsabilità tra i vari livelli aiutando la manutenzione e l'estensione del sistema, proteggere i dati separando i modelli interni con oggetti esposti come i *DTO G*. Nel contesto del progetto questi sono i livelli principali:

1. *Presentation layer - Controllers*

I *controller* sono collocati all'interno della cartella **Controllers** e fungono da punto di contatto tra la logica interna del sistema e l'esterno, nel caso della nostra applicazione il motore di DevLab e il *chatbot*. Dentro l'applicazione VisionAI ogni *controller* viene associato a una funzionalità o a un gruppo funzionale, come **CliForController** che gestisce l'accesso alle anagrafiche clienti/fornitori, o come **BancheController** che si occupa dell'accesso all'elaborazione dei saldi e dei conti correnti aziendali. Ogni

controller viene incaricato di svolgere diverse operazioni, ricevere richieste *HTTP_G* dal motore GPT in formato *GET* o *POST*, valida i parametri in ingresso controllando che siano corretti rispetto alla logica definita, inoltre la richiesta al servizio corrispondente e restituisce una risposta in formato JSON. In questo progetto non esiste una interfaccia utente, il *controller* funge solo da *entry point* per le richieste esterne.

2. *Application & data access layer - Services / Repository*

Nel progetto VisionAI la logica applicativa e il *data access layer* sono stati accorpati in un'unica struttura contenuta nella cartella **Services**, in cui ogni *file* è dedicato in un contesto funzionale differente. Questo approccio ci fa uscire dagli *standard* di un *design pattern* classico, andando a creare una struttura ibrida orientata alla praticità e alla velocità di sviluppo. Ciascun *file Repository*, questo il nome dei *file* contenuti nella cartella, ad esempio **CliforRepository** contiene sia metodi che si occupano della *business logic* specifica del dominio (implementazione di parametri e filtri), sia dell'interazione diretta con il *database* utilizzando LINQ o Drapper. Come già detto ogni *file* è focalizzato su un'area specifica del contesto applicativo, questo consente di avere *repository ad hoc* per ogni gruppo di *API_G*.

3. *Data definition e transfer - Model / DTO_G*

Nel progetto VisionAI non viene definito un *layer* apposito per i *model*, come accade nell'architettura MVC (*Model-View-Controller*) classica. Tuttavia, i *model* svolgono un ruolo essenziale all'interno del nostro sistema, servono a definire come sono strutturati i dati all'interno del *database*, fornendo una rappresentazione diretta delle tabelle. Queste classi vengono richiamate all'interno dei *services* per permettere l'interazione con i dati e comprenderne la struttura. A supporto di questa operazione è presente il *file ApplicationDbContext* che definisce il contesto del *database* contenendo le relazioni delle entità interessate, facilitando la navigazione tra le tabelle. Oltre ai *model* il progetto utilizza anche *DTO_G*, non in maniera globale ma in maniera specifica in alcuni casi, non tutti gli *enpoint*

infatti ne fanno uso. Sono stati introdotti per i casi più articolati dove è necessario definire una struttura di risposta standardizzata, controllare l'esposizione evitando la fuoriuscita di dati sensibili e aggregare dati provenienti da tabelle diverse

4. *Database layer*

Questo *layer* rappresenta lo stato più profondo dell'architettura, in questo strato sono presenti i dati aziendali effettivi. Questi dati sono distribuiti su diversi *database*, tutti in formato SQL Server e preesistenti, i *database* utilizzati sono stati *VisionEsempio*, *VisionCommon* e *VisionAnt*. L'accesso alle basi di dati è configurabile nel sistema in modo dinamico tramite l'utilizzo del file `appsetting.json`, il quale contiene le connessioni ai *database* aziendali e i parametri ambientali personalizzabili. Durante l'esecuzione l'utente può modificare la connessione ai *database* anche tramite l'utilizzo dell'interfaccia Windows *form* sviluppata.

La progettazione delle *API_G* oltre dell'infrastruttura già esistente ha dovuto tenere conto di un'altra variabile molto importante, ovvero che queste *API_G* non vengono chiamate da un *frontend* programmato per eseguire chiamate standardizzate ma da un *LLM_G* che effettua chiamate solamente con il contesto fornитogli da un *prompt*. Questo elemento ha comportato scelte progettuali di grande impatto. Innanzitutto le *API_G* sono progettate per essere facilmente interpretabili dal modello mantenendo sempre parametri chiari e assenza totale di ambiguità per i nomi dei campi dati. Inoltre il contesto fornito all'*AI*, in questo caso GPT-4, può essere caricato solamente tramite prompt di testo dalla dimensione limitata, questa caratteristica porta all'impossibilità di costruire *API_G* esageratamente complesse o con troppi campi dati per effettuare chiamate molto precise. Diventa quindi di fondamentale importanza la capacità di descrivere il funzionamento delle *API_G* in modo riassuntivo e specifico, per fornire il *prompt* corretto e consentire al modello *LLM_G* di effettuare una chiamata corretta. Un altro aspetto di estrema importanza progettuale è la robustezza contro *input* imprevedibili, il sistema *backend* quindi deve essere robusto nel controllo degli *input* nelle *API_G* POST, effettuando la normalizza-

zione dei dati, gestendo casi di dati mancanti o inadeguati e gestendo messaggi di errore o comportamenti imprevisti.

Progettazione Windows *form*

Nel contesto dell'applicazione VisionAI è stato introdotto un componente accessorio riutilizzabile con l'obiettivo di centralizzare l'accesso all'applicazione in punti specifici all'interno dei prodotti dell'azienda. L'obiettivo del *form* è centralizzare l'esecuzione dell'applicativo, infatti rappresenta il punto di ingresso per avviare l'applicazione permettendo di visualizzare e modificare in maniera semplice i dati legati al *database* e alle credenziali d'accesso. Il *form* è stato infatti richiesto per due motivi principali ovvero gestire in modo rapido la connessione al *database* senza dover modificare stringhe a livello codice, e supportare configurazioni lanciando l'*app* in modo diretto dal gestionale e permettendo all'utente di salvare i propri dati di accesso e riutilizzarli all'interno di sessioni successive. L'utilizzo di Windows *form* è stata richiesta in modo diretto dall'azienda, mantenendo un interfaccia semplice focalizzata sulla connessione all'*app* vera e propria. Per garantire la connessione al *database* viene modificato in modo diretto il *file appsetting.json*, che contiene tutte le stringhe di connessione. Per quanto riguarda invece la gestione delle credenziali e delle sessioni di accesso il discorso è più lungo. Questa sezione ha l'obiettivo di mantenere la connessione alla *chat* anche in caso la connessione venga interrotta a causa delle sessioni a tempo. Risulta scomodo l'inserimento delle credenziali in maniera manuale ogni trenta minuti. Per risolvere questo problema è stato aggiunto al *form* una sezione che consente l'inserimento delle credenziali iniziali, esegue la chiamata di *login* e ottiene in risposta due *token* uno di accesso e uno di *refresh*. Questi due *token* vengono salvati in *file* locali in particolare:

- **log.json**: salva il *token* di accesso alla sessione che sta per essere avviata.
- **ses.json**: salva il *token* di *refresh*, utilizzato per aggiornare la sessione corrente senza il reinserimento dei dati.
- **data.json**: registra l'orario della generazione dei *token*.

Ogni sessione ha la durata *standard* di trenta minuti e per garantire quindi continuità nell'uso della *chat*, viene letto il valore temporale di **data.json**, se sono

trascorsi più di trenta minuti il sistema esegue automaticamente una chiamata di *refresh* e sovrascrive i *token* precedenti con quelli nuovi ricevuti come risposta.

3.2.4 Difficoltà incontrate durante la codifica

Lo sviluppo delle *API_G* per VisionAI ha richiesto un'attività di codifica approfondita, caratterizzata anche dai vincoli strutturali imposti dalla base dati esistente e dall'architettura predefinita. Una delle complessità principali è stata la gestione dell'accesso a dati distribuiti su più tabelle relazionate. Praticamente tutte le *API_G* sviluppate nel periodo infatti chiedevano la risposta a domande complesse e per garantire ciò era necessario accedere a dati provenienti da entità diverse. Per affrontare queste difficoltà è stato adottato Entity Framework Core, un *ORM_G* che consente di eseguire *query SQL* in maniera astratta tramite LINQ-to-Entities, questo approccio ha permesso di gestire le relazioni tra tabelle attraverso *join* esplicativi all'interno del codice ma senza usare codice SQL, applicare filtri dinamici per permettere l'utilizzo di parametri all'interno della chiamata, recuperare e proiettare i dati in strutture che poi vengono trasformate in *DTO_G*. È possibile visualizzare il funzionamento ad alto livello di un sistema *ORM_G* nella figura 3.3. Particolare attenzione è stata dedicata anche alla validazione dei dati in *input* per quanto riguarda le *API_G POST*, effettuando controlli sui campi obbligatori, normalizzando i dati prima dell'inserimento nel *database* e aggiornando lo *script* di assegnazione automatica del campo *CodCliFor* applicando l'algoritmo di assegnazione aziendale.

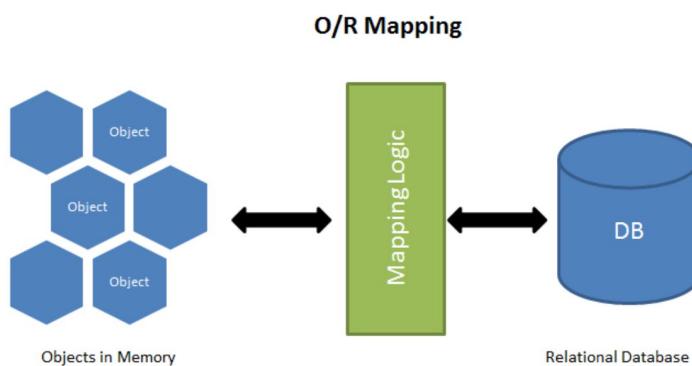


Figura 3.3: Diagramma di funzionamento di un ORM.

Fonte: <https://www.youstable.com/it/blog/cos' Ã-ORM-nella-programmazione/>

3.2.5 Verifica e presentazione dei risultati ottenuti

La verifica ha rappresentato un’attività molto importante del processo di sviluppo, al fine di garantire la correttezza e coerenza delle funzionalità implementate. Sebbene, per motivi di tempo, non siano stati introdotti *test* automatici strutturati, ho condotto un’attenta validazione manuale di tutte le API_G sviluppate, sia negli ambienti locali sia nell’integrazione con il *chatbot*.

Test funzionali delle API_G

Ogni $endpoint_G$ è stato *testato* singolarmente mediante strumenti di ispezione delle API_G tra cui Swagger, utilizzato durante lo sviluppo per l’esposizione degli *enpoint*, il controllo dei parametri, la correttezza delle risposte $HTTP_G$ e la visualizzazione della documentazione interattiva delle API_G . Altro strumento utilizzato è stato Postman, impiegato per simulare chiamate reali dinamiche tramite OData, per visualizzare le risposte in formato JSON. Per ogni API_G sviluppata è stato verificato:

- il corretto *routing*,
- la validità dei parametri e la logica a loro collegata,
- la coerenza delle risposte con i dati effettivamente presenti nel *database*,
- la corretta gestione degli errori,
- coerenza dei dati rispetto la logica richiesta.

Oltre all’integrazione di *test* locali tramite gli strumenti citati, sono stati svolti *test* sull’integrazione effettiva delle API_G con l’assistente utilizzando Ngrok per esporre il *backend* locale rendendolo accessibile al motore esterno di DevLab. Al termine delle attività sono state sviluppate con successo una serie di nuove API_G che integrano funzionalità e permettono all’assistente virtuale di rispondere in modo corretto a domande collegate a nuovi contesi. Nella tabella 3.2 presento tutte le API_G sviluppate e sistemate.

Nome API	Metodo	Tipologia
Dipendenti (Anagrafiche Dipendenti)	<i>GET / POST</i>	Nuova
Banche (Saldi Conti Correnti)	<i>GET</i>	Nuova
Ordini Evasi / Non Evasi	<i>GET</i>	Nuova
Manutenzioni da Eseguire	<i>GET</i>	Nuova
Manutenzioni per Destinazione	<i>GET</i>	Nuova
SqlTraducer	<i>GET</i>	Aggiornata
CliFor (Clienti/Fornitori)	<i>GET / POST</i>	Aggiornata
Contatti (Anagrafiche Contatti)	<i>GET / POST</i>	Aggiornata
Attività	<i>GET / POST</i>	Aggiornata

Tabella 3.2: API sviluppate e sistemate durante il periodo di stage.

Windows *from*

Anche per il Windows *form* non sono stati effettuati dei *test* automatici ma abbiamo controllato le funzionalità verificando l'avvio dell'applicazione, la corretta modifica delle stringhe di connessione al *database*, il salvataggio di una nuova connessione ad un nuovo *database* e funzionalità collegate alla semplice interfaccia sviluppata che è possibile visualizzare nell'immagine [3.4](#).

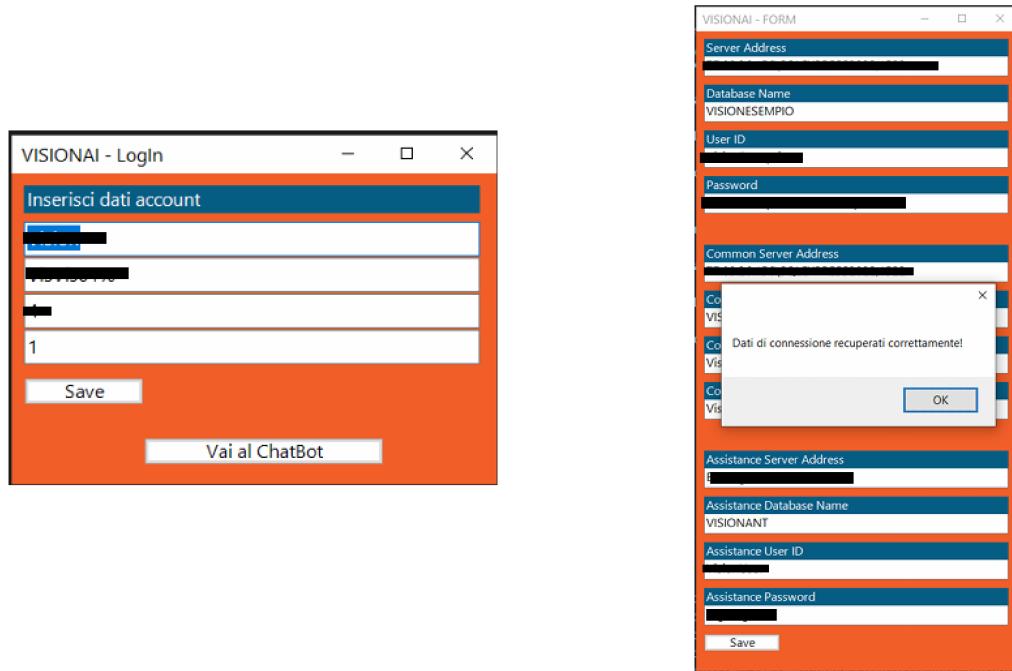


Figura 3.4: Interfaccia finale *form*.

3.3 Sistema di assegnazione automatica per VisionAssistance

Lo sviluppo del modulo di assegnazione automatica ha rappresentato una delle sfide più articolate e stimolanti dell’esperienza di *stage*. L’obiettivo principale era quello di progettare un sistema in grado di rendere automatica l’assegnazione degli interventi ottimizzando una serie di variabili temporari, logistiche e contrattuali. Data la complessità del dominio e l’alto numero di vincoli da rispettare si è deciso di affrontare il problema mediante la realizzazione di un *PoC_G*, prodotto incompleto per dimostrare la fattibilità, tipo un prototipo, producendo un prodotto funzionante ma limitato.

3.3.1 Analisi dei requisiti

L’attività di definizione e raccolta dei requisiti si è basata su una stretta collaborazione con il *tutor* aziendale e con il *team* interno di VisioneImpresa. I requisiti sono stati definiti partendo da una serie di specifiche fornite in forma di documenti e integrate con vari incontri informali e sessioni di chiarimento con

personale specializzato nel contesto di VisionAssistance. Alla base dei requisiti si pone l'obiettivo creare un sistema di l'assegnazione automatica dei tecnici agli interventi da svolgere, effettuando una pianificazione compatibile con diverse variabili come le disponibilità temporali dei tecnici e dei clienti, i vincoli contrattuali, le distanze fisiche, i tempi di spostamento e la priorità .

Nella tabella 3.3 sono contenuti tutti i requisiti individuati, il codice viene definito nel seguente modo:

R[O/D/F]-NN

dove:

- **R** indica che si tratta di un requisito;
- **O/D/F** specifica il grado di priorità:
 - **O**: Obbligatorio – deve essere implementato;
 - **D**: Desiderabile – migliora il sistema ma non è essenziale;
 - **F**: Facoltativo – implementabile solo in caso di tempo o risorse aggiuntive.
- **NN** è un numero progressivo per identificare univocamente ogni requisito.

Codice	Descrizione
RO-01	Rispettare orari di apertura e giorni lavorativi specificati dal cliente tramite la tabella <code>P_Orari</code> .
RO-02	Considerare gli orari lavorativi settimanali dei tecnici contenuti nella tabella <code>CalendSett</code> .
RO-03	Rispettare festività e orari speciali contenuti nella tabella <code>CalendDay</code> .
RF-04	Utilizzare <i>Google Places API</i> per verificare apertura delle attività.
RO-05	Aggiungere tempo amministrativo fisso o in percentuale configurato per ogni intervento.
RO-06	Pianificare interventi solo in presenza di sovrapposizione oraria tra cliente e tecnico.

RO-07	Considerare partenza del tecnico dalla sede aziendale di appartenenza.
RO-08	Mantenere orario se la data prevista è futura, ricalcolare se è già trascorsa.
RO-09	Rispettare l'assegnazione del tecnico da contratto se presente.
RD-10	Se il tecnico del contratto differisce da quello del buono, sostituirlo con quello corretto.
RD-11	Consentire la sostituzione del tecnico solo se non vincolato da contratto.
RO-13	Supportare assegnazioni spezzate per interventi di lunga durata.
RF-14	Prevedere <i>fallback</i> in caso di assenza di tecnici disponibili.
RD-15	Generare <i>output</i> in formato JSON.

Tabella 3.3: Tabella dei requisiti individuati per il modulo di assegnazione interventi.

3.3.2 progettazione

Come anticipato nella sezione 3.1 per motivi legati ai tempi del progetto, abbiamo deciso di dividere il lavoro, io mi sono concentrato principalmente sulla progettazione e implementazione della *pipeline* dei dati. Questa parte del progetto costituisce la base di informazioni su cui opera l'algoritmo di assegnazione automatica sviluppato in parallelo dal mio collega. La progettazione ha seguito principi di modularità e riusabilità e disaccoppiamento delle responsabilità, come è visibile anche solo a livello di struttura della *repository* nella figura 3.5 fissando come obiettivi principali:

- Modularità del sistema, ogni componente svolge un compito preciso e può essere modificato indipendentemente.
- Automazione del flusso dei dati, la *pipeline* è progettata per essere avviata con un unico comando.
- Standardizzazione, l'*output* di questo modulo è standardizzato per essere compatibile con il sistema di assegnazione automatica.
- Estensibilità, la struttura modulare permette l'aggiunta di nuovi componenti in maniera semplice senza modificare la logica esistente.

```
Pipeline/
└── Scripts/
    ├── main.py          # Punto di ingresso per eseguire la pipeline.
    ├── interventi.py    # Script che crea 'interventi_finale.json'
    ├── tecnicci.py      # Script che crea 'tecnicci_finale.json'
    ├── Orari.py         # Script che converte orari (12.00) in minuti
    └── distance_matrix.py # Calcola la matrice delle distanze

    └── data_json/
        ├── Finale/       # Cartella dove vengono salvati i file JSON
        │   └── finali
        └── file/         # File dati raw

    └── DB/
        ├── connection.py # Gestisce la connessione al database
        └── queries.py     # Contiene le query per raccogliere i dati

    └── utils/
        └── json_builder.py # Contiene la logica per salvare i dati in
                            # formato JSON
```

Figura 3.5: Struttura delle *repository* della *pipeline*.

Dal punto di vista architetturale la *pipeline* è organizzata in moduli indipendenti come si può visualizzare nella figura 3.6, sviluppati in Python e raggruppati in diverse cartelle come **Scripts**, **DB** e **data_json**, ogni modulo svolge un passaggio chiave:

- **Connection:** stabilisce la connessione con il *database* SQL Server utilizzando SQLAlchemy e Pyodbc, riunendo tutta la gestione delle variabili di accesso.
- **Queries:** contiene tutte le *query* SQL utilizzate per estrarre tutti i dati necessari al fine di creare una base dati robusta per l'algoritmo di assegnazione.
- **Interventi:** modulo che si occupa di normalizzare tutti i dati relativi agli interventi da svolgere, gestendo vincoli, priorità, orari dei clienti e posizioni, andando a produrre il *file* *interventi_finale.json*.
- **Tecnicci:** modulo che in maniera simile a quello degli interventi va a organizzare e normalizzare tutti i dati relativi ai tecnici andando a creare il *file* *tecnicci_finale.json*.

- **Orari:** modulo che si occupa di convertire gli orari in minuti, per semplificare la manipolazione temporale nella sezione di *scheduling*.
- **Distance_matrix:** Modulo che si occupa di calcolare le distanze (in minuti) tra sedi e interventi utilizzando le *API G Google Maps Distance Matrix*, producendo il file `distance_matrix.json`.

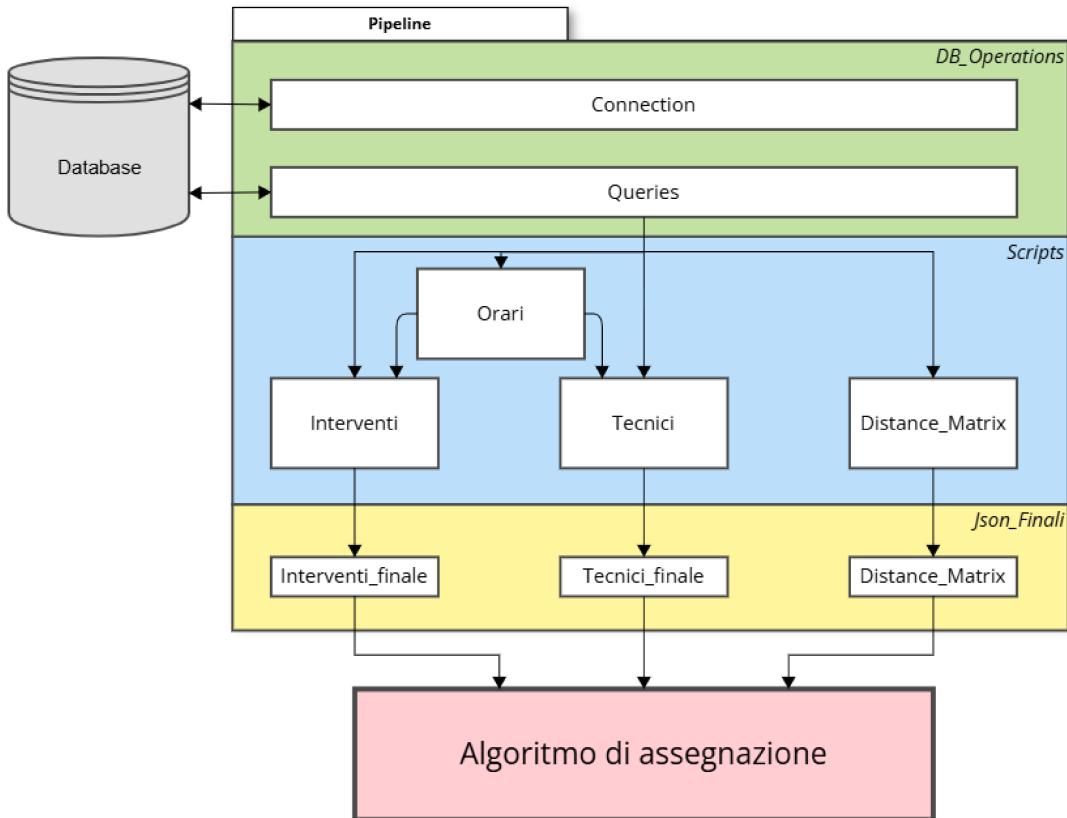


Figura 3.6: Struttura architetturale della *pipeline* dei dati.

Nella tabella 3.4 vengono presentate le scelte progettuali ritenute chiave.

Decisione	Motivazione
Linguaggio Python	Facilita l'integrazione, lo sviluppo rapido e l'uso di librerie avanzate per l'elaborazione dati.
SQLAlchemy + ODBC	Permette <i>query</i> flessibili e sicure su SQL Server con gestione centralizzata delle connessioni.
Script modulari	Ogni parte della <i>pipeline</i> è isolata e facilmente modificabile/ <i>testabile</i> .
Gestione orari in minuti	I dati temporali sono convertiti in formato numerico per ottimizzare la gestione dei vincoli nell'algoritmo.
<i>Google Distance Matrix API</i>	Calcolo realistico dei tempi di viaggio.

Tabella 3.4: Scelte progettuali chiave adottate nella realizzazione della *pipeline*.

3.3.3 Verifica e risalutati ottenuti

La verifica per il modulo di *pipeline* dei dati ha rappresentato un'attività fondamentale per garantire la correttezza delle informazioni generate, che venendo poste alla base dell'algoritmo di decisione dei tecnici, è di estrema importanza quindi la verifica della corretta presentazione. Per questa sezione essendo anche un *PoC* non abbiamo introdotto sistemi di *testing* automatici o semiautomatici incentrando la fase di verifica sulla validazione funzionale dei singoli script e assicurandosi che l'*output* fosse coerente con i dati nel *database* e integrabile con il modulo di *scheduling*.

Sono stati fatti quindi *test* manuali sistematici, che si concentrano sulla coerenza e la qualità dei dati prodotti.

- Controllo diretto delle *query* SQL: ogni *query* eseguita nel processo di estrazione dei dati è stata confrontata con i dati reali presenti nel *database*, per verificarne la corrispondenza e la coerenza.
- Verifica dei dati aggregati: i *file* JSON generati sono stati analizzati manualmente per accertare che ogni informazione fosse coerente rispetto all'origine nel *database* e non ci fossero dati nulli non giustificati.

- Validazione del processo di trasformazione dei dati: sono state verificate le strutture dati finali per assicurare che riflettessero in maniera corretta quanto previsto in fase di progettazione
- Matrice delle distanze: Una attenzione maggiore è stata dedicata alla verifica della matrice dei tempi di percorrenza tra le sedi e i luoghi di intervento. Sono stati effettuati diversi *test* per verificare la correttezza degli indirizzi presi in considerazione per il calcolo delle distanze, avvenuto un controllo a campione della correttezza dei tempi di viaggio restituiti, assicurare la funzionalità del sistema *testando* la divisione in chiamate per rispettare i limiti imposti da Google e la gestione delle risposte **null**.

La *pipeline* dei dati sviluppata è stata integrata nel *PoC_G*, questa unione ha confermato la corretta realizzazione di essa, in quanto tutti i *file JSON* prodotti sono risultati compatibili con l'algoritmo di pianificazione, l'elaborazione dei dati ha portato a risultati concreti in rispetto dei vincoli imposti e l'*output* dell'algoritmo ha evidenziato una risposta funzionale coerente pur essendo in uno scenario di prova. L'intero lavoro si è tradotto quindi nella realizzazione di un prototipo funzionante concreto, che dimostra la fattibilità e la possibile integrazione futura nei prodotti aziendali. Per quanto riguarda i requisiti inizialmente individuati, i requisiti obbligatori sono stati tutti integrati assicurando che il sistema di assegnazione dei tecnici rispettasse i vincoli imposti mentre i requisiti desiderabili e facoltativi non sono stati integrati nel sistema.

Capitolo 4

Valutazione retrospettiva

4.1 Valutazione degli obiettivi

4.1.1 Obiettivi aziendali

Gli obiettivi aziendali rivolti al progetto di *stage* erano diversi e distribuiti sui due moduli differenti che ne facevano parte. Come indicato all'interno della tabella 2.2 e 2.3, gli obiettivi per questi progetti erano divisi in obbligatori, desiderabili e facoltativi.

Nel caso del modulo legato all'estensione del raggio di competenza della *chatbot* aziendale chiamata VisionAI, ho raggiunto pienamente i due obiettivi obbligatori:

- O001: sviluppo di interfacce con DevExpress - l'interfaccia Windows *form* è stata implementata con successo, contenendo tutte le funzionalità richieste come connessione diretta alla *chat* salvando i dati e le sessioni e modificare e aggiungere connessioni ai *database*
- O002: sviluppo *API_G REST_G* in C# - ho contribuito attivamente allo sviluppo delle nuove *API_G* ma anche alla correzione di quelle esistenti, al fine di estendere la funzionalità dell'assistente VisionAI. Tutte le *API_G* le ho sviluppate seguendo l'architettura definita dall'azienda utilizzando ASP.NET Core.

Ho raggiunto anche l’obiettivo D01 che comprendeva la documentazione e il piano dei *test*, seppur in parte, vista l’assenza di una *suite* di *test* automatici, compensato però da una rigida verifica manuale delle funzionalità e una documentazione dettagliata. Per quanto riguarda il modulo legato al contesto di VisionAssistance, l’obiettivo indicato dall’azienda era facoltativo, F01 ovvero la creazione di un *agent* per la pianificazione automatica degli interventi, questo obiettivo è stato raggiunto sotto forma di *PoC_G* che ha dimostrato la fattibilità del progetto, rispettando i vincoli imposti e dando risposte coerenti.

In generale ho realizzato tutti gli obiettivi aziendali, ottenendo anche un buono stato di soddisfacimento da parte del referente aziendale al momento della presentazione del lavoro svolto.

4.1.2 Obiettivi personali

Prima che iniziasse lo *stage*, mi ero posto alcuni obiettivi personali con l’intenzione di sfruttare l’esperienza per apprendere molto sia dal un punto di vista formativo che umano. Gli obiettivi che mi sono posti non erano riguardanti solo l’acquisizione di competenze tecniche, ma anche il modo in cui volevo affrontare il lavoro in un contesto aziendale.

Uno degli aspetti riguardava la volontà di apprendere nuove tecnologie, in particolare legate all’ambiente di .NET e allo sviluppo in linguaggio C#. Provenendo da un ambiente accademico incentrato molto sulla teoria mi interessava ampliare il mio *stack* tecnologico con soluzioni legate prettamente al mondo del lavoro. Lavorare su un progetto reale utilizzando queste tecnologie mi ha permesso di crescere molto. Un altro obiettivo importante per me era quello di sviluppare un metodo di lavoro, imparando a pianificare le attività rispettare le scadenze e avere un flusso di lavoro continuo e positivo. Volevo seguire una struttura ben definita organizzandomi le cose da fare e arrivando a terminare il lavoro senza ritardi e difficoltà e il contesto aziendale si è rivelato molto utile per raggiungere questo obiettivo. Avevo inoltre la volontà di rafforzare le mie capacità relazionali, sia all’interno del *team* di sviluppo sia nel confronto con altri membri dell’azienda. La possibilità di collaborare al progetto è stata molto

importante, permettendomi di lavorare sempre insieme a qualcuno migliorando molto le mie capacità comunicative e rendendo il progetto più interessante. Infine uno degli aspetti che ritenevo più importanti era quello di mettermi alla prova in un ambiente lavorativo, allontanandomi dalla zona di *comfort* dell'università. Ho cercato di affrontare con positività nuove situazioni e a valorizzare questa esperienza come possibilità di crescita. Per concludere posso confermare che gli obiettivi personali sono stati raggiunti in modo pieno, ho trovato questa esperienza molto formativa sotto sia il punto di vista tecnico che umano.

4.2 Conoscenze e competenze acquisite

Durante il periodo di *stage* ho avuto la possibilità di imparare moltissimo accrescendo il mio bagaglio tecnico e di competenze trasversali, l'esperienza è stata anche più ricca di quanto mi aspettassi poiché mi ha permesso di lavorare su progetti complessi e concreti e con tecnologie utilizzate nel mondo del lavoro. Uno degli aspetti più importanti se parliamo di competenze acquisite è sicuramente l'ambiente .NET e il linguaggio C#, che ho utilizzato in maniera quotidiana per lo sviluppo del *backend* di VisionAI. Pur avendo qualche nozione di base durante lo *stage* ho appreso l'organizzazione di un vero progetto aziendale con un architettura a strati e l'importanza di alcuni concetti base per la scrittura di un buon codice. In particolare ho appreso familiarità con lo sviluppo di *API* tramite ASP.NET Core, la progettazione di un sistema *controller*, *repository* e *service* e l'utilizzo di tecnologie e *framework* correlate all'ambiente .NET. Ho imparato molto anche dal punto di vista del lavoro con *database*, lo *stage* ha rappresentato proprio un salto di livello. Pur partendo con una base di nozioni solida sia per quanto riguarda SQL sia per quanto riguarda il *database*, confrontarmi con una base di dati complessa e un sistema di grandi dimensioni basato sulle relazioni tra tabelle mi ha portato a sviluppare competenze più avanzate. Sicuramente ho imparato a scrivere *query* complesse, lavorare con un *database* di grandi dimensioni comprendendone la struttura e utilizzare SQL Server Management Studio come strumento quotidiano. Oltre a questi ho potuto utilizzare molti altri strumenti come Google Or-Tools e ap-

prndere nozioni sulla programmazione vincolata, utilizzare *ORM_G* come Entity framework e SQLAlchemy conoscendo molte nuove tecnologie e soluzioni. Oltre alle competenze tecniche lo *stage* mi ha permesso di crescere anche con abilità trasversali fondamentali per il mondo del lavoro, come il lavorare in *team* in maniera continuativa, confrontandomi ogni giorno con un'altra persona e condividendo idee e problemi. Migliorare sotto il punto di vista comunicativo e di organizzazione del lavoro, relazionarmi con un ambiente lavorativo e tutte le dinamiche interne. Tutte queste competenze e conoscenze che ho appreso in questo periodo le reputo molto preziose.

4.3 Considerazioni finali sul percorso universitario

Il percorso universitario che ho affrontato è stato sicuramente impegnativo e con molti ostacoli, allo stesso tempo però molto stimolante e costruttivo. Questi anni all'interno dell'ambiente universitario sono stati molto formativi non solo dal punto di vista di apprendimento ma anche dal punto di vista metodologico e umano, e tornassi indietro sceglierrei ancora questo percorso consapevole del valore che mi ha restituito. Durante questo periodo, lavorando all'interno di un'azienda e partecipando in maniera attiva a progetti veri mi sono reso conto che l'università ti prepara al mondo del lavoro, anche se in un modo meno diretto di quello che si può pensare. Non ti insegna ad usare ogni strumento o a conoscere ogni linguaggio, sarebbe impossibile, soprattutto nel mondo dell'informatica che è in continua evoluzione. Quello che invece ti offre è una base molto solida di conoscenze teoriche e un metodo di ragionamento. Ti insegna come imparare, come risolvere problemi e a risolverli da solo. Con questa esperienza ho capito in modo significativo di quanto le basi siano essenziali, perché senza fondamenta non si costruisce niente di solido. In informatica imparare come affrontare un problema, anche senza conoscere a priori la tecnologia, è più importante che conoscere la tecnologia stessa e questo è alla base di questa università.

Glossario

API (Application Program Interface) In un programma informatico per *API* si intende un insieme di definizioni o protocolli che permettono a componenti *software* di comunicare con altri componenti *software* e scambiarsi dati. Le *API* permettono la condivisione solo dei dati necessari mantenendo nascosti i dettagli interni del sistema.

ERP Un *enterprise resource planning* è un tipo di *software* di gestione che integra tutti i processi aziendali come vendite, acquisti, gestione magazzino, finanza e contabilità. I sistemi *ERP* centralizzano i dati provenienti da un insieme di processi di *business* rendendoli disponibili a tutti. Oggi i sistemi *ERP* risultano fondamentali per la gestione delle aziende.

PBX Un *private branch exchange* nelle telecomunicazioni è una rete telefonica privata che viene usata all'interno di un'azienda o organizzazione. Questa infrastruttura permette la comunicazione interna ed esterna. Un *PBX* consente di avere più telefoni rispetto alle linee telefoniche fisiche e consente chiamate gratuite tra gli utenti.

IDE Con *integrated development environment* o ambiente di sviluppo integrato si intende un *software* progettato per la realizzazione di applicazioni che riunisce diversi strumenti di sviluppo in un'unica interfaccia grafica. Costituito da un *editor* di codice sorgente, un'automazione della *build* locale e un *debugger*.

REST *Representational state transfer* si tratta di un'architettura *software* che impone condizioni sul funzionamento di un *API*. Si utilizza l'architettura basata su *REST* per supportare comunicazioni affidabili e con alte

prestazioni su larga scala. Gli sviluppatori di *API* possono progettarle utilizzando diverse architetture e quelle che seguono le architetture *REST* sono chiamate *RESTful*.

ORM *Object-relational mapping* nell'informatica è un *pattern* di programmazione che crea un ponte virtuale tra il paradigma orientato agli oggetti e il modello relazionale dei *databases*, permettendo agli sviluppatori di manipolare dati ed effettuare chiamate utilizzando gli oggetti nel linguaggio di programmazione corrente senza inserire codice in *SQL*.

DTO *data transfer object* è un oggetto che definisce le modalità di invio dati in rete o tra sottoinsiemi di un'applicazione software, rendendo possibile mappare i dati che si vogliono trasferire nascondendo quelli non opportuni o sensibili.

HTTP *Hypertext Transfer Protocol* è un protocollo di comunicazione alla base della trasmissione di dati sul *web*, e definisce i comandi e i servizi utilizzati per il trasferimento dei dati in rete.

LLM *large language model*, in italiano modello linguistico di grandi dimensioni, è una tecnologia molto avanzata nell'ambito *AI*, incentrata sulla comprensione, l'analisi e la generazione del testo in ambito generale.

PoC *proof of concept* ha lo scopo di determinare la fattibilità dell'idea e verificare che l'idea funzionerà come previsto. Può essere visto come una fase di *test* iniziale dell'idea e come un prototipo.

Legacy Un sistema *legacy* in informatica è un sistema utilizzato per un periodo prolungato che generalmente presenta una tecnologia obsoleta, prestazioni inefficienti, vulnerabilità di sicurezza, costi di manutenzione elevati, scalabilità limitata e scarsa attendibilità.

Endpoint un *endpoint* è un luogo digitale in cui un *API* riceve chiamate, note come richieste, per le risorse sul suo *server*. Gli *endpoint* sono complementari alle *API* e si presentano molto spesso sotto forma di *URL*.

Clean architecture Il termine *Clean architecture* deriva dall'omonimo libro di

Robert C. Martin e possiamo pensarla come un insieme di linee guida per progettare l'architettura di un *software*. I suoi principi si possono riassumere in indipendenza dal *framework*, indipendenza dall'*UI*, indipendenza dal *database* e *testabile*.

Sitografia

Definizione API, ibm. URL: <https://www.ibm.com/it-it/think/topics/api>.

Definizione Clean architecture, techisfun.github. URL: <https://github.com/techisfun/clean-architecture-introduction.html>.

Definizione DTO, microsoft. URL: <https://learn.microsoft.com/it-it/aspnet/web-api/overview/data/using-web-api-with-entity-framework/part-5>.

Definizione endpoint, ibm. URL: <https://www.ibm.com/it-it/think/topics/api-endpoint>.

Definizione ERP, oracle. URL: <https://www.oracle.com/it/erp/what-is-erp/t/>.

Definizione HTTP, geekandjob. URL: <https://www.geekandjob.com/wiki/http>.

Definizione IDE, redhat. URL: <https://www.redhat.com/it/topics/middleware/what-is-ide>.

Definizione legacy, ibm. URL: <https://www.ibm.com/it-it/think/topics/legacy-application-modernization>.

Definizione LLM, hpe. URL: <https://www.hpe.com/it/it/what-is/large-language-model.html>.

Definizione ORM, codegrind. URL: <https://www.codegrind.it/blog/cosa-sono-orm>.

APPENDICE . SITOGRAFIA

Definizione PBX, 3CX. URL: <https://www.3cx.it/voip-sip/sistema-telefonico-pbx/>.

Definizione PoC, microsoft. URL: <https://learn.microsoft.com/it-it/aspnet/web-api/overview/data/using-web-api-with-entity-framework/part-5>.

Definizione REST, aws.amazon. URL: https://docs.aws.amazon.com/it_it/appsync/latest/devguide/what-is-rest.html.

Schwaber, Ken e Jeff Sutherland. *La Guida Definitiva a Scrum: Le Regole del Gioco.* URL: <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Italian.pdf> (cit. a p. 7).

Società Benefit - VisioneImpresa. URL: <https://www.vsh.it/azienda/societa-benefit/> (cit. a p. 1).