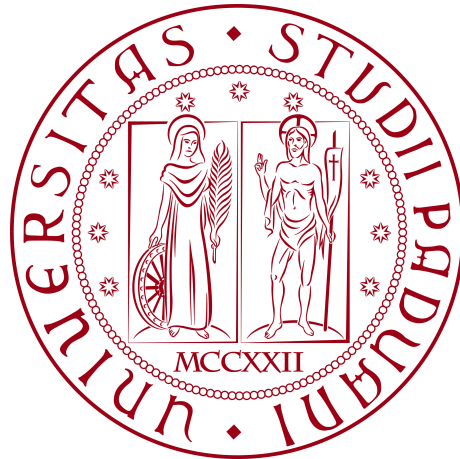


Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA “TULLIO LEVI-CIVITA”

CORSO DI LAUREA IN INFORMATICA



**Estrazione automatica di dati assicurativi:
confronto tra tecniche di OCR tradizionali e
basate sull'Intelligenza Artificiale.**

Tesi di Laurea Triennale

Relatore

Prof. Bresolin Davide

Laureanda

Caterina Vallotto

Matricola 2076434

ANNO ACCADEMICO 2024-2025

*“I knew exactly what to do. But in a much more real sense, I had no idea
what to do.”*

— Michael Scott, *The Office*, Season 5: Stress Relief

Ringraziamenti

Desidero esprimere la mia gratitudine al professor Bresolin Davide, mio relatore, per la disponibilità, la competenza e il prezioso supporto con cui mi ha accompagnato durante il percorso di stesura di questa tesi.

Vorrei anche ringraziare la mia famiglia per il loro sostegno (non solo economico), il grande aiuto e la loro presenza in ogni momento durante gli anni di studio, anche quando la sessione mi trasformava in un essere asociale e puzzolente. Li ringrazio soprattutto per la grande pazienza, che posso assicurare è stata davvero tanta. Ma proprio tanta.

Un ringraziamento speciale va a tutte le persone a me care, gli amici di una vita e quelli che ho incontrato lungo il cammino, che in modi diversi mi hanno incoraggiata e spinta a dare il meglio di me.

Al gruppo Cluedo, lascio il mio set di Lego Creator 3in1 Adorabili cagnolini e le mie Vans a quadri bucate, spero sia abbastanza per ripagarvi della quotidiana leggerezza che mi avete regalato. In ordine alfabetico (e non di simpatia): Agnese, Anna, Marco, Maria, Miriam, Valentina, Viola. Siete per me come una seconda famiglia quindi per favore, continuate a essere miei amici.

Grazie a chi, fin dai banchi delle superiori, mi è stato sempre accanto. Melissa, Melissa ed Elena, siamo cresciute insieme e abbiamo condiviso sia i momenti belli che quelli brutti dell'adolescenza. La nostra amicizia è stata una costante

in questi anni, un punto di riferimento nei momenti difficili pronto a offrirmi supporto emotivo, più o meno volontario.

Valentina, grazie per essere stata al mio fianco, per avermi supportata in ogni scelta importante, per avermi ascoltata e incoraggiata a dare sempre il massimo. La tua presenza mi ha donato sicurezza e mi ha aiutata a superare gli ostacoli che ho incontrato lungo il mio percorso.

Ultimi, ma non per importanza, mille grazie ai miei compagni di sventura dell'università, per l'aiuto reciproco che ci siamo dimostrati, per le infinite ore passate a studiare insieme e soprattutto per il sostegno morale pre-esame. Con le vostre battute (per non dire s*****e), avete reso questi tre anni più sopportabili e spensierati. Resterete sempre i miei fratelli.

Padova, Luglio 2025

Caterina Vallotto

Sommario

Questo documento descrive il lavoro che ho svolto durante il periodo di stage presso l'azienda Wavelop SRL, incentrato sull'analisi, progettazione e sviluppo di una web application in ambito assicurativo. L'applicativo è destinato agli agenti assicurativi e consente il caricamento e l'interpretazione automatica dello schedario viticolo di un cliente, con l'obiettivo di estrarre in modo efficiente tutte le informazioni necessarie alla formulazione di preventivi. Le fasi iniziali hanno previsto l'analisi e la progettazione dell'intero applicativo, attraverso la creazione di wireframes e user stories che rappresentano le fondamenta dell'intero progetto. Inoltre, lo sviluppo ha previsto una fase di ricerca, volta a individuare la soluzione più efficace per l'analisi dei documenti, considerando sia tecniche OCR tradizionali che approcci basati sull'Intelligenza Artificiale. Successivamente, ho implementato una parte della soluzione individuata, ponendo particolare attenzione al salvataggio strutturato dei dati estratti e allo sviluppo di un'interfaccia utente in grado di agevolare l'interazione dell'agente con il sistema.

Indice

Acronimi e abbreviazioni	xi
1 Introduzione	1
1.1 L'azienda	1
1.2 L'idea	2
1.3 Organizzazione del testo	3
2 Processi e metodologie	4
2.1 Modello di sviluppo Agile	4
2.1.1 Evoluzione dei modelli di sviluppo: nascita del modello Agile	4
2.1.2 Manifesto Agile	5
2.1.3 Caratteristiche distintive del metodo Agile	6
2.2 Scrum	7
2.2.1 Concetti essenziali del framework Scrum	7
2.2.2 Ruoli	8
2.2.3 Sprint e cerimonie	8
2.3 Benefici e sfide dell'approccio Agile	10
2.4 Applicazione di Agile Scrum in Wavelop SRL	11
2.5 Gestione del versionamento del codice	14
2.6 Metodi e processi applicati allo stage	15
3 Descrizione dello stage	16
3.1 Introduzione al progetto Insurance Estimator	16
3.2 Pianificazione del lavoro	17

3.3	Risorse, tecnologie e strumenti utilizzati	18
4	Analisi	23
4.1	Elementi iniziali	23
4.2	Struttura applicativo e wireframes	23
4.3	Epiche, user stories e story points	26
5	Tecniche OCR e di Intelligenza Artificiale per l'estrazione di dati	29
5.1	Introduzione	29
5.2	Tecniche OCR tradizionali	30
5.2.1	Definizione e contesto	30
5.2.2	Evoluzione storica	30
5.2.3	Architettura e funzionamento	31
5.2.3.1	Acquisizione del documento	32
5.2.3.2	Pre-elaborazione dell'immagine (Image Preprocessing)	32
5.2.3.3	Segmentazione (Layout Analysis)	35
5.2.3.4	Estrazione delle caratteristiche (Feature Extraction)	36
5.2.3.5	Post-elaborazione (Postprocessing)	37
5.2.4	Punti di forza e criticità	38
5.2.4.1	Punti di forza	39
5.2.4.2	Criticità e limitazioni	39
5.3	Tecniche di Intelligenza Artificiale per l'estrazione di dati	41
5.3.1	Definizione e contesto	41
5.3.2	Evoluzione storica	42
5.3.3	Architettura e funzionamento	44
5.3.3.1	Acquisizione e pre-processing del documento	45
5.3.3.2	Parsing visivo e analisi del layout	45
5.3.3.3	Tokenizzazione ed embedding	46
5.3.3.3.1	Tokenizzazione	46
5.3.3.3.2	Embedding statici	46

5.3.3.3.3	Embedding contestuali	47
5.3.3.3.4	Embedding multimodali	48
5.3.3.4	Named Entity Recognition (NER)	49
5.3.3.5	Relationship extraction e modellazione semantica	49
5.3.3.6	Output strutturato	50
5.3.4	Punti di forza e criticità	51
5.3.4.1	Punti di forza	51
5.3.4.2	Criticità e limitazioni	52
5.4	Confronto tra OCR tradizionale e AI	54
5.4.1	Casi d'uso e applicazioni pratiche	55
5.5	Scelta adottata	56
6	Progettazione	57
6.1	Progettazione database	57
6.2	Progettazione frontend	60
6.3	Progettazione backend	62
6.4	Progettazione estrazione dati	63
7	Sviluppo	68
7.1	Sviluppo del frontend	68
7.2	Sviluppo del backend	69
7.3	Sviluppo del flusso di estrazione dati	70
7.4	Condivisione delle interfacce TypeScript	73
8	Conclusioni	75
8.1	Consuntivo finale	75
8.2	Valutazione personale	76
	Glossario	i
	Bibliografia	iv
	Sitografia	vi

Elenco delle figure

1.1	Logo dell'azienda Wavelop SRL.	1
2.1	Cerimonie Sprint - Atlassian.	9
2.2	Gitflow Workflow - Atlassian.	14
4.1	Struttura ad alto livello dell'applicativo.	24
4.2	Wireframe lista clienti.	25
4.3	Wireframe schedario.	25
4.4	Wireframe preventivo.	26
5.1	OCR pipeline.	32
5.2	AI pipeline.	45
6.1	Diagramma E/R.	58
6.2	Collections MongoDB.	60
6.3	Architettura frontend.	61
6.4	Schema delle relazioni tra le pagine del frontend.	62
6.5	Schema generale del flusso di estrazione dei dati.	64
6.6	Schema del flusso di estrazione con confronto dei dati cliente. . .	66

Acronimi e abbreviazioni

AVEPA Agenzia Veneta per i Pagamenti. [16](#)

AWS Amazon Web Services. [2](#)

OCR Optical Character Recognition. [2](#)

Capitolo 1

Introduzione

Questo primo capitolo ha lo scopo di presentare l'azienda presso cui ho svolto lo stage, Wavelop SRL, e di introdurre all'idea generale del progetto svolto. Nei successivi capitoli andrò poi ad approfondire questi aspetti.

1.1 L'azienda



Figura 1.1: Logo dell'azienda Wavelop SRL.

Wavelop SRL è una software house con sede operativa a Treviso, specializzata nella progettazione e sviluppo di soluzioni digitali su misura per le aziende di ogni settore. Ha un approccio che si concentra sull'utente e mette al centro la collaborazione con i clienti per soddisfare le loro esigenze e quelle del mercato. Offre una molteplicità di servizi, come:

- Frontend development: utilizza tecnologie come React, Vue e Angular per offrire un'interfaccia dinamica e accattivante;
- Backend development: offre soluzioni che si adattano a qualsiasi tipo di database, sia SQL che NoSQL, appoggiandosi a tecnologie come Node.js, Java e PHP;
- *Amazon Web Services (AWS)*_G development: offre soluzioni Cloud, garantendo sicurezza e protezione dei dati;
- Mobile app: sviluppa app native per iOS e Android, utilizzando React Native e Flutter;
- Chat bot: crea chatbot su misura, utilizzando l'Intelligenza Artificiale per ottimizzare l'efficienza aziendale e l'esperienza dell'utente.

1.2 L'idea

Lo scopo dello stage prevede l'analisi, la progettazione e l'implementazione di un'applicazione web in ambito assicurativo. L'utente, che è un agente assicurativo, può caricare uno schedario viticolo di un cliente e l'applicativo si deve occupare di interpretare il file caricato ed estrarre le informazioni necessarie da esso. Da queste poi deve essere in grado di formulare dei preventivi utilizzando i tassi assicurativi offerti dalle varie compagnie. L'idea è quindi nata dalla necessità di automatizzare il processo di calcolo dei preventivi, attualmente svolto "a mano" dagli agenti assicurativi. Oltre all'attività di analisi dei requisiti, il focus principale del mio stage è stato quello di individuare la soluzione migliore per l'estrazione dei dati, tra soluzione *Optical Character Recognition (OCR)*_G tradizionale o mediante tecniche di Intelligenza Artificiale.

1.3 Organizzazione del testo

Il secondo capitolo descrive i processi e le metodologie applicate da Wavelop SRL.

Il terzo capitolo introduce lo scopo dello stage, la sua pianificazione e le risorse utilizzate.

Il quarto capitolo illustra l'analisi svolta sul dominio applicativo.

Il quinto capitolo presenta un approfondimento teorico delle tecnologie OCR e di Intelligenza Artificiale per l'estrazione di dati.

Il sesto capitolo approfondisce la progettazione dell'applicativo.

Il settimo capitolo descrive lo sviluppo e l'implementazione di quanto progettato.

L'ottavo capitolo trae le conclusioni dello stage, evidenziando gli obiettivi raggiunti.

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

- Gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
- Per la prima occorrenza dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: *Hook_G*.

Capitolo 2

Processi e metodologie

Questo capitolo è dedicato alla descrizione dei processi e delle metodologie di lavoro applicate dall'azienda Wavelop SRL. In particolare tratterò il modello di sviluppo Agile e il framework Scrum, e come essi vengono applicati all'interno dei processi aziendali.

2.1 Modello di sviluppo Agile

La metodologia Agile¹ è un approccio allo sviluppo del software che enfatizza la flessibilità, la collaborazione e la consegna continua di valore. Nata come risposta ai limiti dei modelli tradizionali, ritenuti troppo rigidi, burocratici e inadatti a gestire la complessità e il dinamismo dei progetti software moderni, Agile promuove cicli di sviluppo brevi e iterativi, consentendo ai team di adattarsi rapidamente ai cambiamenti e di coinvolgere attivamente i clienti durante tutto il processo.

2.1.1 Evoluzione dei modelli di sviluppo: nascita del modello Agile

La necessità di introdurre un modello di sviluppo del software trova le sue radici dal “non-modello” Code-and-Fix, che si basava su uno stile caotico, senza

¹*Atlassian - Agile*. URL: <https://www.atlassian.com/agile>

regole nè pianificazione. La mancanza di struttura e documentazione ha portato alla “crisi del software”, evidenziando la necessità di una vera e propria disciplina.

Il primo tentativo fu il modello sequenziale (a cascata), ispirato alla produzione industriale. Esso prevedeva fasi rigide e successive (analisi, progettazione, implementazione, test e rilascio), senza possibilità di tornare indietro alla fase precedente. Nonostante l'introduzione dell'ordine e del controllo, si è poi rilevato un modello eccessivamente rigido che non permetteva modifiche dei requisiti in corso d'opera o l'integrazione di feedback tempestivi.

Per superare questa rigidità si affermano i modelli incrementali e iterativi:

- Incrementale: le funzionalità vengono aggiunte in maniera progressiva, per fornire valore in tempi brevi e ridurre il rischio totale del progetto;
- Iterativo: si basa su revisione e raffinamento continuo, adatto per gestire incertezze. Presenta però il rischio di non convergenza e aumento dei costi.

Questi approcci hanno permesso maggiore adattabilità, ma senza una reale strategia per gestire il feedback continuo e il *debito tecnico*_G.

Un altro modello di sviluppo è quello a componenti, basato sul riutilizzo di parti già esistenti con l'obiettivo di raggiungere efficienza e scalabilità. Esso però non riesce ad affrontare la dinamicità dei progetti moderni.

Il modello Agile nasce quindi come reazione alla rigidità dei modelli precedenti, puntando sull'adattabilità, sull'interazione continua con il cliente e soprattutto sul feedback rapido e costante, essendo però consapevoli della necessità di migliorare il debito tecnico accumulato.

2.1.2 Manifesto Agile

La base del modello di sviluppo Agile si concretizza nel Manifesto Agile², pubblicato nel 2001 da un gruppo di 17 esperti dello sviluppo software. Esso rappresenta il punto di svolta concettuale e pratico per l'adozione della metodologia Agile, articolandosi intorno a quattro valori fondamentali:

²Manifesto Agile. URL: <https://agilemanifesto.org/iso/it/manifesto.html>

- Individui e interazioni più che processi e strumenti: si pone al centro l'importanza del team e della comunicazione diretta tra le persone, rispetto all'eccessiva dipendenza da strumenti e procedure;
- Software funzionante più che documentazione esaustiva: il valore è misurato sulla base di ciò che effettivamente funziona, anziché sulla quantità e completezza della documentazione prodotta;
- Collaborazione con il cliente più che negoziazione dei contratti: la partecipazione attiva del cliente è considerata essenziale, anche oltre quanto previsto da un contratto formale. Il cliente diventa parte integrante del processo;
- Rispondere al cambiamento più che seguire un piano: la capacità di adattarsi ai cambiamenti viene preferita rispetto all'aderenza rigida a un piano iniziale.

2.1.3 Caratteristiche distintive del metodo Agile

L'approccio Agile è concepito per affrontare progetti complessi e in continua evoluzione, dove la comprensione del problema si sviluppa progressivamente, e dove il coinvolgimento con gli stakeholder e la capacità di adattamento diventano elementi strategici per il successo del prodotto finale. La sua efficacia nei contesti moderni deriva infatti dalle seguenti caratteristiche:

- Sviluppo iterativo e incrementale: il lavoro viene suddiviso in cicli brevi e incrementi successivi di prodotto funzionante, per consentire una validazione continua;
- Feedback costante e miglioramento continuo: ogni fase del progetto prevede momenti di revisione e valutazione, attraverso i quali è possibile correggere la rotta e migliorare il prodotto;
- Team cross-funzionali e auto-organizzati: i gruppi di lavoro possiedono competenze eterogenee e autonomia decisionale, migliorando reattività e senso di responsabilità;

- Adattabilità ai cambiamenti: Agile considera il cambiamento come una risorsa preziosa per migliorare il prodotto e rispondere meglio alle esigenze degli utenti.

2.2 Scrum

Scrum³ è un framework Agile iterativo, progettato per aiutare i team a sviluppare prodotti complessi in modo collaborativo e adattivo. Si basa su cicli di lavoro regolari chiamati Sprint, durante i quali il team produce incrementi di prodotto potenzialmente rilasciabili. Il framework definisce ruoli, eventi (cerimonie) e artefatti specifici per facilitare il raggiungimento degli obiettivi che il modello Agile si pone.

2.2.1 Concetti essenziali del framework Scrum

Per comprendere e applicare correttamente il framework Scrum, è fondamentale acquisire familiarità con una serie di concetti chiave che costituiscono la base operativa di ogni Sprint. Di seguito si presentano i principali elementi costitutivi:

- Sprint: è un intervallo di tempo fisso durante il quale viene sviluppato un incremento di prodotto;
- Product Backlog: è un elenco ordinato e continuamente aggiornato di tutte le funzionalità, requisiti e modifiche desiderate per il prodotto. Gestito dal Product Owner, rappresenta la fonte principale da cui vengono selezionati gli elementi da sviluppare durante ciascuno Sprint;
- Sprint Backlog: è il piano operativo dello Sprint, comprendente gli elementi selezionati dal Product Backlog. È uno strumento dinamico che viene costantemente aggiornato, in base al progresso giornaliero e alle nuove informazioni emerse;

³Atlassian - Scrum. URL: <https://www.atlassian.com/agile/scrum>

- Definition of Done (DoD): è un insieme condiviso di criteri qualitativi e tecnici che stabilisce quando un elemento del backlog può considerarsi realmente completato. La DoD garantisce che ogni incremento sia conforme agli standard di qualità attesi e pronto per l'integrazione o il rilascio;
- Incremento: è il risultato dell'attività svolta durante lo Sprint, ovvero l'insieme di tutti gli elementi completati che rispettano la Definition of Done. L'incremento deve essere funzionante, coerente e in uno stato potenzialmente rilasciabile.

2.2.2 Ruoli

Il framework Scrum prevede tre ruoli principali, che insieme costituiscono lo Scrum Team:

- Product Owner: è responsabile di massimizzare il valore del prodotto risultante dal lavoro del team. Gestisce e ordina il Product Backlog, assicurandosi che gli elementi siano chiari, visibili e compresi dal team. Collabora strettamente con gli stakeholder per allineare le priorità di business;
- Scrum Master: agisce come facilitatore e coach per il team Scrum. Aiuta tutti a comprendere e applicare correttamente il framework, rimuove impedimenti e promuove un ambiente di lavoro efficace;
- Developer: sono i membri del team che lavorano alla creazione dell'incremento di prodotto. Sono responsabili della pianificazione del lavoro durante lo Sprint, del mantenimento della qualità e dell'adattamento continuo per raggiungere l'obiettivo dello Sprint.

2.2.3 Sprint e cerimonie

Lo Sprint costituisce il fulcro del framework Scrum, configurandosi come un intervallo temporale di durata fissa – solitamente compreso tra due e quattro settimane – all'interno del quale il team Scrum lavora per realizzare un incremento di prodotto. Tale incremento consiste in una versione concretamente funzionante del prodotto, che rispetta la “Definition of Done” stabilita dal

team. Ogni incremento deve apportare un valore aggiunto rispetto agli stati precedenti del prodotto, rappresentando così un avanzamento tangibile e misurabile verso l'obiettivo finale. Questo concetto è coerente con i principi fondanti dell'approccio empirico su cui si basa Scrum, ovvero l'ispezione, l'adattamento e la trasparenza.

L'empirismo in Scrum presuppone che la conoscenza derivi dall'esperienza e che le decisioni debbano essere basate sull'osservazione diretta dei risultati. L'incremento prodotto in ogni Sprint diventa quindi uno strumento centrale per ispezionare il progresso, raccogliere feedback da stakeholder e utenti, e adattare di conseguenza le prossime iterazioni. Ciò consente al team di rispondere rapidamente a eventuali cambiamenti nei requisiti, nel mercato o nelle priorità, mantenendo al tempo stesso un orientamento continuo al valore.

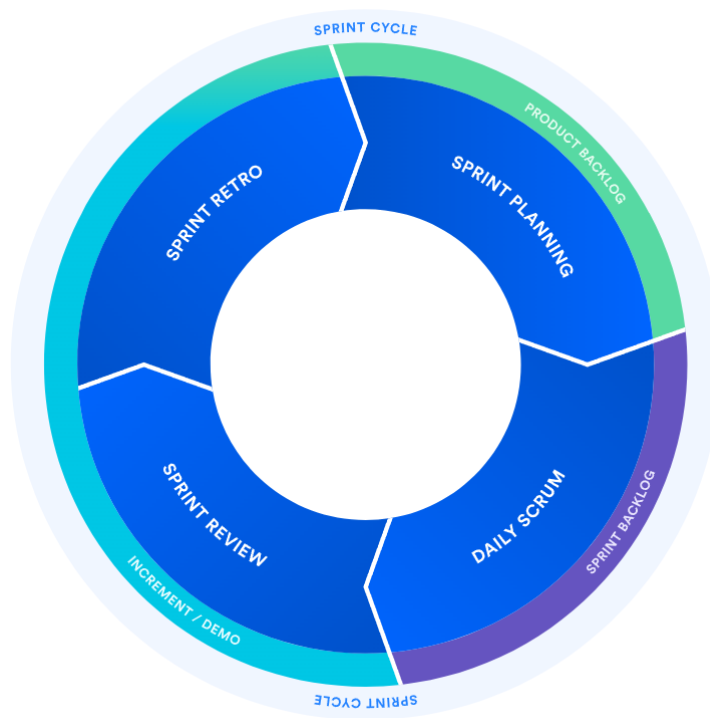


Figura 2.1: Cerimonie Sprint - Atlassian.

Ogni Sprint ha inizio con una fase di pianificazione (cerimonia chiamata **Sprint Planning**), durante la quale l'intero Scrum Team collabora per definire quali elementi del Product Backlog potranno essere completati entro la fine dello Sprint e in che modo tali attività verranno realizzate.

Durante lo svolgimento dello Sprint, è importante evitare qualsiasi modifica che possa compromettere il raggiungimento di quanto pianificato. Tuttavia, è ammesso un certo grado di flessibilità operativa, in quanto il team può rinegoziare il lavoro pianificato con il Product Owner qualora emergano nuove informazioni od ostacoli. Questa caratteristica consente di operare in modo adattivo, mantenendo al contempo un forte orientamento al risultato. Un ruolo cruciale in tale adattamento è svolto dal **Daily Scrum**, un incontro giornaliero della durata massima di 15 minuti, durante il quale i membri del team di sviluppo ispezionano i progressi e pianificano il lavoro delle successive 24 ore. Questo evento favorisce la trasparenza e consente un allineamento costante, offrendo l'opportunità di identificare tempestivamente ostacoli, riallocare compiti, e rivedere le priorità operative nel rispetto della pianificazione iniziale.

Al termine dello Sprint si svolgono altre due cerimonie fondamentali: la **Sprint Review** e la **Sprint Retrospective**. La Review è un incontro collaborativo con gli stakeholder, finalizzato alla presentazione dell'incremento realizzato e alla raccolta di feedback utile per l'evoluzione del prodotto. La Retrospective, invece, rappresenta uno spazio di riflessione interna al team Scrum, volto a individuare aspetti di miglioramento sia in termini di processi che di dinamiche collaborative. Questi momenti conclusivi rivestono un ruolo essenziale nel ciclo di miglioramento continuo e di ricezione dei feedback.

2.3 Benefici e sfide dell'approccio Agile

L'adozione dell'approccio Agile, e in particolare di Scrum, offre numerosi vantaggi:

- Flessibilità e adattabilità: i team possono rispondere rapidamente ai cambiamenti nei requisiti, nelle condizioni di mercato o nelle priorità;
- Consegna continua di valore: gli incrementi regolari permettono di fornire funzionalità utilizzabili in tempi brevi;
- Maggiore coinvolgimento degli stakeholder: la collaborazione costante con i clienti assicura che il prodotto finale risponda alle loro esigenze;

- Miglioramento continuo: le retrospettive promuovono l'apprendimento e l'ottimizzazione dei processi interni.

Tuttavia, la transizione verso un approccio Agile può comportare diverse criticità, specialmente in contesti organizzativi tradizionalmente legati a metodologie predittive e fortemente gerarchiche. Introdurre un paradigma basato sulla flessibilità e sull'autonomia richiede un cambiamento culturale profondo e un forte investimento in formazione e cambiamento organizzativo. Le principali sfide riscontrabili includono:

- Elevato livello di autonomia e responsabilità: l'efficacia dei team Agile dipende dalla loro capacità di auto-organizzarsi e prendere decisioni in modo autonomo. Questo richiede un alto grado di disciplina e responsabilizzazione individuale;
- Esigenza di trasparenza e collaborazione continua: in contesti con team distribuiti geograficamente o in ambienti scarsamente cooperativi, mantenere un elevato livello di trasparenza può risultare particolarmente complesso;
- Difficoltà nelle stime iniziali: la natura iterativa e adattiva di Agile rende meno immediato stimare con precisione tempi, costi e risorse necessarie, specialmente nei primi cicli di adozione. In assenza di esperienza e metriche consolidate, ciò può generare incertezza nella pianificazione e nella gestione delle aspettative.

2.4 Applicazione di Agile Scrum in Wavelop SRL

Durante il periodo di stage presso Wavelop SRL, ho potuto osservare l'applicazione concreta del framework Agile Scrum all'interno di un'organizzazione che opera nello sviluppo software. L'adozione di Scrum da parte dell'azienda riflette un orientamento alla flessibilità, alla collaborazione costante e al miglioramento continuo, in linea con le pratiche agili più consolidate.

Il ciclo di sviluppo è strutturato in Sprint di durata regolare, durante i quali i team si concentrano sul completamento di un insieme di funzionalità prioritarie, selezionate durante la fase di Sprint Planning. Il lavoro pianificato viene tracciato e aggiornato attraverso una dashboard integrata in *GitLab*_G, che funge da strumento centrale per la gestione del Product Backlog, dello Sprint Backlog e dello stato di avanzamento delle singole attività.

Il Product Backlog viene alimentato in modo strutturato a partire dall'analisi iniziale del progetto, durante la quale vengono individuate le epiche⁴ principali. Queste rappresentano funzionalità ad alto livello o macro-obiettivi del sistema, e vengono successivamente scomposte in user stories⁵ più granulari, ognuna delle quali descrive una specifica esigenza o funzionalità dal punto di vista dell'utente finale e che comprende l'implementazione della funzionalità sia frontend che backend. Le user stories così generate costituiscono l'unità di lavoro fondamentale, e vengono progressivamente raffinate e priorizzate nel backlog in funzione del valore di business, della complessità stimata e della dipendenza tra le attività. A ciascuna user story viene inoltre assegnato uno story point⁶, una misura astratta della complessità e dello sforzo richiesto per la sua realizzazione. In Wavelop SRL viene adottata la classica scala di Fibonacci estesa (1, 2, 3, 5, 8, 13, 21), che consente di esprimere in modo relativo le differenze di complessità tra le varie attività, favorendo una stima rapida e condivisa all'interno del team di sviluppo.

Una prassi quotidiana ben consolidata in Wavelop è lo svolgimento del Daily Stand-up, che avviene ogni mattina alle 9:15 tramite la piattaforma di comunicazione Discord. Durante questo breve incontro, il responsabile del team condivide lo schermo per mostrare la dashboard GitLab e guida il team nella revisione del lavoro. Per ciascun membro vengono discusse tre dimensioni fondamentali:

- Le attività completate il giorno precedente;

⁴*Atlassian - Epics*. URL: <https://www.atlassian.com/agile/project-management/epics>

⁵*Atlassian - User Stories*. URL: <https://www.atlassian.com/agile/project-management/user-stories>

⁶*Atlassian - Story Points*. URL: <https://www.atlassian.com/agile/project-management/estimation>

- Le attività pianificate per la giornata corrente;
- Eventuali ostacoli o impedimenti riscontrati.

Oltre a GitLab, l'azienda fa uso dello strumento Clockify per la registrazione delle ore lavorative e delle attività svolte. Questo permette di mantenere una traccia precisa del tempo impiegato su ciascun task, facilitando la rendicontazione interna, l'analisi della produttività e il monitoraggio dell'effort individuale in ottica di ottimizzazione dei processi.

In conclusione dello Sprint, Wavelop SRL organizza due cerimonie fondamentali previste dal framework Scrum: la Sprint Review e la Sprint Retrospective. La Sprint Review si svolge alla fine di ogni sprint e rappresenta un momento di confronto tra il team di sviluppo, il Product Owner e il committente. Durante questo momento vengono presentati gli incrementi completati e si analizza il grado di raggiungimento degli obiettivi prefissati. L'incontro ha anche una funzione esplorativa, in quanto consente di raccogliere feedback utili e di aggiornare il Product Backlog in vista della pianificazione del ciclo successivo.

Segue poi la Sprint Retrospective, un momento riservato al team per riflettere sulle modalità di lavoro adottate durante lo Sprint appena concluso. In questa fase, ciascun membro può condividere osservazioni su ciò che ha funzionato, su eventuali criticità riscontrate e su possibili azioni di miglioramento. Anche questa cerimonia si svolge in modalità remota, con l'ausilio di strumenti collaborativi digitali, e si configura come un'opportunità strategica per promuovere l'apprendimento continuo, la coesione del team e l'efficienza operativa.

Questa combinazione di strumenti e cerimonie consente a Wavelop di mantenere elevati livelli di trasparenza e coordinamento. La quotidiana visibilità sullo stato delle attività, unita alla misurazione puntuale delle performance, contribuisce a creare un contesto organizzativo orientato alla responsabilità condivisa e al raggiungimento degli obiettivi comuni.

2.5 Gestione del versionamento del codice

All'interno della metodologia di sviluppo adottata da Wavelop, un ruolo fondamentale è ricoperto dalla gestione del versionamento del codice. Per garantire un flusso di lavoro ordinato, tracciabile e collaborativo, l'azienda ha scelto di adottare un modello ispirato al Gitflow Workflow, uno standard ampiamente riconosciuto per lo sviluppo distribuito con Git⁷.

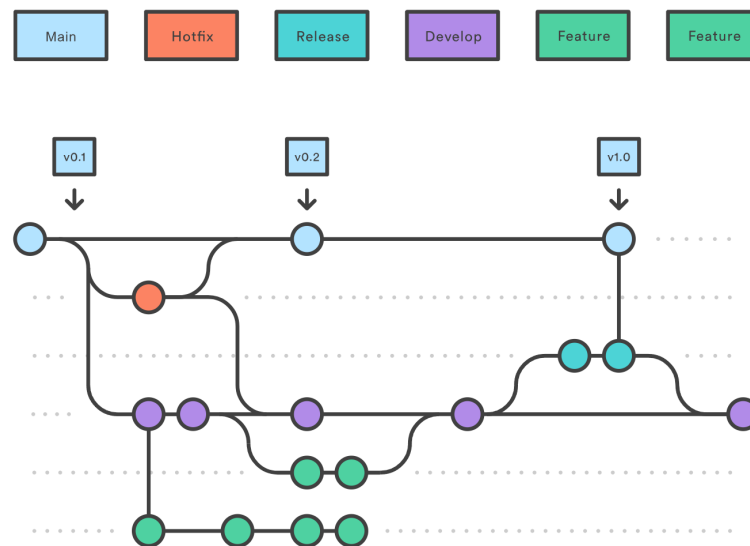


Figura 2.2: Gitflow Workflow - Atlassian.

Questo approccio prevede una struttura ramificata del progetto, in cui ogni ramo ha una funzione specifica nel ciclo di vita del software, come mostrato nella Figura 2.2. In particolare:

- Il ramo **main** contiene esclusivamente versioni stabili del prodotto, pronte per essere rilasciate in produzione;
- Il ramo **develop** funge da linea di sviluppo principale, dove confluiscono le nuove funzionalità dopo essere state testate e validate;
- I **feature branches**, con convenzioni del tipo `feature/{nome}`, vengono utilizzati per sviluppare nuove funzionalità in modo isolato, mantenendo il codice stabile sul ramo di sviluppo principale;

⁷Atlassian - *Gitflow Workflow*. URL: <https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>

- I **release branches**, derivati da **develop**, consentono di preparare una nuova versione, effettuando gli ultimi test, correzioni e documentazioni prima della pubblicazione ufficiale;
- I **hotfix branches**, creati a partire da **main**, permettono di risolvere rapidamente problemi critici riscontrati in produzione, garantendo che le correzioni vengano propagate anche su **develop**.

Grazie a questa organizzazione, il team di sviluppo può lavorare in parallelo su più funzionalità, riducendo il rischio di conflitti e semplificando le attività di testing e rilascio.

2.6 Metodi e processi applicati allo stage

Nell'ambito dello stage ho applicato in prima persona i seguenti metodi e processi:

- Definizione di epiche e user stories;
- Assegnazione story points;
- Sprint: uno sprint iniziale di una settimana per predisporre l'ambiente di lavoro più 5 sprint di una settimana ciascuno;
- Daily scrum: ogni giorno alle 9:15 su piattaforma Discord con tutto il team di Wavelop;
- Sprint review: ogni venerdì, con i tutor aziendali, è prevista la revisione dei progressi svolti e l'analisi degli obiettivi raggiunti durante lo sprint con successiva definizione delle user stories da assegnare allo sprint successivo;
- Versionamento del codice: adozione del metodo Gitflow Workflow.

Capitolo 3

Descrizione dello stage

Questo capitolo è dedicato alla descrizione dell'esperienza di tirocinio curricolare, con particolare attenzione alla presentazione del progetto svolto, alla pianificazione delle attività e all'analisi delle risorse preesistenti che sono state integrate o utilizzate nel corso dello sviluppo.

3.1 Introduzione al progetto Insurance Estimator

Il progetto, chiamato Insurance Estimator, è volto alla realizzazione di un'applicazione web a supporto dell'attività di un agente assicurativo operante nel settore agricolo, con particolare riferimento alla valutazione assicurativa di beni appartenenti a cantine vinicole e coltivatori di vino. Il progetto nasce dall'esigenza di automatizzare e semplificare il processo di calcolo del valore assicurabile di tali beni, a partire da una tabella standard fornita da *Agenzia Veneta per i Pagamenti (AVEPA)_G*.

L'applicazione si propone di leggere e importare i dati contenuti nella tabella AVEPA, per poi effettuare elaborazioni strutturate che includono: raggruppamenti per tipologia di coltivazione o superficie, calcoli relativi alle metrature e determinazione del valore assicurabile del bene. Inoltre, l'interfaccia utente consente la modifica manuale delle righe importate, offrendo così la flessibilità

necessaria per l'adeguamento dei dati a casi specifici o per errori di estrazione dei dati.

Una volta validati i dati da parte dell'utente, il sistema permette la generazione di preventivi assicurativi, distinti per compagnia assicurativa e per singola polizza, rendendo l'intero processo più efficiente e tracciabile.

Oltre alla parte operativa legata alla gestione dei dati, l'applicazione integra un'area di back-office destinata alla configurazione di elementi fondamentali per la corretta generazione dei preventivi. In particolare, è possibile caricare i tassi delle compagnie assicurative, gli schedari tecnici contenenti i dati anagrafici e tabellari del coltivatore e le stime per tipologia di piantagione.

Infine, il sistema implementa funzionalità di gestione dell'anagrafica clienti e la visualizzazione dello storico dei preventivi generati e delle polizze stipulate, fornendo così un supporto completo all'attività quotidiana dell'agente assicurativo.

3.2 Pianificazione del lavoro

Lo stage è iniziato in una fase preliminare dello sviluppo del progetto Insurance Estimator, quando ancora non esisteva alcun sistema implementato. Ciò ha comportato un'attività focalizzata sulla progettazione e l'analisi iniziale, piuttosto che sull'integrazione di componenti preesistenti. L'approccio adottato è stato quindi orientato alla definizione e costruzione dell'intero sistema, a partire dalla raccolta e formalizzazione dei requisiti, in assenza di una struttura predefinita. La pianificazione delle attività si è articolata in tre fasi principali:

- **Analisi:** ha interessato l'intero sistema applicativo, non limitandosi agli obiettivi specifici dello stage. Tale fase ha condotto all'elaborazione delle user stories e delle epiche necessarie per organizzare e guidare lo sviluppo della web app;
- **Progettazione:** ha riguardato la definizione dell'architettura del database, dell'interfaccia utente (frontend) e del sistema di backend, compreso il flusso di estrazione, elaborazione e gestione dei dati;

- Sviluppo: si è concentrato sull'implementazione delle componenti progettate e sulla realizzazione delle user stories correlate agli obiettivi previsti nel piano formativo dello stage, per poi proseguire anche con ulteriori stories aggiuntive.

3.3 Risorse, tecnologie e strumenti utilizzati

Come precedentemente evidenziato, al momento dell'avvio dello stage il progetto Insurance Estimator si trovava ancora in una fase iniziale di definizione, con una struttura progettuale non ancora formalizzata. Le risorse disponibili nella fase preliminare erano limitate a un insieme di documenti campione, comprendenti schedari viticoli, stime del valore in euro per quintale in funzione della tipologia di vigneto, e preventivi redatti manualmente da parte dell'agente assicurativo.

In aggiunta, l'azienda ha messo a disposizione il codice sorgente di un'integrazione sviluppata in un progetto da un precedente stagista. Tale codice, tra le varie funzionalità, includeva un modulo dedicato all'estrazione di dati tabellari da documenti PDF, basato sull'impiego di tecniche di Intelligenza Artificiale.

Infine, presento le tecnologie e gli strumenti selezionati e adottati per la realizzazione del progetto.

GitLab¹

GitLab è una piattaforma DevOps che integra funzionalità di versionamento del codice, gestione delle issue, continuous integration/continuous delivery (CI/CD) e collaborazione tra sviluppatori. È stata utilizzata per il versionamento del codice tramite Git, per la gestione delle user stories e delle epiche attraverso il sistema di ticketing integrato, nonché per il monitoraggio dell'avanzamento delle attività tramite *milestone*_G. L'adozione di GitLab ha contribuito a strutturare il processo di sviluppo in modo tracciabile, collaborativo e in linea con le metodologie Agile.

¹ *GitLab*. URL: <https://about.gitlab.com/>

React²

React è una libreria JavaScript per la costruzione di interfacce utente interattive, basata su un approccio dichiarativo e component-based. Consente lo sviluppo modulare e riutilizzabile di componenti, facilitando la gestione dello stato e del ciclo di vita dell'interfaccia. È stata scelta per la sua ampia diffusione, il supporto fornito dalla documentazione, l'integrazione con l'ecosistema JavaScript/TypeScript e la facilità con cui consente la gestione di interfacce dinamiche e responsive.

Shadcn/ui³

Shadcn/ui è una libreria di componenti React costruita su Tailwind CSS, progettata per essere componibile, accessibile e altamente personalizzabile. È stata selezionata per accelerare lo sviluppo dell'interfaccia utente attraverso componenti già pronti, rispettando le best practice in termini di accessibilità e design coerente, oltre che per la sua grafica minimale e semplice che evita di appesantire l'interfaccia.

TanStack Table⁴

TanStack Table è una libreria headless per la gestione avanzata di tabelle in React, che offre funzionalità come ordinamento, filtraggio e impaginazione. È stata adottata per la sua flessibilità e per l'elevato livello di controllo offerto sulla logica di visualizzazione dei dati tabellari, essenziale per la visualizzazione dei dati tabellari estratti da uno schedario e per il loro raggruppamento in sottoinsiemi.

²React. URL: <https://react.dev/>

³Shadcn/ui. URL: <https://ui.shadcn.com/>

⁴TanStack Table. URL: <https://tanstack.com/table/latest>

Zod⁵

Zod è una libreria TypeScript per la definizione e validazione degli schemi dei dati in modo type-safe. È stata scelta per garantire la coerenza e la robustezza dei dati validati, riducendo il rischio di errori dovuti a input non conformi.

React Router⁶

React Router è una libreria di routing che permette la gestione della navigazione in applicazioni React a più viste. È stata utilizzata per implementare una navigazione chiara e scalabile tra le diverse sezioni dell'applicazione, con supporto a routing annidato e protetto.

i18next⁷

i18next è una libreria JavaScript per l'internazionalizzazione di applicazioni, con supporto a namespace, fallback e caricamento asincrono. È stata scelta per la necessità di supportare più lingue nell'interfaccia utente, garantendo un'adeguata flessibilità nella gestione delle traduzioni.

Node.js⁸

Node.js è un ambiente di esecuzione JavaScript lato server, orientato alla costruzione di applicazioni scalabili. È stato adottato come ambiente di esecuzione del backend per l'ottima integrazione con TypeScript e le prestazioni elevate nella gestione di I/O asincrono.

TypeScript⁹

TypeScript è un superset di JavaScript che introduce il tipaggio statico e strumenti avanzati per lo sviluppo. È stato utilizzato per migliorare la qualità

⁵ *Zod*. URL: <https://zod.dev/>

⁶ *React Router*. URL: <https://reactrouter.com/>

⁷ *i18next*. URL: <https://www.i18next.com/>

⁸ *Node.js*. URL: <https://nodejs.org/en>

⁹ *TypeScript*. URL: <https://www.typescriptlang.org/>

del codice, agevolare il refactoring e ridurre gli errori a tempo di compilazione, aumentando l'affidabilità generale del progetto.

Fastify¹⁰

Fastify è un framework web per Node.js che si distingue per leggerezza, modularità e prestazioni elevate. È stato selezionato per la sua rapidità nella gestione delle richieste HTTP, il supporto alla validazione degli input tramite schema e la semplicità di integrazione con librerie esterne.

Python¹¹

Python è un linguaggio di programmazione ad alto livello, noto per la sua leggibilità e ricca dotazione di librerie per la manipolazione dei dati. È stato impiegato per lo sviluppo degli script di estrazione dei dati da PDF, grazie alla disponibilità di strumenti specifici per il trattamento documentale.

Camelot¹²

Camelot è una libreria Python specializzata nell'estrazione di tabelle da file PDF mediante parsing strutturato. È stata utilizzata per l'efficacia nell'estrazione di tabelle da PDF ben strutturati, consentendo il recupero di dati in formato tabellare con elevata precisione.

pdfplumber¹³

pdfplumber è una libreria Python che permette l'accesso ai contenuti testuali e grafici di file PDF, inclusi layout e coordinate. È stata impiegata per l'estrazione avanzata di contenuti testuali e strutture non tabellari, integrando le capacità di Camelot nei casi in cui quest'ultimo risultava insufficiente.

¹⁰ *Fastify*. URL: <https://fastify.dev/>

¹¹ *Python*. URL: <https://www.python.org/>

¹² *Camelot*. URL: <https://camelot-py.readthedocs.io/en/master/>

¹³ *pdfplumber*. URL: <https://github.com/jsvine/pdfplumber>

MongoDB¹⁴

MongoDB è un sistema di gestione di basi di dati NoSQL orientato ai documenti. Utilizza una struttura flessibile basata su documenti, permettendo di modellare dati complessi in modo naturale e dinamico. È stato scelto per la sua flessibilità nella modellazione dei dati, che ben si adatta a strutture non rigidamente relazionali, e per la sua semplicità di integrazione con stack JavaScript/TypeScript.

Docker¹⁵

Docker è una piattaforma open-source che consente di automatizzare il deployment di applicazioni all'interno di *container* software. Utilizza un motore leggero e performante per orchestrare la creazione, l'esecuzione e la distribuzione dei container, rendendo il processo di sviluppo più modulare e ripetibile. Docker è stato adottato per assicurare portabilità e coerenza tra gli ambienti di sviluppo, test e produzione. Grazie all'isolamento dei container, ha permesso di evitare conflitti tra dipendenze e di semplificare il provisioning del backend e dei servizi ausiliari (es. database). Inoltre, l'utilizzo di immagini versionate ha facilitato l'integrazione continua e la distribuzione riproducibile del sistema.

¹⁴MongoDB. URL: <https://www.mongodb.com/>

¹⁵Docker. URL: <https://www.docker.com/>

Capitolo 4

Analisi

Questo capitolo descrive il processo di analisi svolto per il progetto Insurance Estimator, a partire dalla definizione della struttura generale dell'applicativo, passando per la progettazione dei wireframes, fino all'elaborazione delle epiche, delle user stories e all'assegnazione degli story points.

4.1 Elementi iniziali

L'analisi dei requisiti è risultata essenziale nello sviluppo di questo progetto in quanto era presente solo un'idea iniziale, con dei requisiti ancora non ben strutturati e da definire. I primi approcci all'analisi si sono quindi focalizzati sulla definizione della struttura dell'applicativo assieme ai tutor aziendali. Questa ha rappresentato il punto d'inizio per la continuazione dell'analisi attraverso la creazione di wireframes, che hanno concretizzato i requisiti da soddisfare. L'attività di analisi si è poi conclusa con la definizione delle epiche e delle user stories, secondo un approccio iterativo tipico della metodologia agile, a cui è stata associata una stima degli effort tramite story points.

4.2 Struttura applicativo e wireframes

L'analisi è iniziata con la definizione della struttura dell'applicativo ad alto livello, esposta nella Figura [4.1](#):

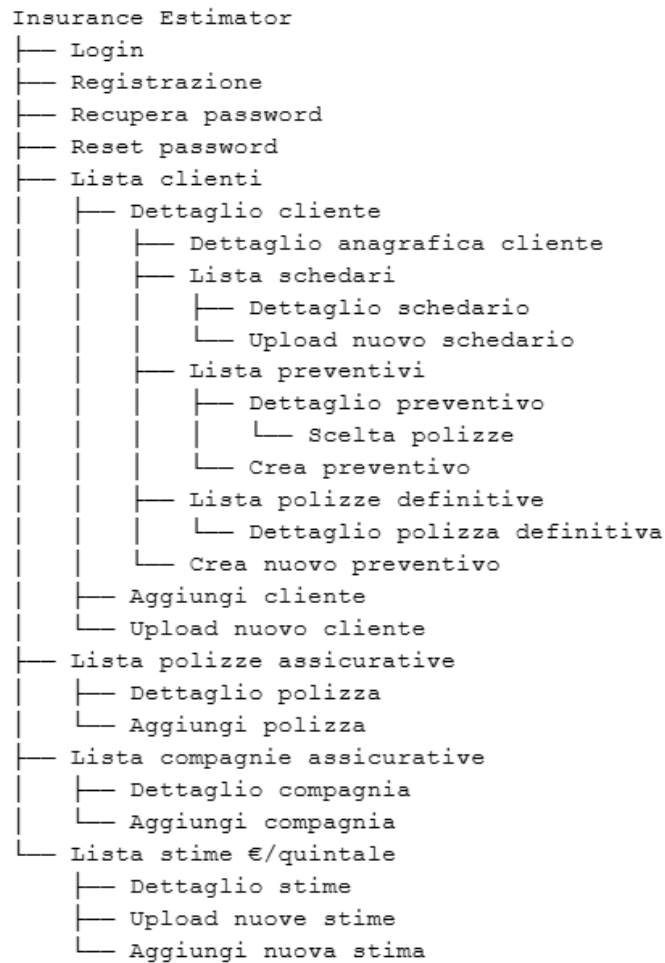
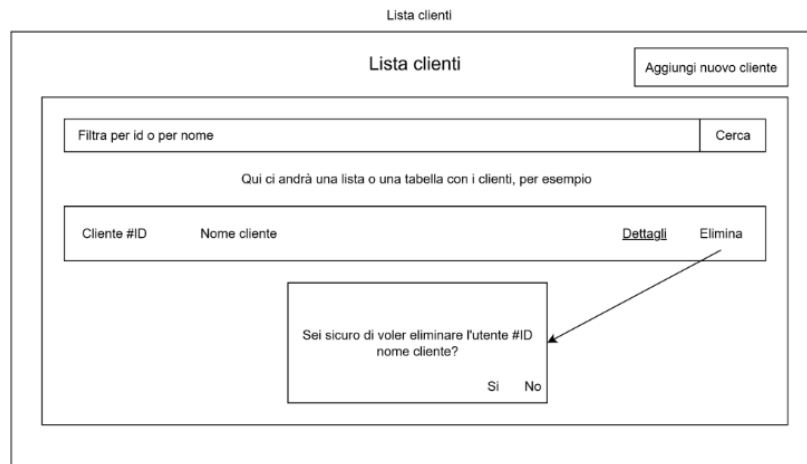


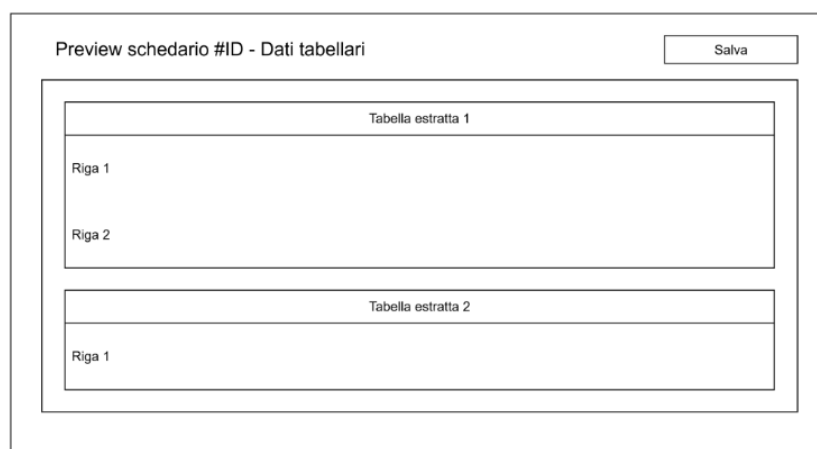
Figura 4.1: Struttura ad alto livello dell'applicativo.

Questa struttura ha posto le basi per lo sviluppo dei wireframes, ovvero delle rappresentazioni visive statiche delle interfacce dell'applicativo, utili per delineare l'organizzazione dei contenuti e i principali flussi utente. Essi sono risultati fondamentali per comprendere i flussi da implementare nell'applicativo e per individuare eventuali mancanze o incongruenze. Di seguito riporto delle figure di esempio di alcuni wireframes prodotti:

- Wireframe lista: rappresenta la struttura generale delle pagine di tipo lista, utilizzata anche per le polizze, le compagnie e le stime (Figura 4.2);

**Figura 4.2:** Wireframe lista clienti.

- Wireframe schedario: mostra l'anteprima dei dati tabellari estratti da un file PDF caricato. Questa sezione rappresenta una delle componenti centrali dell'applicativo, poiché visualizza i risultati del processo di estrazione automatica (Figura 4.3);

**Figura 4.3:** Wireframe schedario.

- Wireframe preventivo: rappresenta la struttura di un preventivo generato automaticamente a partire dai dati estratti e dalle scelte utente (Figura 4.4).

Preventivo #ID v. 1.0

Stato: preventivo/bozza

Modifica Invia preventivo v. 1.0 Promuovi a polizza

Dati del cliente

Tabella preventivo

Riga 1

Riga 2

Riga 3

Figura 4.4: Wireframe preventivo.

4.3 Epiche, user stories e story points

Nella fase finale dell'analisi è stato adottato un approccio agile per la scomposizione delle funzionalità in epiche, user stories e la stima mediante story points.

Epiche

Le epiche rappresentano insiemi di funzionalità correlate, che definiscono macro-obiettivi del sistema. Per il progetto ho individuato le seguenti epiche:

1. **Autenticazione:** gestione di login, registrazione, recupero e reset della password;
2. **Lista clienti:** visualizzazione e gestione dei clienti, inclusi schedari, preventivi, polizze e caricamento documenti;
3. **Modelli di polizze:** creazione, modifica e consultazione delle polizze assicurative presenti all'interno dell'applicativo;
4. **Compagnie assicurative:** gestione dell'anagrafica delle compagnie assicurative;
5. **Stime €/quintale:** aggiornamento e consultazione delle stime;

6. **Gestione errori e operazioni CRUD:** validazione, gestione degli errori utente e operazioni di base (creazione, lettura, aggiornamento, eliminazione) su tutte le entità del sistema.

User stories

Ogni epica è stata scomposta in più user stories, ovvero descrizioni sintetiche di funzionalità dal punto di vista dell'utente finale. Ogni user story segue la forma:

[**Come**] persona [**voglio**] obiettivo [**per**] beneficio atteso

Ad esempio:

- Come utente autenticato, voglio poter filtrare gli schedari per il numero e per la data di upload per cercarli in modo più rapido;
- Come utente autenticato, voglio poter creare un nuovo preventivo da uno schedario esistente per poter scegliere e filtrare tra gli schedari disponibili.

Per ogni entità presente all'interno del sistema - schedari, stime, compagnie, preventivi, clienti, ecc. - ho definito le storie che riguardano tutte le operazioni CRUD (Create - Read - Update - Delete), oltre che la funzionalità di filtraggio tra i dati presenti nel database e di lettura ed estrazione dei dati dai file PDF caricati. Le storie appartenenti all'epica "Lista clienti" hanno richiesto particolare dettaglio, includendo funzionalità per:

- Gestione degli schedari;
- Visualizzazione e creazione dei preventivi;
- Caricamento e parsing dei file PDF;
- Navigazione tra le entità collegate (cliente - schedario - preventivo - polizza).

Story points

Ad ogni user story ho assegnato un valore in story points, in base alla complessità tecnica, al carico cognitivo e al tempo stimato per l'implementazione. Come indicato dall'azienda, ho utilizzato una scala di tipo Fibonacci (1, 2, 3, 5, 8, ecc.), tipica nei contesti agile, per facilitare la comparazione relativa tra task.

Questo processo di analisi ha fornito una visione chiara delle funzionalità da realizzare e ha permesso di pianificare lo sviluppo in modo incrementale e tracciabile.

Integrazione con GitLab

Per garantire la tracciabilità delle attività e organizzare in modo efficace il lavoro di sviluppo, ho inserito ogni user story all'interno del sistema di ticketing offerto da GitLab. Per ciascuna storia, ho provveduto ad associare l'epica di riferimento e ad assegnare un valore di story points, sfruttando le funzionalità di gestione delle issue e delle milestone offerte dalla piattaforma.

Questa organizzazione ha permesso non solo di monitorare lo stato di avanzamento delle singole funzionalità, ma anche di facilitare la pianificazione e la suddivisione del lavoro in sprint, in linea con i principi del metodo Agile adottato. L'impiego di GitLab ha quindi costituito un supporto fondamentale per mantenere coerenza tra analisi, progettazione e implementazione del sistema.

Conclusa la fase di definizione delle user stories, ho proseguito l'analisi con lo studio delle tecnologie per l'estrazione automatica dei dati, in particolare le tecniche di OCR e quelle basate sull'Intelligenza Artificiale. Il capitolo successivo è dedicato all'approfondimento teorico di entrambe le soluzioni, seguito da un confronto tra i due approcci e dalla motivazione della scelta adottata per la realizzazione del progetto.

Capitolo 5

Tecniche OCR e di Intelligenza Artificiale per l'estrazione di dati

Questo capitolo ha l'obiettivo di analizzare e confrontare due approcci principali all'estrazione automatica di dati: le tecniche OCR tradizionali e quelle basate su AI. Verranno analizzati l'evoluzione storica, il loro funzionamento, i punti di forza e le limitazioni, terminando infine con un confronto diretto tra le due tecnologie.

5.1 Introduzione

L'estrazione automatica di dati fa riferimento all'insieme di tecniche e strumenti in grado di convertire contenuti testuali - spesso contenuti in immagini, PDF o documenti strutturati - in dati digitali leggibili, interpretabili e utilizzabili da sistemi informatici.

Tradizionalmente, questa operazione è stata affidata alla tecnologia OCR (Optical Character Recognition), che consente il riconoscimento ottico dei caratteri. Tuttavia, l'evoluzione tecnologica e la diffusione delle tecniche di Intelligenza Artificiale (AI), in particolare quelle basate su Deep Learning e Natural Language Processing (NLP), hanno reso possibile non solo la trascrizione meccanica del testo, ma anche l'interpretazione semantica del contenuto e la strutturazione automatica dei dati.

5.2 Tecniche OCR tradizionali

5.2.1 Definizione e contesto

L'Optical Character Recognition (OCR) è un'area di ricerca che coinvolge l'intelligenza artificiale, la visione artificiale e il riconoscimento di pattern. Si tratta di una tecnologia che consente di analizzare documenti contenenti testo - stampato o manoscritto - al fine di convertirli in formato digitale, rendendoli così leggibili da una macchina e modificabili attraverso comuni editor di testo. L'OCR trova applicazione in numerosi ambiti, tra cui la digitalizzazione di archivi cartacei, la lettura automatica di moduli e fatture, il supporto all'accessibilità per persone con disabilità visive e l'estrazione di informazioni da immagini e video.

5.2.2 Evoluzione storica

Il riconoscimento ottico dei caratteri ha una storia lunga oltre un secolo, iniziata con i primi esperimenti meccanici svolti da Goldberg e Fournier d'Albe tra il 1912 e il 1914, focalizzati sul supporto della lettura per le persone ipovedenti. L'OCR ha conosciuto una progressiva evoluzione tecnica, passando da macchine rigide e limitate a sistemi intelligenti in grado di comprendere la struttura e il contenuto di un documento.

Anni '40-'60: Le origini

I primi sistemi OCR elettronici riconoscevano un numero ristretto di caratteri prestabiliti e solo in font standard. Il loro funzionamento si basava su tecniche che cercavano di semplificare l'elaborazione riducendo i caratteri a forme riconoscibili meccanicamente. A partire dagli anni '40, diverse aziende statunitensi iniziarono a sviluppare "macchine da lettura" per non vedenti, ma il primo vero e proprio impiego su larga scala avvenne nel 1965, quando il sistema postale statunitense adottò un sistema OCR per la lettura automatica dei codici posta-

li, utilizzato per facilitare lo smistamento della corrispondenza tramite codici a barre.

Anni '70-'90: Verso la generalizzazione

Nel 1974 Ray Kurzweil introdusse il primo sistema omni-font, capace di leggere testi in molteplici caratteri tipografici, aprendo definitivamente la strada a un uso pratico e commerciale dell'OCR. Negli anni '80 e '90 si assistette all'introduzione di algoritmi più sofisticati, come il classificatore bayesiano e le prime reti neurali artificiali. Venne anche riconosciuta l'importanza del supporto alla scrittura a mano, seppure inizialmente con accuratezza limitata.

Dal 2000 in poi: OCR e Intelligenza Artificiale

L'evoluzione più significativa è avvenuta con l'integrazione del Machine Learning e del Deep Learning. L'uso di reti neurali convoluzionali (CNN) e ricorrenti (RNN) ha reso possibile il riconoscimento accurato di testi complessi, lingue non latine e scritture irregolari. L'applicazione del Deep Learning all'OCR - in particolare in documenti degradati - ha indubbiamente portato a significativi miglioramenti rispetto ai metodi tradizionali, soprattutto per la sua capacità di comprendere anche il significato delle informazioni estratte. Tuttavia, l'accuratezza dipende ancora fortemente dalla qualità dei dati e dal dominio applicativo.

5.2.3 Architettura e funzionamento


L'architettura di un sistema OCR tradizionale si basa su una pipeline composta da fasi sequenziali, ciascuna delle quali svolge un ruolo essenziale per il corretto riconoscimento del testo. L'obiettivo è convertire una rappresentazione grafica (*immagine raster* ) in una sequenza di caratteri codificati, mantenendo quanto più possibile l'integrità semantica e visiva del contenuto originale.



Figura 5.1: OCR pipeline.

5.2.3.1 Acquisizione del documento

La prima fase consiste nella digitalizzazione del documento da elaborare, tramite scanner, PDF generati da scansioni, fotocamere o smartphone.

Il risultato è un'immagine raster (formati comuni: TIFF, PNG, JPEG), che rappresenta la pagina come una griglia di pixel. Questo primo step è fondamentale in quanto l'accuratezza delle fasi successive dipende fortemente dalla qualità dell'acquisizione.

5.2.3.2 Pre-elaborazione dell'immagine (Image Preprocessing)

Questa fase ha lo scopo di standardizzare e migliorare l'immagine, riducendo rumore e variazioni visive che potrebbero compromettere il riconoscimento del contenuto. Le operazioni tipiche includono:

- **Binarizzazione:** conversione in immagine binaria (bianco e nero), utile per separare in modo chiaro il testo dallo sfondo. L'immagine, in scala di grigi, viene convertita in un'immagine a due livelli, pixel neri (valore 0) e pixel bianchi (valore 255). Tra le tecniche più comuni troviamo:
 - Threshold globale: imposta una soglia fissa (per esempio 127) che determina la classificazione di ogni pixel. Se il valore del pixel è minore della soglia allora sarà classificato come testo, se il valore del pixel è maggiore della soglia sarà considerato come sfondo;
 - Otsu: calcola automaticamente la soglia ottimale, analizzando l'istogramma dei toni di grigio. Questo metodo funziona bene quando c'è molto contrasto tra testo e sfondo, se invece sono presenti delle zone in cui la luce è irregolare potrebbe generare errori;

- Adaptive Thresholding: divide l'immagine in piccole regioni e calcola una soglia diversa per ogni zona, adattandosi a situazioni in cui la luce è irregolare e sono presenti molte zone d'ombra.
- **Rimozione del rumore:** applicazione di filtri per eliminare artefatti visivi e imperfezioni attraverso:
 - Filtro mediano: elimina il rumore impulsivo, ovvero pixel errati che appaiono come puntini neri o bianchi isolati. Per ogni pixel, il filtro esamina una finestra centrata su di lui e prende la mediana, sostituendola al valore del pixel centrale;
 - Morfologia matematica: lavora su immagini binarie, applicando le operazioni di erosione e di dilatazione. L'operazione di erosione ha lo scopo di rimpicciolire le zone nere attraverso una "maschera" che viene fatta scorrere sull'immagine e permette di mantenere un pixel nero solo se tutti i pixel vicini sono anch'essi neri. L'operazione di dilatazione, invece, ha l'obiettivo di ingrandire le zone nere; in questo caso un pixel diventa nero se almeno uno dei suoi vicini lo è. Spesso queste metodologie vengono utilizzate in combinazione, in base alla qualità dell'immagine.
- **Normalizzazione della luminosità:** ha lo scopo di rendere più uniforme l'illuminazione su tutta l'immagine. Corregge zone sovraesposte/sottoesposte tramite equalizzazione dell'istogramma, ovvero distribuendo uniformemente i valori di luminosità grazie all'aumento del contrasto tra testo e sfondo;
- **Deskewing:** rilevamento e correzione dell'inclinazione del documento, che potrebbe compromettere la segmentazione in righe e caratteri. Tra le tecniche più comuni troviamo:
 - Trasformata di Hough: rileva le linee rette nel documento, tipicamente le righe di testo, e calcola l'angolo medio di inclinazione. Adatto soprattutto con documenti ben strutturati e con righe regolari;

- PCA (Principal Component Analysis): mira a trovare la direzione dominante del contenuto nero nel documento. Inizia estraendo le coordinate di tutti i pixel neri e calcola la retta che rappresenta la loro distribuzione spaziale. L'immagine viene poi ruotata per allineare il contenuto;
- Bounding box heuristics: stima l'inclinazione del testo attraverso le bounding box di righe o parole. Il procedimento si basa sulla segmentazione delle righe e delle parole nel documento, calcolando i rettangoli minimi che contengono ogni riga/parola. Per ciascuno di essi si misura l'angolo tra il suo lato inferiore e l'asse orizzontale e infine si fa la media di tutti gli angoli calcolati. L'immagine viene poi ruotata. Il più grande limite di questo sistema è che richiede segmentazione preventiva e affidabile.

Questa fase è molto importante perchè anche un'inclinazione di pochi gradi può causare errori nella segmentazione;

- **Smoothing e Thinning:** raffinamento dei bordi dei caratteri per una migliore analisi morfologica. Lo Smoothing rende i contorni dei caratteri più regolari, eliminando piccole fluttuazioni o irregolarità. Applica dei filtri che attenuano le discontinuità nei bordi delle lettere, smussando le variazioni troppo rapide di intensità o forma. La tecnica Thinning riduce invece i tratti dei caratteri ad una linea sottile centrale - detta scheletro - per supportare il riconoscimento basato su struttura anzichè immagine. Tra le tecniche più comuni, è rilevante il Zhang-Suen Thinning Algorithm, un algoritmo iterativo per immagini binarie che, ad ogni passo, rimuove i pixel periferici senza rompere la struttura del carattere (cioè mantenendo la connettività).

Queste tecniche migliorano notevolmente la leggibilità della struttura testuale e riducono gli errori nella fase di segmentazione.

5.2.3.3 Segmentazione (Layout Analysis)

La segmentazione - o layout analysis - è una fase cruciale della pipeline OCR, in quanto permette di scomporre l'immagine del documento nei suoi costituenti logici e testuali. Essa rappresenta un passaggio di pre-elaborazione essenziale, il cui obiettivo principale è identificare le regioni contenenti testo (caratteri, parole, righe) e distinguerle da elementi non testuali come figure, tabelle, grafici o aree di margine.

Un'accurata segmentazione della pagina incide direttamente sulla precisione complessiva del sistema OCR, poiché consente di isolare blocchi omogenei di informazione e preservare la struttura del layout originale, inclusi l'ordine di lettura e l'allineamento. Inoltre, consente l'elaborazione frammentata del documento, evitando l'analisi completa di intere pagine quando non necessario.

Gli algoritmi di segmentazione possono essere classificati in tre principali categorie:

- **Approcci Top-down:** segmentano ricorsivamente il documento in regioni più piccole, sfruttando le aree vuote (es. spazi bianchi) come delimitatori. Un esempio rappresentativo è l'algoritmo XY Cut, basato su struttura ad albero: la radice rappresenta l'intera pagina, mentre le foglie corrispondono ai blocchi segmentati. Ad ogni passo, vengono calcolati i profili di proiezione (orizzontale e verticale) per identificare i punti di separazione, utilizzando soglie per la rimozione del rumore. Il processo si arresta quando non è più possibile dividere ulteriormente le regioni;
- **Approcci Bottom-up:** partono da componenti connessi di pixel, che vengono aggregati progressivamente in strutture testuali via via più complesse: da caratteri a parole, poi righe e infine blocchi. Questo approccio è più adattabile a layout non strutturati o irregolari, sebbene possa essere più sensibile al rumore e alla distorsione dell'immagine;
- **Approcci Ibridi:** combinano strategie top-down e bottom-up, cercando di trarre vantaggio da entrambe le filosofie per ottenere maggiore robustezza e flessibilità nell'analisi di layout complessi o multicolonna.

Una volta segmentato il layout della pagina in regioni strutturalmente omogenee, si procede alla **segmentazione testuale fine**, che ha l'obiettivo di individuare le linee di testo, le parole e infine i singoli caratteri. La linea di testo è definita come una sequenza di parole allineate orizzontalmente, secondo un orientamento coerente. Le tecniche tradizionali si basano su istogrammi di proiezione, analizzando le variazioni di densità lungo l'asse verticale per identificare le interlinee. Tuttavia, queste metodologie possono risultare inadeguate in presenza di layout complessi, documenti manoscritti o degradati. In tali contesti, approcci moderni basati su reti neurali si sono dimostrati più efficaci. Una volta identificate le righe, il processo continua con la segmentazione in parole e, successivamente, in caratteri:

- **Segmentazione in parole:** avviene mediante l'analisi delle distanze spaziali tra i gruppi di caratteri. Spazi più ampi tra componenti connessi vengono interpretati come separatori tra parole;
- **Segmentazione in caratteri:** può essere realizzata tramite diverse tecniche, tra cui:
 - Bounding box applicati a ciascun componente connesso;
 - Contour detection per identificare i margini dei caratteri;
 - Approcci neurali end-to-end capaci di apprendere direttamente dai dati la segmentazione contestuale.

5.2.3.4 Estrazione delle caratteristiche (Feature Extraction)

Nella fase di estrazione delle caratteristiche, l'immagine di input - tipicamente in formato binario o raster - viene trasformata in una rappresentazione numerica vettoriale. Questo passaggio intermedio è indispensabile per l'attività di classificazione, poiché fornisce un insieme di attributi discriminanti che descrivono le proprietà visive e strutturali dei caratteri.

Le feature estratte possono essere raggruppate in diverse categorie, a seconda del tipo di informazione che codificano:

- **Morfologiche:** identificano la struttura del carattere. Per individuare margini e angoli si utilizzano tecniche di rilevamento dei contorni, facilitando la distinzione tra caratteri visivamente simili (es. differenza tra “a” e “o”, basata sulla presenza o meno di anelli chiusi);
- **Statistiche:** analizzano la distribuzione dei pixel all'interno dell'immagine del carattere. Comprendono densità pixelistica, centroide e istogrammi di intensità che aiutano a catturare la forma indipendentemente da traslazione o rotazione;
- **Geometriche:** includono attributi come altezza, larghezza e relazioni spaziali tra parti del carattere, utili per disambiguare simboli che condividono strutture simili.

Esistono diversi approcci all'estrazione e classificazione delle caratteristiche:

1. **Template Matching:** consiste nella comparazione diretta dell'immagine del carattere con una serie di modelli predefiniti. È efficace su testi stampati ad alta qualità, ma risulta fragile in presenza di variazioni di font, rumore o deformazioni;
2. **Machine Learning tradizionale:** algoritmi come Support Vector Machine (SVM), k-Nearest Neighbors (kNN) e Decision Tree utilizzano le feature estratte manualmente come input per la classificazione. Sebbene meno flessibili rispetto al Deep Learning, questi approcci offrono buone prestazioni su dataset puliti o alfabeti semplici;
3. **Deep Learning con reti convoluzionali (CNN):** le CNN apprendono automaticamente rappresentazioni gerarchiche dei caratteri. I primi strati convoluzionali identificano pattern locali (es. bordi), mentre gli strati più profondi apprendono composizioni più astratte e invarianti.

5.2.3.5 Post-elaborazione (Postprocessing)

La fase di post-elaborazione interviene sull'output grezzo prodotto dal classificatore OCR, con l'obiettivo di correggere gli errori residui e migliorare la

leggibilità e l'utilizzabilità del testo risultante. Questi errori possono essere causati da molteplici fattori, per esempio:

- **Ambiguità visiva:** caratteri visivamente simili, come “0” (zero) e “O” (lettera maiuscola), oppure “I” e “l” (i maiuscola e elle minuscola), sono spesso confusi in fase di classificazione;
- **Caratteri o parole fuori vocabolario:** l'assenza di termini rari o specialistici nei modelli linguistici o dizionari può compromettere la correzione automatica;
- **Layout non standard o rumore residuo:** documenti con formattazioni atipiche, testi curvi o distorsioni introdotte dalla digitalizzazione possono generare errori sintattici o semantici.

Per mitigare tali problemi, vengono impiegate varie tecniche di postprocessing:

- **Spell checking:** controllo ortografico basato su dizionari lessicali, che confronta le parole estratte con liste di vocaboli noti, suggerendo sostituzioni in caso di mismatch;
- **Modelli linguistici:** l'utilizzo di language models, come *modelli a n-grammi*_G o *catene di Markov*_G, consente di valutare la probabilità di sequenze di parole e correggere errori contestuali;
- **Parsing grammaticale e semantico:** sistemi OCR avanzati integrano strumenti di analisi grammaticale per rilevare incongruenze sintattiche e correggere errori basandosi sulla struttura linguistica della frase.

Il risultato finale di questa fase è un testo digitale coerente, ricercabile e modificabile, che può essere esportato in formati strutturati.

5.2.4 Punti di forza e criticità

Nonostante i notevoli progressi tecnologici che mettono in luce i punti di forza dei sistemi OCR tradizionali, essi presentano anche una serie di limitazioni strutturali e operative che ne condizionano l'efficacia in contesti complessi o reali.

5.2.4.1 Punti di forza

Elevata accuratezza in condizioni controllate

In ambienti standardizzati, con documenti stampati e ben scannerizzati, i motori OCR moderni raggiungono livelli di accuratezza molto elevati.

Efficienza e velocità di elaborazione

Gli OCR tradizionali presentano latenze estremamente contenute, con capacità di elaborazione in tempo reale. Questa caratteristica è particolarmente vantaggiosa nei flussi di digitalizzazione massiva, come nell'archiviazione automatica di documenti. L'efficienza è ulteriormente incrementata da ottimizzazioni hardware (GPU) e parallelizzazione dei processi.

Supporto multilingua e multialfabeto

I motori OCR più avanzati supportano decine di lingue e alfabeti, incluso il riconoscimento di caratteri speciali e simboli tecnici. Questo rende la tecnologia particolarmente utile in contesti globalizzati o multilingue.

Integrazione consolidata nei workflow documentali

Le tecnologie OCR sono ampiamente integrate in ambienti enterprise, software di produttività e sistemi di gestione documentale. La standardizzazione di formati output ne ha facilitato la diffusione nei processi aziendali e istituzionali.

5.2.4.2 Criticità e limitazioni

Sensibilità alla qualità dell'immagine

L'OCR classico è fortemente dipendente dalla qualità dell'immagine in input. La presenza di elementi come sfocature da movimento o fuori fuoco e basso contrasto o rumore (es. macchie, pieghe, texture del foglio) può compromettere seriamente la segmentazione e la classificazione. In particolare, l'inclinazione del documento e le distorsioni prospettiche rendono più difficile la binarizzazione e l'allineamento dei caratteri.

Limitata flessibilità ai layout complessi

I sistemi OCR convenzionali non sono progettati per interpretare la semantica strutturale di un documento. Di conseguenza, presentano difficoltà in presenza di tabelle non grigliate, layout multi-colonna, moduli con campi da compilare, documenti misti con grafici, firme e timbri. In questi casi, l'OCR restituisce spesso una sequenza lineare di testo, perdendo relazioni spaziali fondamentali tra blocchi.

Scarso supporto per font irregolari e scrittura a mano

Sistemi OCR basati su template o su modelli statici soffrono di un'elevata rigidità:

- Non riconoscono correttamente font poco comuni;
- Non gestiscono bene caratteri decorativi o stilizzati;
- Hanno performance molto basse su testi manoscritti, a meno di addestramento specifico.

Sebbene alcuni OCR integrino CNN, la scrittura a mano richiede soluzioni specialistiche per decodificare la variabilità nei tratti e nelle interruzioni.

Mancanza di comprensione semantica

L'OCR classico esegue un'estrazione "cieca" del testo: non distingue entità come nomi propri, importi o date, né comprende la relazione tra elementi.

Inoltre:

- Non interpreta abbreviazioni, sigle o formule;
- Non esegue validazioni di coerenza logica;
- Non effettua classificazione del contenuto (es. intestazione vs corpo).

Dipendenza da linguaggio e dizionario

Molti motori OCR integrano dizionari linguistici per correggere ambiguità, ma questi strumenti hanno limiti in quanto sono spesso monolingua, non riconoscono neologismi, nomi propri o termini tecnici e generano falsi positivi se il contesto linguistico è incoerente. Questo causa problemi in ambiti specialistici (giuridico, medico, tecnico), dove il vocabolario è altamente specifico.

Scalabilità e mancanza di apprendimento adattivo

A differenza dei moderni sistemi AI-based, l'OCR tradizionale:

- Non impara dai propri errori;
- Non si adatta a nuovi tipi di documenti senza riconfigurazione manuale;
- Richiede ottimizzazioni per ogni layout/documento diverso.

In ambienti dinamici (es. acquisizione da mobile, documenti generati da diversi fornitori), questo comporta un'elevata manutenzione operativa.

5.3 Tecniche di Intelligenza Artificiale per l'estrazione di dati

5.3.1 Definizione e contesto

L'Intelligenza Artificiale (AI) è l'insieme di tecnologie e metodi che permettono a un sistema informatico di emulare capacità tipiche dell'intelligenza umana, come il ragionamento, l'apprendimento, la percezione e l'interazione linguistica. In ambito documentale, l'AI viene utilizzata per automatizzare la comprensione e l'analisi di testi, immagini e strutture complesse contenute in documenti digitali o scannerizzati.

Una delle sue applicazioni più rilevanti è la Document Intelligence, ovvero la capacità di comprendere, strutturare ed estrarre informazioni utili da documenti eterogenei, spesso in formati non strutturati (come PDF, immagini scan-

nerizzate, moduli compilati a mano, ecc.). Questo approccio integra diverse discipline:

- Computer Vision, per riconoscere e interpretare il layout grafico del documento (intestazioni, tabelle, colonne);
- Natural Language Processing (NLP), per comprendere il contenuto linguistico e identificare entità rilevanti;
- Machine Learning e Deep Learning, per apprendere automaticamente modelli di estrazione e generalizzare su documenti nuovi.

Rispetto all'OCR tradizionale, che si limita a trascrivere testo da immagini in formato digitale, l'AI consente di:

- Comprendere il contesto semantico del testo;
- Estrarre dati strutturati anche da layout complessi;
- Identificare relazioni logiche tra le informazioni (es. campo - valore).

L'uso dell'AI nell'estrazione di dati da documenti rappresenta un'evoluzione significativa rispetto alle tecniche OCR classiche, poiché introduce una comprensione semantica e strutturale del contenuto, rendendo possibile una vera e propria “lettura intelligente” dei documenti.

5.3.2 Evoluzione storica

L'evoluzione dell'Intelligenza Artificiale applicata all'estrazione di dati da documenti si intreccia con la storia del Natural Language Processing (NLP), della visione artificiale e dell'apprendimento automatico. Se nei primi decenni l'estrazione di dati avveniva tramite sistemi a regole, oggi viene gestita da architetture neurali avanzate in grado di comprendere il linguaggio, la struttura e la semantica dei documenti.

Anni '60-'90: sistemi a regole e approcci statici

I primi sistemi di estrazione automatica di informazioni nascono negli anni '60-'70 in ambito accademico, con l'obiettivo di analizzare testi scientifici o legali. Questi erano basati su pattern regolari (espressioni regolari, regole sintattiche) e necessitavano di una progettazione manuale per ogni tipo di documento. L'efficacia era limitata a formati rigidi e linguaggi prevedibili.

Negli anni '90, la crescente disponibilità di testi digitali e l'introduzione del Machine Learning supervisionato portano all'uso di classificatori statistici per riconoscere entità in testi non strutturati. Tuttavia, questi metodi erano ancora deboli nel catturare il contesto linguistico e incapaci di gestire layout visivi.

Anni 2000-2010: NLP statistico e modelli sequenziali

Con la diffusione di internet e dei corpus testuali su larga scala, l'NLP diventa più sofisticato. Si affermano modelli sequenziali capaci di etichettare entità in sequenze di testo in modo probabilistico, utilizzati nel contesto del Named Entity Recognition (NER).

A partire dal 2008, le reti neurali ricorrenti (RNN) e le loro varianti iniziano ad essere utilizzate per modellare la dipendenza contestuale nel testo, migliorando notevolmente le prestazioni nel riconoscimento di entità e segmenti informativi.

Dal 2018: l'era dei Transformer

Il vero punto di svolta si ha nel 2018 con la pubblicazione di BERT (Bidirectional Encoder Representations from Transformers) da parte di Google. Questo modello ha introdotto un nuovo paradigma di rappresentazione linguistica: ogni parola viene rappresentata in relazione a tutte le altre nella frase, grazie al meccanismo di *self-attention*. BERT ha portato miglioramenti nei task di NER, classificazione del testo e question answering.

A seguito di BERT, altri modelli transformer-based sono stati adattati specificamente all'estrazione documentale:

- LayoutLM (Microsoft, 2020): combina testo, posizione e immagine per analizzare documenti con layout complessi;
- TAPAS (Google, 2020): progettato per comprendere tabelle e rispondere a domande sui dati contenuti nelle celle;
- LayoutLMv2 e v3: estensioni che integrano visione artificiale (CNN) e embedding posizionali avanzati, aumentando la precisione su documenti non lineari.

Dal 2020 a oggi: AI generativa e multimodalità

Con l'avvento dei modelli GPT-3 e GPT-4 di OpenAI, basati su decine di miliardi di parametri, l'estrazione di informazioni da documenti non si concentra più solo sul riconoscimento, ma anche sul ragionamento, inferenza e sintesi automatica.

Parallelamente, le tecniche di Vision-Language Modeling permettono a modelli neurali di operare direttamente su documenti PDF o scansionati, senza separare testo e immagine, in una pipeline completamente end-to-end.

Oggi, le tecnologie AI sono in grado di:

- Interpretare documenti destrutturati o rumorosi;
- Estrarre e classificare tabelle, grafici, diagrammi;
- Comprendere documenti in linguaggio naturale e rispondere a query complesse.

5.3.3 Architettura e funzionamento

Un sistema sofisticato di Document AI integra più moduli specializzati, superando la semplice trascrizione testuale dell'OCR tradizionale e introducendo anche l'interpretazione strutturale, semantica e relazionale dei documenti.



Figura 5.2: AI pipeline.

5.3.3.1 Acquisizione e pre-processing del documento

Il processo inizia con la digitalizzazione del documento, che può essere un'immagine raster o un PDF nativo. Il pre-processing, se necessario, include:

- **Pulizia e binarizzazione**, utile quando il documento è un'immagine;
- **OCR integrato** per estrarre il testo;
- **Parsing strutturale**, con ricavo di bounding box e layout visivo.

5.3.3.2 Parsing visivo e analisi del layout

Il parsing visivo è una fase fondamentale nei sistemi AI di estrazione dei dati, poiché consente di analizzare non solo il contenuto testuale, ma anche la struttura spaziale del documento. L'obiettivo è identificare e segmentare blocchi logici come titoli, paragrafi, sezioni, tabelle o grafici, che assumono significato solo se considerati nel contesto del loro layout.

La segmentazione del layout può essere affrontata con metodi di visione artificiale classici, trattati nella Sezione 5.2.3.3, ma anche con approcci neurali che hanno affiancato - e a volte sostituito - i metodi tradizionali.

Una volta segmentate le regioni visive, ciascun token testuale viene associato al proprio bounding box (es. coordinate x_0, y_0, x_1, y_1), espresso rispetto alla dimensione della pagina. Queste coordinate non sono solo *metadati_G*, ma vengono trasformate in embedding posizionali a due dimensioni, che permettono al modello di apprendere relazioni spaziali tra le parole. Questo approccio è particolarmente utile in presenza di documenti multi-colonna o layout non lineari.

Tali embedding vengono concatenati o sommati agli embedding testuali per costruire un input multimodale.

L'inclusione della struttura visiva è ciò che distingue i modelli OCR tradizionali da quelli Document AI. Senza il parsing del layout, un modello linguistico rischia di “leggere” il contenuto in ordine errato, specialmente nei casi di tabelle o documenti con layout complessi. L'analisi visiva fornisce una mappa strutturata su cui i moduli di estrazione possono operare in modo preciso e robusto.

5.3.3.3 Tokenizzazione ed embedding

La fase di tokenizzazione ed embedding ha l'obiettivo di convertire il testo estratto in una rappresentazione numerica adatta al processamento da parte dei modelli di apprendimento automatico. Questa fase consiste in due passaggi principali: la suddivisione del testo in unità minime (token) e la trasformazione di ciascun token in un vettore numerico (embedding) che ne catturi le proprietà semantiche e sintattiche.

5.3.3.3.1 Tokenizzazione

La tokenizzazione consiste nel suddividere una stringa testuale in elementi significativi, che possono essere parole, sottoparole o caratteri, a seconda del modello adottato:

- **Word-level:** ogni parola è considerata un token; questo modello però è sensibile a vocaboli rari o fuori dizionario;
- **Subword-level:** le parole sono divise in unità più piccole, permettendo una rappresentazione robusta anche per neologismi e parole composte;
- **Character-level:** meno usata in contesti documentali, utile per lingue agglutinanti o rumore tipografico.

5.3.3.3.2 Embedding statici

Gli embedding statici sono rappresentazioni numeriche associate a ciascuna parola del vocabolario, generate indipendentemente dal contesto in cui la parola appare. Quindi ogni parola possiede un singolo vettore fisso che ne descrive il

significato generale, valido per tutte le sue occorrenze nel documento. I modelli di embedding statico mappano ciascuna parola in un vettore, in uno spazio continuo di dimensione ridotta (tipicamente 100-300 dimensioni), tale da preservare relazioni semantiche - ad esempio,

$$\text{vec}(\text{"re"}) - \text{vec}(\text{"uomo"}) + \text{vec}(\text{"donna"}) \approx \text{vec}(\text{"regina"}).$$

Tuttavia, questi metodi presentano un limite significativo: la rappresentazione di una parola rimane invariata a prescindere dal contesto in cui appare. Ciò significa che parole con più significati saranno sempre rappresentate dallo stesso vettore, rendendo difficile la disambiguazione semantica in compiti come l'estrazione di informazioni da documenti reali, dove il contesto può modificare profondamente il significato. Per risolvere queste ambiguità, sono stati introdotti i modelli di embedding contestuali, che producono una rappresentazione diversa per ciascuna occorrenza della parola, a seconda del suo uso nella frase.

5.3.3.3 Embedding contestuali

I modelli di embedding contestuali producono rappresentazioni che variano in base alla frase in cui il termine compare. Questa proprietà è fondamentale per catturare sfumature semantiche e ambiguità lessicali tipiche del linguaggio naturale. I modelli basati su architettura Transformer, in particolare, hanno reso possibile questa rappresentazione dinamica grazie al meccanismo di self-attention, che permette a ciascun token di “osservare” tutti gli altri token della sequenza durante la codifica.

I principali modelli in questo ambito sono:

- **BERT** (Bidirectional Encoder Representations from Transformers): genera embedding bidirezionali in cui il significato di ciascun token riflette sia il contesto precedente che quello seguente;
- **RoBERTa**: un'estensione di BERT con miglioramenti nel processo di pre-training, tra cui l'uso di un corpus più ampio e training più lungo.

Questi modelli sono utilizzati come encoder generici del testo, ma possono essere facilmente adattati a task specifici, come il riconoscimento di entità

(NER), la classificazione, o l'estrazione di relazioni. In scenari documentali, offrono un vantaggio significativo rispetto agli approcci statici, poiché possono modellare le dipendenze semantiche anche in testi complessi o poco strutturati.

5.3.3.3.4 Embedding multimodali

Nel contesto dell'estrazione di dati da documenti è fondamentale anche codificare la struttura visiva e spaziale del documento, che veicola spesso informazioni cruciali (es. ordine di lettura, relazioni tabellari, gerarchie visive). Per rispondere a questa esigenza, sono stati sviluppati modelli di embedding multimodale, come la famiglia LayoutLM, progettati per combinare testo, posizione e immagine.

- **LayoutLM v1:** estende l'architettura BERT aggiungendo, oltre agli embedding testuali, anche embedding posizionali derivati dalle coordinate dei bounding box di ciascun token. In questo modo, il modello può apprendere relazioni spaziali 2D tra elementi testuali sulla pagina;
- **LayoutLMv2:** introduce la componente visiva, integrando feature estratte direttamente dall'immagine del documento mediante una rete convoluzionale. L'attenzione self-attentive del modello è quindi applicata su tre modalità: testo, posizione e visione;
- **LayoutLMv3:** sostituisce la CNN con un Vision Transformer (ViT), che consente una rappresentazione più ricca e astratta dell'immagine. Inoltre, adotta una strategia di pre-training unificata che combina tre obiettivi:
 - Masked Language Modeling (MLM) - predizione di token nascosti nel testo;
 - Masked Image Modeling (MIM) - predizione di regioni visive oscurate;
 - Word-Patch Alignment - apprendimento della corrispondenza tra parole e regioni visive.

Questi modelli forniscono una rappresentazione multimodale e strutturata dei documenti, dove ogni token è rappresentato non solo in base al suo significato linguistico, ma anche in base alla sua posizione sulla pagina e al contesto visivo circostante.

5.3.3.4 Named Entity Recognition (NER)

Il Named Entity Recognition (NER) rappresenta la fase in cui il testo viene analizzato per individuare e classificare segmenti che corrispondono a entità semantiche rilevanti. Queste entità possono appartenere a categorie generiche (es. nomi propri, date, località) oppure a tipologie specifiche dipendenti dal dominio applicativo (es. codice fiscale, partita IVA, codici catastali). Il NER riceve in input gli embedding dei token e assegna a ciascuno un'etichetta che indica la sua appartenenza a una determinata entità. L'assegnazione delle etichette avviene tramite un modulo di classificazione sequenziale, che riceve in input gli embedding dei token e, per ciascuno di essi, calcola la probabilità di appartenenza a una determinata classe semantica. Questo viene realizzato tipicamente tramite un layer lineare seguito da una funzione che seleziona l'etichetta con la massima probabilità.

5.3.3.5 Relationship extraction e modellazione semantica

Una volta riconosciute le entità nel testo, il sistema procede con la fase di estrazione delle relazioni (Relationship Extraction, RE), che ha l'obiettivo di determinare se e come due entità sono logicamente collegate all'interno del documento. Questa fase è fondamentale per trasformare una lista di entità isolate in una rappresentazione strutturata della conoscenza.

Le tecniche principali includono:

- **Dependency Parsing:** viene utilizzato per estrarre relazioni sintattiche all'interno della frase. Il parsing dipendenziale genera un albero in cui i nodi sono parole e gli archi rappresentano relazioni grammaticali. Relazioni semantiche semplici possono essere dedotte direttamente da queste connessioni;

- **Modelli Transformer-based per entità-pair:** in contesti più complessi, si utilizzano modelli Transformer fine-tuned per la classificazione binaria su coppie di entità. L'input è costituito dalla sequenza contenente le due entità marcate, e l'output è la predizione della presenza o meno di una relazione, con relativa tipologia. Questo approccio è particolarmente utile per documenti con frasi lunghe o strutture deboli;
- **Graph Neural Networks (GNN):** quando le entità e le frasi sono distribuite su più sezioni o pagine, viene costruito un grafo semantico, in cui:
 - I nodi rappresentano le entità estratte;
 - Gli archi modellano potenziali relazioni (basate su co-occorrenza, distanza testuale, struttura del documento, ecc.);

Le GNN apprendono funzioni di aggiornamento dei nodi e di propagazione delle informazioni per classificare i tipi di relazione anche in contesti di lungo raggio. Questo approccio è efficace in scenari document-level, dove la relazione tra due entità non si trova nella stessa frase, ma emerge dalla struttura complessiva del documento.

L'estrazione delle relazioni costituisce la base per la costruzione di knowledge graph documentali, che possono essere interrogati o analizzati automaticamente per generare insight strutturati.

5.3.3.6 Output strutturato

Al termine della pipeline, le informazioni estratte vengono organizzate in un formato strutturato, che rende il contenuto del documento interpretabile automaticamente. Ogni entità riconosciuta è rappresentata come un oggetto dotato di attributi:

- **Tipo semantico:** la categoria dell'entità;
- **Valore:** il testo originale o normalizzato estratto dal documento;

- **Coordinate di posizione:** bounding box o posizione testuale, utile per il riferimento visivo o per ricostruire il layout;
- **Relazioni:** collegamenti con altre entità riconosciute, rappresentate in forma relazionale o grafo.

Questa fase rappresenta il passaggio dalla comprensione testuale alla fruizione computazionale dell'informazione documentale, chiudendo il ciclo della pipeline di Document AI.

5.3.4 Punti di forza e criticità

Le tecniche di Intelligenza Artificiale hanno rivoluzionato l'estrazione automatica di informazioni da documenti, superando molte delle limitazioni dell'OCR tradizionale. Tuttavia, anche questi approcci presentano sfide specifiche. Di seguito si analizzano i principali punti di forza e le principali criticità delle soluzioni AI-based.

5.3.4.1 Punti di forza

Comprensione semantica

A differenza dei sistemi OCR classici, i modelli AI (es. BERT, LayoutLM, GPT) non si limitano a riconoscere il testo, ma comprendono il significato e il contesto delle parole. Ciò consente:

- Il riconoscimento di entità;
- L'identificazione automatica di relazioni logiche tra entità.

Robustezza ai layout complessi

Grazie alla combinazione di componenti visive (CNN, Vision Transformer) e posizionali (bounding box), i modelli come LayoutLMv2 sono in grado di interpretare documenti con:

- Strutture tabellari irregolari;

- Layout multi-colonna o con box sovrapposti;
- Documenti tecnici, fiscali o giuridici con formattazioni ibride.

Adattabilità al dominio

I modelli AI possono essere:

- Fine-tuned su documenti specifici;
- Arricchiti con nuove entità e regole semantiche;
- Potenziati con tecniche di *transfer learning*_G per ottenere buone prestazioni anche con dataset limitati.

Output strutturato e machine-readable

L'output prodotto da questi modelli è spesso in formato JSON, XML o CSV, pronto per:

- Alimentare database o sistemi ERP/CRM;
- Supportare workflow documentali automatizzati (Intelligent Document Processing);
- Abilitare l'analisi semantica su larga scala.

Continuità evolutiva e multimodalità

I modelli recenti consentono una pipeline completamente end-to-end, in grado di operare direttamente su immagini e documenti PDF senza necessità di un OCR separato. Questo riduce l'errore cumulativo e semplifica l'architettura.

5.3.4.2 Criticità e limitazioni

Dipendenza dai dati annotati

L'efficacia dei modelli supervisionati dipende fortemente dalla disponibilità di dati etichettati di qualità. In molti contesti creare dataset sufficientemente grandi è costoso e richiede competenze specialistiche.

Complessità computazionale

I modelli transformer-based sono costosi in termini di:

- Risorse hardware (GPU, RAM, spazio disco);
- Tempo di addestramento e inferenza;
- Manutenzione e aggiornamento continuo.

L'uso in tempo reale su dispositivi edge o in ambienti a bassa potenza può risultare limitato.

Sensibilità al layout degradato o rumore visivo

Sebbene le reti neurali siano più robuste rispetto all'OCR classico, persistono difficoltà con:

- Documenti mal scannerizzati o fotografati con distorsioni;
- Immagini con artefatti grafici, watermark, firme o timbri sovrapposti;
- Testi in lingua mista o con simboli speciali non presenti nel pretraining.

Interpretabilità e trasparenza

I modelli di deep learning, pur essendo altamente efficaci, sono spesso delle “black box”. Comprendere il motivo di un'estrazione errata o di un fallimento è complesso e limita l'applicabilità in settori sensibili.

Bias e generalizzazione limitata

I modelli pre-addestrati su dati generici possono contenere bias linguistici o culturali e non generalizzare bene su:

- Linguaggi specialistici;
- Lingue minori o documenti non standardizzati;
- Entità con nomi ambigui o inconsueti.

5.4 Confronto tra OCR tradizionale e AI

Accuratezza e prestazioni

L'OCR tradizionale, sebbene sia una tecnologia già ampiamente esplorata, presenta forti limiti in termini di accuratezza, specialmente in presenza di documenti:

- Con formattazione complessa (tabelle, colonne multiple);
- Degradata o scansionati male;
- Contenenti manoscritti, font rari o simboli speciali.

Al contrario, i modelli AI moderni hanno mostrato:

- Un'accuratezza maggiore nel riconoscimento di entità testuali in contesti strutturati e semi-strutturati;
- Capacità di generalizzare anche su layout non affrontati in fase di addestramento;
- Maggiore resistenza a rumore, inclinazioni e distorsioni.

Tuttavia, la precisione dell'AI è strettamente legata alla disponibilità di dati etichettati di alta qualità e a un corretto *[fine-tuning](#)*_G. Inoltre, l'inferenza richiede risorse computazionali superiori, e non sempre è possibile.

Flessibilità e adattabilità

L'OCR classico è un sistema rigido e tende a fallire fuori da tale dominio. Non comprende il contenuto, non si adatta a layout nuovi, e non può apprendere dalle correzioni.

Le tecnologie AI, invece, offrono:

- Apprendimento continuo: i modelli possono essere aggiornati con nuovi esempi;
- Adattabilità al dominio: si possono specializzare per vari scenari;

- Gestione della multimodalità: capacità di elaborare testo, immagine e struttura.

Tale flessibilità comporta anche una maggiore complessità ingegneristica: serve un'infrastruttura per addestrare, validare, monitorare e aggiornare i modelli. In ambienti con requisiti normativi o limitazioni tecniche, ciò può rappresentare un ostacolo.

5.4.1 Casi d'uso e applicazioni pratiche

OCR tradizionale

È ancora largamente utilizzato in contesti dove:

- La struttura del documento è semplice e omogenea;
- Il contenuto è stampato con font standard;
- La priorità è la leggerezza computazionale e la semplicità di integrazione;
- I documenti hanno struttura fissa.

Document Intelligence

Trova impiego in casi più complessi, dove il contenuto testuale è strettamente legato alla struttura visiva. Alcuni esempi:

- Estrazione di dati con layout non standardizzati;
- Lettura automatica di moduli compilati a mano;
- Riconoscimento di entità come codici e nomi nei documenti.

In questi contesti, la Document Intelligence permette non solo l'estrazione, ma anche la classificazione, la validazione e il popolamento di sistemi gestionali, riducendo drasticamente l'intervento umano.

5.5 Scelta adottata

Durante la fase di analisi ho valutato il codice preesistente messo a disposizione dall'azienda, relativo a un'integrazione per l'estrazione automatica di dati da documenti PDF tramite tecniche basate sull'Intelligenza Artificiale. Dopo aver esaminato e testato tale soluzione insieme ai referenti aziendali, è emerso che l'effort richiesto per l'integrazione, la personalizzazione e la manutenzione di quel sistema risultava sproporzionato rispetto alle reali necessità del progetto.

Alla luce di questa valutazione, abbiamo deciso di orientarci verso un approccio più diretto e mirato, basato sull'utilizzo di strumenti tradizionali per l'estrazione dei dati da documenti PDF con struttura nota e costante. Tale scelta è stata resa possibile dal fatto che i documenti da analizzare provengono da un portale ufficiale (AVEPA) e rispettano un formato standardizzato, rendendo superflua l'adozione di modelli intelligenti.

Questa decisione ha permesso di ottimizzare non solo le risorse di sviluppo, ma anche i tempi di esecuzione del processo di estrazione, che si è rivelato significativamente più rapido grazie alla struttura prevedibile e ripetibile dei documenti in ingresso. L'adozione di un approccio basato su template ha infatti ridotto la complessità computazionale, garantendo un'elaborazione efficiente e stabile, senza la necessità di ricorrere a modelli generici di interpretazione.

Capitolo 6

Progettazione

Questo capitolo presenta la progettazione del database, del frontend, del backend e del flusso di estrazione dati.

6.1 Progettazione database

Il passo successivo all'analisi è stato la progettazione del database. Prima di procedere con l'analisi e la scelta della tipologia di database più adatta - relazionale (SQL) o non relazionale (NoSQL) - ho realizzato un diagramma Entità/Relazioni (E/R), con l'obiettivo di ottenere una rappresentazione concettuale e ad alto livello dei dati. Tale modellazione ha consentito di tradurre le informazioni emerse dall'analisi del dominio applicativo in uno schema concettuale strutturato, evidenziando le entità rilevanti e le relazioni tra di esse. Questo passaggio si è rivelato fondamentale per garantire coerenza semantica e integrità nell'applicativo.

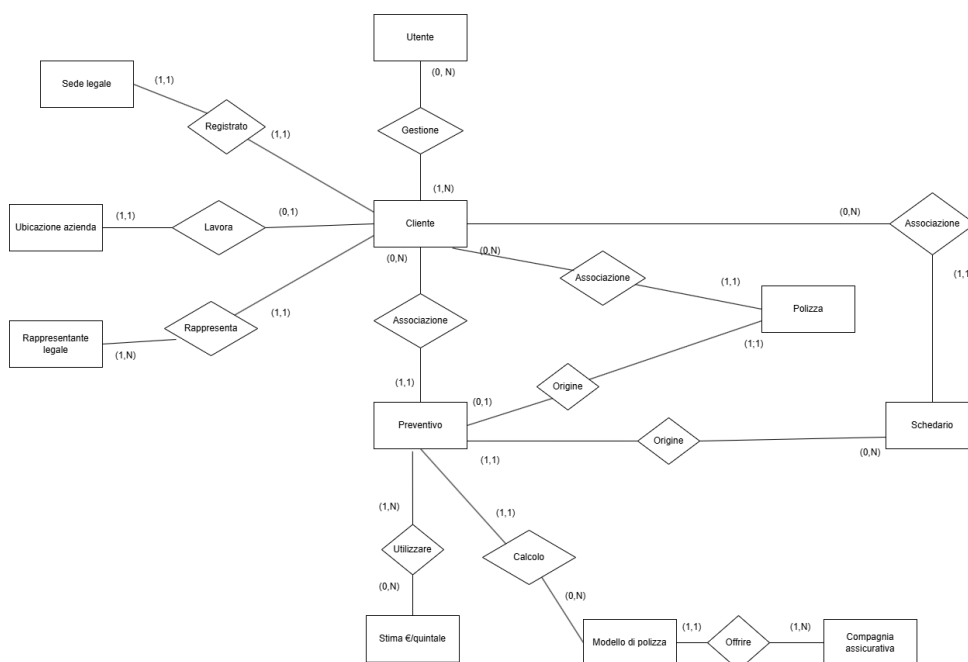


Figura 6.1: Diagramma E/R.

Dopo aver individuato le entità principali attraverso la modellazione concettuale, visibile nella Figura 6.1, ho svolto un'analisi comparativa tra due paradigmi di gestione dei dati: i database relazionali (SQL) e i database non relazionali (NoSQL), con particolare riferimento a MongoDB. La valutazione si è basata su criteri quali flessibilità dello schema, scalabilità, coerenza dei dati, performance e complessità di sviluppo.

Database relazionali (SQL)

I database relazionali, come MySQL o PostgreSQL, si basano su uno schema rigido definito a priori, in cui i dati sono organizzati in tabelle normalizzate collegate da vincoli di integrità. Questo approccio garantisce:

- **Pro:**
 - Elevata consistenza e integrità referenziale dei dati;
 - Supporto completo a transazioni ACID (Atomicità, Coerenza, Isolamento, Durabilità);
 - Linguaggio di interrogazione standardizzato (SQL).

- **Contro:**

- Rigidità dello schema, che richiede modifiche strutturali anche per semplici evoluzioni del modello;
- Complessità nella modellazione di strutture dati gerarchiche o annidate.

Database non relazionali (NoSQL)

I database NoSQL, e in particolare MongoDB, utilizzano una struttura a documenti (in formato *JSON*) che consente una rappresentazione flessibile e dinamica dei dati. I principali vantaggi includono:

- **Pro:**

- Schema flessibile e modificabile in fase di esecuzione;
- Gestione nativa di strutture annidate e documenti complessi;
- Maggiore rapidità nelle operazioni CRUD per documenti di grandi dimensioni.

- **Contro:**

- Assenza nativa di vincoli relazionali e di integrità referenziale;
- Richiede maggiore attenzione nella progettazione per evitare ridondanze e incoerenze.

Scelta progettuale

Alla luce delle esigenze specifiche del progetto - in particolare la presenza di dati suscettibili a modifiche nel tempo - è stato scelto di adottare un database non relazionale, nello specifico MongoDB. Tale scelta è stata motivata dalla necessità di garantire:

- Una maggiore flessibilità nella definizione e nell'evoluzione dello schema dei dati;
- Una gestione più agevole di documenti annidati e relazioni deboli;

- Una migliore scalabilità in previsione di un'evoluzione futura del sistema.

Questa decisione ha consentito di ridurre la complessità nella serializzazione e deserializzazione dei dati tra frontend e backend, favorendo un'integrazione più diretta e coerente con l'architettura basata su TypeScript.

A fronte di questa scelta, le *collections_G* presenti all'interno del database - e le relazioni tra di esse - sono rappresentate nella Figura 6.2:

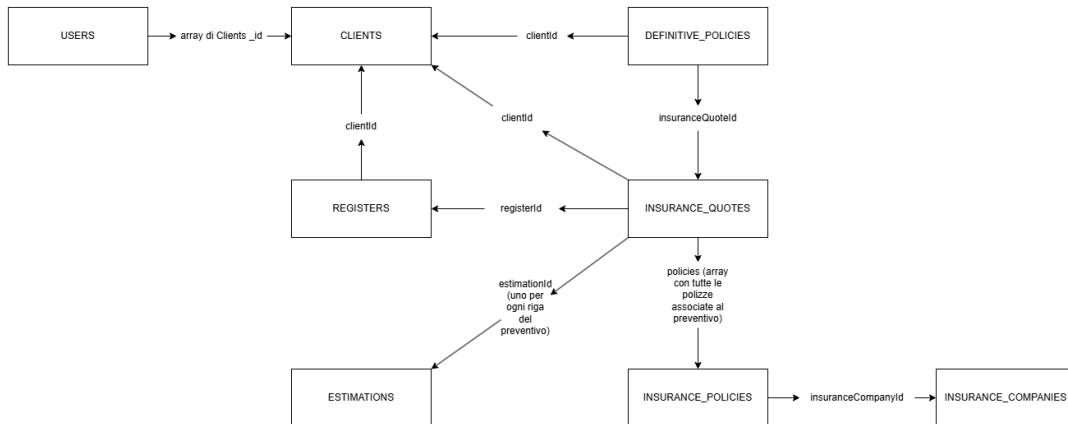


Figura 6.2: Collections MongoDB.

6.2 Progettazione frontend

Nel progettare l'architettura del frontend, ho adottato un approccio component-based, sfruttando la struttura modulare offerta da React. Ciascuna vista dell'applicativo è stata modellata come una pagina indipendente, a sua volta suddivisa in componenti funzionali riutilizzabili.

Ogni sezione dell'interfaccia riflette una specifica area funzionale, coerente con le epiche individuate in fase di analisi (gestione clienti, compagnie, stime, polizze).

Per mantenere coerenza e scalabilità, ho strutturato il progetto secondo una suddivisione logica dei moduli, distinguendo tra:

- Pagine: viste principali del sistema, corrispondenti alle entità o ai flussi chiave;
- Componenti: elementi riutilizzabili dell'interfaccia (es. tabelle, form, modali);

- Moduli di servizio: gestione della comunicazione con il backend;
- *Hook_G* personalizzati: astrazioni logiche per l'interazione con i dati.

Ogni chiamata al backend segue uno schema consolidato, che prevede il passaggio dei dati attraverso una catena logica composta da: **pagina - hook personalizzato - funzione di servizio (fetch)**, come mostrato nella Figura 6.3.



Figura 6.3: Architettura frontend.

La Figura 6.4 mostra lo schema delle relazioni tra le pagine principali dell'applicativo.



Figura 6.4: Schema delle relazioni tra le pagine del frontend.

6.3 Progettazione backend

La progettazione del backend è stata improntata a principi di modularità, chiarezza e separazione delle responsabilità. Ho adottato un'architettura a livelli, in cui ciascun componente ha un ruolo ben definito nel ciclo di gestione delle richieste HTTP.

Il flusso generale seguito per ogni endpoint è strutturato nel seguente modo:

- Rotta: definisce il percorso HTTP esposto dal server e associa la richiesta al relativo controller;

- **Controller:** contiene la logica applicativa associata alla rotta. Si occupa di coordinare le operazioni richieste e restituire una risposta al client. Il controller non accede direttamente al database, ma delega la gestione dei dati al livello successivo;
- **Repository:** rappresenta il livello di accesso ai dati. Qui vengono definite le operazioni CRUD (Create, Read, Update, Delete) sulle collezioni MongoDB. Questo livello isola l'accesso al database e favorisce la riusabilità del codice.

Questa suddivisione ha permesso di mantenere il codice backend organizzato, facilmente estendibile e testabile. Inoltre, ha reso possibile una gestione chiara della logica applicativa, semplificando l'integrazione con il frontend e migliorando la manutenibilità generale del progetto.

6.4 Progettazione estrazione dati

L'estrazione dei dati rappresenta una delle componenti centrali del progetto. Essa consente di analizzare i documenti PDF caricati dagli utenti e di trasformarli in strutture dati utili per l'applicativo. La progettazione di questo flusso ha richiesto una particolare attenzione alla modularità e alla separazione tra logica applicativa e logica di elaborazione. I file caricati possono essere di due tipologie:

- **Stime:** file PDF contenenti le stime economiche in euro per quintale, suddivise per varietà viticola;
- **Schedario:** file PDF, generato dal portale AVEPA e contenente:
 - **Quadro A:** le informazioni anagrafiche del cliente;
 - **Quadro B:** le informazioni delle coltivazioni, organizzate in formato tabellare.

Flusso generale di estrazione

La Figura 6.5 illustra il flusso generale seguito per l'estrazione dei dati a partire da una richiesta proveniente dal frontend.

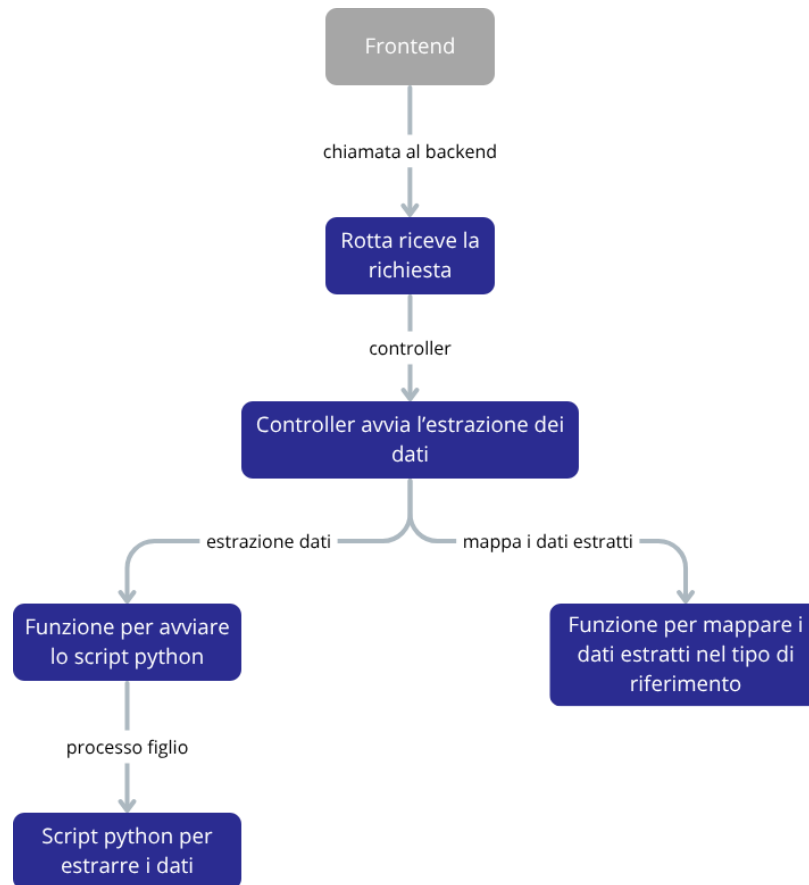


Figura 6.5: Schema generale del flusso di estrazione dei dati.

Il processo si articola nei seguenti passaggi:

1. Il frontend invia una richiesta HTTP al backend, includendo il file PDF da elaborare e i metadati associati;
2. La rotta backend riceve la richiesta e la inoltra al controller corrispondente;
3. Il controller si occupa di avviare il flusso di estrazione, coordinando due operazioni distinte:
 - L'estrazione dei dati tramite l'esecuzione di uno script Python. Questo script utilizza strumenti specializzati per analizzare la struttura del PDF e ricavare i dati grezzi;

- La mappatura dei dati estratti in un formato coerente con i tipi previsti all'interno dell'applicativo, tramite una funzione dedicata.

Ho utilizzato questo flusso per l'estrazione di dati dai file di stime e per l'inserimento di un nuovo cliente nel database a partire da uno schedario. In quest'ultimo caso ho estratto le informazioni del cliente e poi proseguito, sempre con lo stesso flusso, ad estrarre in automatico anche le informazioni tabellari presenti nello schedario caricato.

Questa architettura ha permesso di mantenere il backend leggibile e manutenibile, delegando l'analisi del documento a strumenti più adatti alla manipolazione di PDF.

Gestione della duplicazione dati e confronto con il database

Nel caso dell'upload di uno schedario per aggiungere un nuovo Quadro B (parte tabellare), è previsto un passaggio intermedio di confronto tra i dati estratti dal Quadro A (anagrafica cliente) e quelli già presenti nel database. Questo confronto consente di evitare duplicazioni e di lasciare all'utente la scelta se aggiornare o meno i dati esistenti.

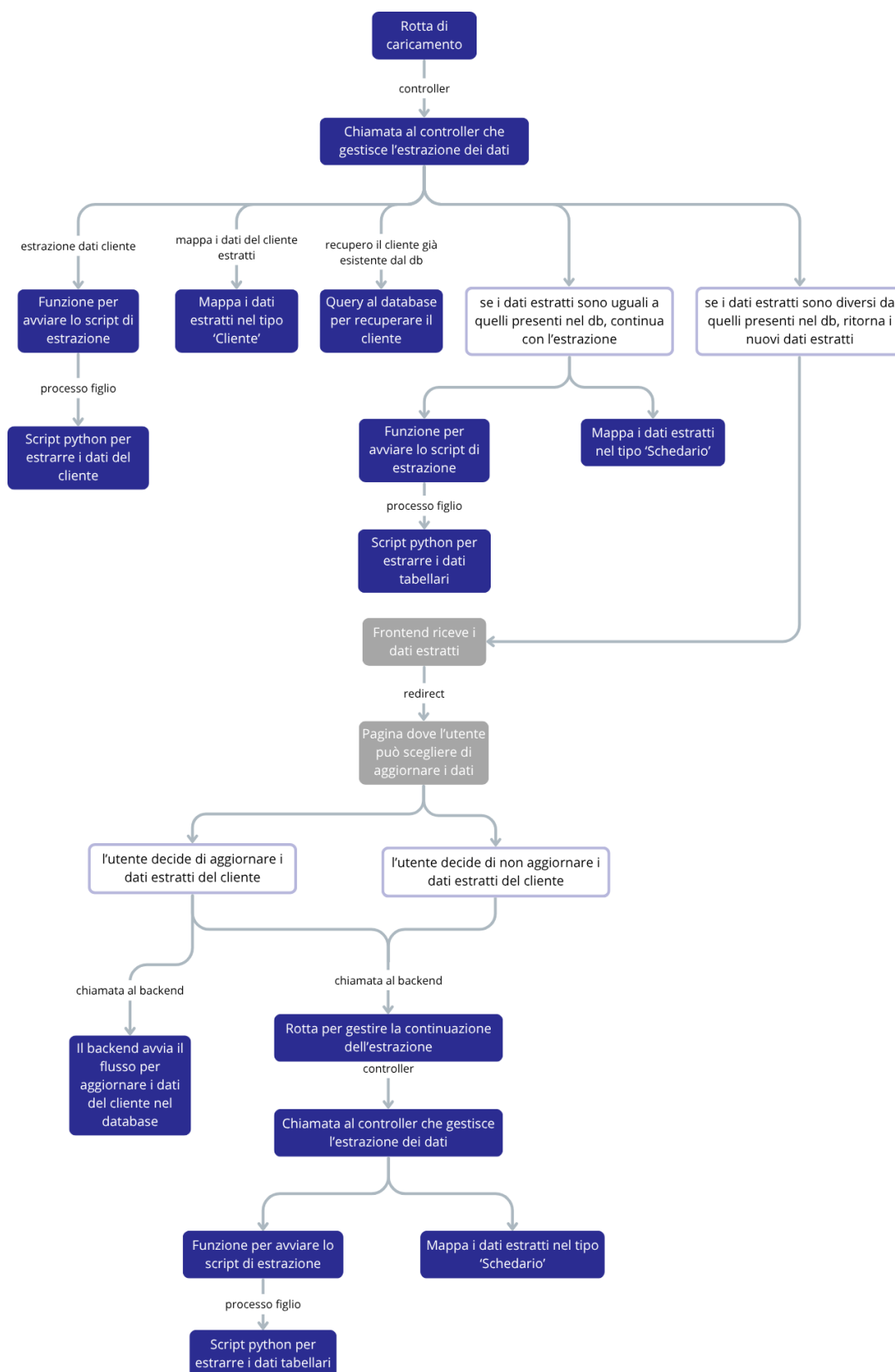


Figura 6.6: Schema del flusso di estrazione con confronto dei dati cliente.

Come mostrato nella Figura 6.6, il sistema procede come segue:

- I dati anagrafici vengono estratti e mappati nel tipo Cliente;
- Il backend interroga il database per verificare se il cliente esiste già;
- Se i dati coincidono, l'estrazione prosegue direttamente con il Quadro B;
- Se i dati differiscono, il sistema restituisce al frontend le informazioni per consentire all'utente di decidere se aggiornare o meno i dati presenti;
- In caso di aggiornamento, il backend aggiorna i dati e prosegue con l'estrazione;
- In caso contrario, i dati vengono utilizzati senza modificare quelli esistenti.

Questo meccanismo garantisce all'utente il controllo sull'integrità dei dati inseriti, mantenendo al tempo stesso la fluidità del flusso di lavoro.

Capitolo 7

Sviluppo

Questo capitolo descrive le principali attività di sviluppo svolte nell'ambito del progetto Insurance Estimator, con l'obiettivo di implementare quanto definito nella fase di progettazione. L'implementazione è stata suddivisa logicamente in tre macroaree: frontend, backend ed estrazione dei dati.

7.1 Sviluppo del frontend

Ho suddiviso ogni funzionalità dell'applicativo in pagine, componenti e moduli di servizio, in linea con quanto definito nella fase di progettazione.

Per ciascuna sezione dell'applicativo ho definito un insieme di componenti specializzati, responsabili della visualizzazione e della gestione dei dati. In particolare, ho strutturato il frontend in tre livelli principali:

- Pagina: rappresenta la vista principale per ciascuna rotta dell'applicativo. Ogni pagina gestisce lo stato globale e l'interazione tra i vari componenti interni;
- Hook personalizzati: ogni pagina utilizza uno o più hook definiti ad hoc per incapsulare la logica relativa all'accesso ai dati e alla loro validazione, mantenendo così separata la logica di business dalla logica di presentazione;

- Servizi: all'interno del modulo `services`, ho implementato le funzioni che gestiscono le chiamate HTTP al backend (tramite `fetch`), secondo i metodi previsti (GET, POST, PUT, DELETE).

Ho inoltre curato la coerenza visiva e funzionale dell'interfaccia utente adottando la libreria `shadcn/ui`, che mi ha permesso di sviluppare rapidamente componenti accessibili e visivamente coerenti, basati su `Tailwind CSS`. Questa libreria mi ha fornito una base solida per mantenere uno stile minimale, modulare e facilmente adattabile, evitando al tempo stesso un'eccessiva personalizzazione manuale.

Per la gestione dell'internazionalizzazione ho utilizzato `i18next`, grazie a questa libreria ho potuto predisporre l'interfaccia al supporto multilingua fin dalle prime fasi dello sviluppo, garantendo una struttura facilmente estendibile per contesti internazionali.

Un aspetto particolarmente rilevante ha riguardato la gestione della navigazione. Ho implementato il sistema di routing utilizzando `React Router`. Ho strutturato la configurazione delle rotte in modo gerarchico, utilizzando:

- Rotte annidate, per riflettere la relazione logica tra entità;
- Parametri dinamici, che mi hanno consentito di gestire in modo flessibile l'accesso a contenuti specifici attraverso URL dinamiche.

Questa configurazione ha reso possibile una navigazione intuitiva e coerente con la struttura ad alto livello dell'applicativo, riducendo il rischio di comportamenti inattesi durante la fruizione da parte dell'utente.

7.2 Sviluppo del backend

Lo sviluppo del backend ha seguito fedelmente l'architettura progettuale definita in precedenza, basata sulla separazione tra rotte, controller e repository. Tutte le rotte implementate sono state organizzate per dominio funzionale - ad esempio `clienti`, `preventivi`, `polizze`, `compagnie`, `schedari` - e mappate nel file principale di avvio dell'applicazione. Questo approccio mi ha permesso di mantenere una struttura modulare e facilmente scalabile.

Ogni rotta espone un endpoint HTTP che accetta richieste provenienti dal frontend, ne valida i parametri iniziali e li inoltra al controller dedicato. All'interno del controller ho definito la logica applicativa di alto livello, occupandomi di gestire i dati ricevuti e di orchestrare le operazioni tra validazioni, estrazioni dati (dove previste) e interazioni con il database.

L'accesso ai dati è stato incapsulato nel livello **repository**, dove ho implementato le principali operazioni CRUD (creazione, lettura, aggiornamento, eliminazione) per ciascuna collezione del database MongoDB. Tale separazione ha facilitato la riusabilità del codice e semplificato la manutenzione dell'intera applicazione.

Per garantire performance elevate, ho sfruttato le caratteristiche asincrone e non bloccanti del runtime Node.js, in combinazione con il framework **Fastify**. Quest'ultimo mi ha fornito strumenti leggeri e ad alte prestazioni per la gestione delle richieste HTTP, oltre a un sistema di validazione automatica degli input basato su schema, integrato direttamente a livello di rotta.

7.3 Sviluppo del flusso di estrazione dati

Lo sviluppo del flusso di estrazione dati ha rappresentato una delle attività più complesse e centrali del progetto. Ho adottato un'architettura ibrida, in cui il backend Node.js invoca uno script Python come processo figlio, delegando a quest'ultimo l'analisi del contenuto dei file PDF.

Librerie utilizzate

Per l'implementazione degli script Python ho fatto uso di due librerie principali: **pdfplumber** e **camelot**. Queste librerie, sebbene differenti per finalità, si sono rivelate complementari nella gestione di documenti con layout semi-strutturati.

pdfplumber

`pdfplumber` è una libreria Python costruita su `pdfminer.six`, pensata per l'estrazione di contenuti testuali da PDF, inclusi layout, box e coordinate. L'ho utilizzata nei seguenti casi:

- Estrazione di testo leggibile da intere pagine o da porzioni specifiche tramite bounding box;
- Segmentazione del contenuto testuale in blocchi tematici, basata su pattern testuali e regex (es. "AZIENDA", "DOMICILIO", ecc.);
- Gestione di layout a doppia colonna, mediante divisione logica della pagina e lettura separata delle due aree;
- Riconoscimento di intestazioni geografiche come le province, tramite espressioni regolari applicate al testo della prima pagina.

camelot

`camelot` è una libreria Python progettata per l'estrazione automatica di tabelle da file PDF e la loro conversione in strutture dati come `CSV`, `JSON` o `pandas.DataFrame`. Il suo funzionamento si basa sull'analisi visiva e geometrica del layout del documento, e offre due modalità operative distinte, denominate `flavor: lattice` e `stream`.

- **Lattice** è il flavor più adatto a documenti in cui le tabelle sono delimitate da linee di contorno visibili. Internamente, `camelot` utilizza `OpenCV` per rilevare linee orizzontali e verticali e costruisce una griglia logica basata sugli incroci tra queste linee. Le celle vengono quindi identificate in corrispondenza delle aree delimitate;
- **Stream** si basa sull'analisi delle coordinate spaziali e della spaziatura tra blocchi testuali. Il contenuto viene segmentato in righe e colonne in base alla distanza orizzontale e verticale tra le parole, inferendo così la struttura tabellare anche in assenza di tracciati visivi.

Nel contesto di questo progetto, ho scelto di utilizzare il flavor `stream` per l'estrazione del Quadro B dei documenti schedario, in quanto offre maggiore flessibilità nella segmentazione e fornisce parametri personalizzabili per adattare l'estrazione al layout del PDF caricato.

L'estrazione è stata configurata utilizzando anche altri parametri avanzati messi a disposizione da `camelot`, tra cui:

- `pages`: per selezionare le pagine su cui eseguire il parsing;
- `strip_text`, `edge_tol`, `row_tol`: parametri fondamentali per il fine-tuning dell'algoritmo di parsing in modalità `stream`, che influenzano il modo in cui il testo viene pulito e segmentato all'interno delle tabelle. In particolare:
 - `strip_text="\n"`: specifica i caratteri da rimuovere dal testo all'interno delle celle. Ho utilizzato il valore `"\n"` per eliminare i ritorni a capo, che in molti casi spezzavano erroneamente i contenuti su più righe, compromettendo la leggibilità dei dati tabellari;
 - `edge_tol=100`: imposta la tolleranza massima per considerare una parola come appartenente al bordo orizzontale di una tabella. Un valore elevato come 100 è stato scelto per garantire una maggiore tolleranza in presenza di testi vicini ai margini o con layout non perfettamente centrato, evitando così che porzioni valide del contenuto venissero scartate;
 - `row_tol=10`: definisce la tolleranza verticale per raggruppare parole sulla stessa riga. Ho adottato il valore 10 per consentire una discreta elasticità nella rilevazione delle righe, utile nei casi in cui l'allineamento verticale dei testi non è preciso, ad esempio per righe con campi numerici di larghezza variabile o con valori assenti.
- `flavor="stream"`: per attivare la modalità di parsing spaziale.

Le tabelle estratte vengono restituite come oggetti contenenti l'attributo `.df`, ovvero la rappresentazione tabellare sotto forma di `pandas.DataFrame`.

Iterando sulle righe, ho trasformato i dati in array di celle, infine convertiti in JSON. Questi dati vengono poi integrati con i contenuti testuali estratti tramite `pdfplumber`, così da ottenere una rappresentazione completa e strutturata dell'intero contenuto del documento.

Script e integrazione nel backend

Ho realizzato tre script principali in Python:

1. `parse-client.py`, per estrarre le informazioni testuali anagrafiche del cliente dal Quadro A. Esso individua i blocchi logici presenti all'interno del documento ed estrae le informazioni associandole al blocco di appartenenza;
2. `parse-pdf.py`, per combinare l'estrazione di tabelle dal Quadro B con il testo presente nella pagina. In questo caso ho utilizzato entrambe le librerie selezionate: `camelot` gestisce l'estrazione delle tabelle, mentre `pdfplumber` si occupa dell'estrazione delle informazioni testuali;
3. `parse-estimation.py`, per leggere e strutturare il contenuto dei documenti di stima in formato multicolonna. Dato che tali documenti sono strutturati in due colonne, lo script prima estrae le informazioni riguardo le province associate al documento, poi divide la pagina verticalmente in due parti e ne estrae il testo.

Gli script vengono avviati dal backend tramite la creazione di un processo figlio asincrono. Il risultato dell'elaborazione è restituito in formato JSON sullo `stdout`, raccolto dal backend e mappato in un oggetto TypeScript conforme agli schemi dell'applicativo. Questo approccio mi ha permesso di mantenere separata la logica di parsing documentale da quella di gestione dati.

7.4 Condivisione delle interfacce TypeScript

Durante lo sviluppo dell'applicativo, ho rivolto particolare attenzione alla coerenza dei dati tra frontend e backend. A tal fine, ho adottato un approc-

cio basato sulla condivisione delle interfacce TypeScript, che ha permesso di mantenere allineata la struttura dei dati nei diversi livelli dell'architettura.

Ho definito le interfacce all'interno di un modulo dedicato e utilizzato da entrambe le parti del sistema. Ogni entità principale - come `Client`, `Register`, `InsuranceQuote`, `InsuranceCompany`, `InsurancePolicies` - è stata modellata tramite una corrispondente interfaccia, che ne definisce esplicitamente le proprietà, i tipi e gli eventuali annidamenti. Per ciascuna entità del dominio, ho organizzato le interfacce secondo una struttura gerarchica che ha permesso di distinguere tra i diversi contesti di utilizzo (frontend, backend, database), mantenendo al contempo un nucleo comune riutilizzabile.

Ad esempio, per l'entità `Client`, sono state definite tre interfacce principali:

- `ClientBase`: contiene gli attributi comuni del cliente, condivisi sia dal frontend che dal backend;
- `ClientDB`: estende `ClientBase` e aggiunge l'attributo `_id` di tipo `ObjectId`, utilizzato per rappresentare un documento MongoDB all'interno del backend;
- `Client`: estende `ClientBase` e aggiunge `_id` di tipo `string`, formato con cui l'identificativo viene serializzato nel frontend.

Questo schema ha consentito di:

- Evitare la duplicazione del codice relativo alle proprietà comuni;
- Gestire in modo flessibile le differenze tra rappresentazioni interne (tipizzate per MongoDB) ed esterne (JSON inviato al frontend);
- Migliorare la chiarezza del codice, esplicitando il contesto di utilizzo di ciascuna interfaccia.

Questa scelta si è rivelata particolarmente utile anche per l'integrazione dei dati estratti dai documenti PDF, in quanto li ho convertiti direttamente in oggetti conformi alle interfacce definite, riducendo la necessità di conversioni intermedie e semplificando il flusso di mappatura.

Capitolo 8

Conclusioni

8.1 Consuntivo finale

Lo stage curricolare ha avuto una durata complessiva di 304 ore. Tutti gli obiettivi obbligatori, desiderabili e facoltativi presenti nel piano di lavoro sono stati raggiunti con successo. In particolare quelli riguardanti:

- Sviluppo di un'interfaccia per il caricamento di un file;
- Sviluppo di un'interfaccia per la visualizzazione dei dati anagrafici e delle righe estrapolate dal file;
- Sviluppo di un algoritmo di raggruppamento delle righe;
- Gestione delle righe estrapolate: modifica, eliminazione o aggiunta righe;
- Sviluppo di una soluzione back-office per l'inserimento di parametri assicurativi;
- Possibilità di aggiungere una riga di surplus ad ogni raggruppamento.

In aggiunta a quanto stabilito inizialmente, ho avuto modo di ampliare il progetto, contribuendo allo sviluppo di ulteriori funzionalità non previste nel piano originario:

- Gestione delle polizze assicurative;
- Gestione delle compagnie assicurative;

- Gestione delle stime;
- Flusso di estrazione dati dai file PDF contenenti le stime €/quintale;
- Generazione di un preventivo a partire da uno schedario presente nell'applicativo.

Anche tutti i prodotti attesi sono stati realizzati secondo quanto pianificato, rispettando la struttura modulare delineata in fase di progettazione. L'adozione di tecnologie moderne e la divisione logica tra frontend, backend e moduli di estrazione ha favorito il raggiungimento degli obiettivi nei tempi previsti, con un elevato livello di affidabilità e coerenza interna del sistema.

8.2 Valutazione personale

L'esperienza di stage si è rivelata estremamente positiva e formativa. Ho avuto l'opportunità di mettere in pratica molte delle conoscenze acquisite durante il percorso universitario, confrontandomi con un progetto reale e strutturato secondo principi consolidati di ingegneria del software. Partecipare attivamente alla progettazione e allo sviluppo di un'applicazione completa mi ha permesso di acquisire nuove competenze tecniche, approfondendo l'utilizzo di strumenti e tecnologie moderne come React, Fastify, MongoDB e Docker.

Allo stesso tempo, lo stage mi ha dato modo di lavorare in autonomia, rispettando vincoli progettuali e requisiti reali. Un ulteriore aspetto che ha arricchito l'esperienza è stata la possibilità di osservare e vivere direttamente l'applicazione del metodo Scrum all'interno dell'azienda. Ho partecipato regolarmente a incontri di pianificazione, revisioni di sprint e momenti di confronto quotidiano, comprendendo da vicino l'importanza dell'organizzazione iterativa del lavoro e della collaborazione all'interno di un team.

In definitiva, questo progetto ha rappresentato un importante valore aggiunto per la mia formazione, permettendomi di integrare teoria e pratica in modo concreto e di acquisire una maggiore consapevolezza sulle dinamiche e sulle esigenze dello sviluppo software nel contesto professionale.

Glossario

BSON BSON (Binary JSON) è un formato di serializzazione binaria basato su JSON, progettato per essere più efficiente nella memorizzazione e trasmissione dei dati per database come MongoDB. A differenza di JSON, che è un formato di testo, BSON utilizza una rappresentazione binaria, rendendolo più compatto e veloce da analizzare. [59](#)

Catene di Markov Le catene di Markov sono modelli matematici che descrivono un sistema che passa da uno stato all'altro secondo una logica probabilistica, dove la probabilità del prossimo stato dipende solo dallo stato attuale, non dalla sequenza completa degli stati precedenti. [38](#)

Collection In MongoDB, una collection è un contenitore di documenti, simile a una tabella in un database relazionale. In esse vengono archiviati i dati, e ogni documento all'interno di una collection è un oggetto di dati strutturato in formato BSON. [60](#)

Container I container sono ambienti isolati che includono tutto il necessario per eseguire un'applicazione — codice, runtime, librerie, dipendenze — garantendo coerenza tra i diversi ambienti di esecuzione. [22](#)

Debito tecnico Il debito tecnico è una metafora usata nello sviluppo software per descrivere il costo a lungo termine di scelte rapide, ma non ottimali, fatte per velocizzare il rilascio di un prodotto. [5](#)

Fine-tuning Il fine-tuning è il processo di addestramento ulteriore di un modello pre-addestrato, adattandolo a un compito specifico o a un dominio particolare attraverso un dataset più mirato. [54](#)

GitLab GitLab è una piattaforma web open source per la gestione del ciclo di vita dello sviluppo software. Offre strumenti per la gestione di repository Git, il tracciamento dei problemi (issue), la revisione del codice, il Continuous Integration/Continuous Delivery (CI/CD) e altre funzionalità per lo sviluppo collaborativo e la gestione dei progetti software. [12](#)

Hook Un hook è una funzione speciale che permette di “agganciare” funzionalità specifiche di React, come lo stato e i metodi del ciclo di vita, nei componenti funzionali. [3](#), [61](#)

Immagine raster Un’immagine raster è un’immagine digitale composta da una griglia di pixel. Ogni pixel ha un colore specifico e la combinazione di tutti i pixel crea l’immagine complessiva. [31](#)

Metadati I metadati sono informazioni che descrivono altri dati, ovvero forniscono dettagli su un documento, un file o una risorsa, aiutando a capirne il contenuto, la struttura e il contesto. [45](#)

Milestone In project management, una milestone è un traguardo significativo o un punto di controllo all’interno di un progetto che indica il completamento di una fase importante o il raggiungimento di un obiettivo specifico. [18](#)

Modelli a n-grammi I modelli a n-grammi sono modelli probabilistici usati nel trattamento del linguaggio naturale per prevedere la probabilità di una parola basandosi sulle parole precedenti. [38](#)

Self-attention Il meccanismo di self-attention consente al modello di rappresentare ogni parola (o token) non isolatamente, ma tenendo conto del contesto completo in cui appare, sia a sinistra che a destra in modo bidirezionale. [43](#)

Transfer learning Il transfer learning (apprendimento per trasferimento) è una tecnica dell’intelligenza artificiale e del machine learning in cui un modello già addestrato su un compito viene riutilizzato o adattato per un

altro compito correlato, solitamente con poche nuove informazioni o pochi dati. [52](#)

Bibliografia

Testi

Kurzweil, Ray. *The Age of Intelligent Machines*. 1990.

Articoli

Barakat, Berat et al. «Text Line Segmentation for Challenging Handwritten Document Images Using Fully Convolutional Network». In: (2021).

Chaudhury, Ayan et al. «A Deep OCR for Degraded Bangla Documents». In: (2022).

Devlin, Jacob et al. «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding». In: (2018).

Fateh, Amirreza, Mansoor Fateh e Vahid Abolghasemi. «Enhancing optical character recognition: Efficient techniques for document layout analysis and text line detection». In: (2023).

Huang, Yupan et al. «LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking». In: (2022).

Jogin, Manjunath et al. «Feature Extraction using Convolution Neural Networks (CNN) and Deep Learning». In: (2018).

Kaur, Sukhvir e P. S. Mann. «Page Segmentation in OCR System-A Review». In: (2013).

Lample, Guillaume et al. «Neural Architectures for Named Entity Recognition». In: (2016).

- Li, Minghao et al. «TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models». In: (2021).
- Liu, Yinhan et al. «RoBERTa: A Robustly Optimized BERT Pretraining Approach». In: (2019).
- Llobet, Rafael et al. «Efficient OCR Post-Processing Combining Language, Hypothesis and Error Models». In: (2010).
- Martinez, David R. et al. «Artificial Intelligence: Short History, Present Developments, and Future Outlook». In: (2019).
- Otsu, Nobuyuki. «A threshold selection method from gray-level histograms». In: (1979).
- Riaz, Mehak, Syed Muhammad Ghazanfor Monir e Rija Hasan. «Evaluation of deep learning approaches for optical character recognition in Urdu language». In: (2022).
- Wang, Junmiao. «A Study of the OCR Development History and Directions of Development». In: (2023).
- Xu, Yang et al. «LayoutLMv2: Multi-modal Pre-training for Visually-Rich Document Understanding». In: (2020).
- Xu, Yiheng et al. «LayoutLM: Pre-training of Text and Layout for Document Image Understanding». In: (2020).
- Yang, Xiao et al. «Learning to Extract Semantic Structure from Documents Using Multimodal Fully Convolutional Neural Network». In: (2017).
- Zhou, Jie et al. «Graph Neural Networks: A Review of Methods and Applications». In: (2020).

Sitografia

Atlassian - Agile. URL: <https://www.atlassian.com/agile> (cit. a p. 4).

Atlassian - Epics. URL: <https://www.atlassian.com/agile/project-management/epics> (cit. a p. 12).

Atlassian - Gitflow Workflow. URL: <https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow> (cit. a p. 14).

Atlassian - Scrum. URL: <https://www.atlassian.com/agile/scrum> (cit. a p. 7).

Atlassian - Story Points. URL: <https://www.atlassian.com/agile/project-management/estimation> (cit. a p. 12).

Atlassian - User Stories. URL: <https://www.atlassian.com/agile/project-management/user-stories> (cit. a p. 12).

Camelot. URL: <https://camelot-py.readthedocs.io/en/master/> (cit. a p. 21).

Docker. URL: <https://www.docker.com/> (cit. a p. 22).

Fastify. URL: <https://fastify.dev/> (cit. a p. 21).

GitLab. URL: <https://about.gitlab.com/> (cit. a p. 18).

i18next. URL: <https://www.i18next.com/> (cit. a p. 20).

Manifesto Agile. URL: <https://agilemanifesto.org/iso/it/manifesto.html> (cit. a p. 5).

MongoDB. URL: <https://www.mongodb.com/> (cit. a p. 22).

Node.js. URL: <https://nodejs.org/en> (cit. a p. 20).

pdfplumber. URL: <https://github.com/jsvine/pdfplumber> (cit. a p. 21).

Python. URL: <https://www.python.org/> (cit. a p. 21).

React. URL: <https://react.dev/> (cit. a p. 19).

React Router. URL: <https://reactrouter.com/> (cit. a p. 20).

Shadcn/ui. URL: <https://ui.shadcn.com/> (cit. a p. 19).

TanStack Table. URL: <https://tanstack.com/table/latest> (cit. a p. 19).

TypeScript. URL: <https://www.typescriptlang.org/> (cit. a p. 20).

Zod. URL: <https://zod.dev/> (cit. a p. 20).