
Politecnico di Torino

**Politecnico di Torino**

Ingegneria Informatica

A.a. 2024/2025

Sessione di laurea Aprile 2025

# Sistema di gestione avanzata dei processi di check-in e check-out nel noleggio di autoveicoli

Relatori:

Luigi De Russis

Giovanni Campolo

Candidato:

Salvatore Acquaviva

# Ringraziamenti

A tutti coloro che, da vicino o lontano, mi hanno aiutato a crederci sempre.

# Indice

# Elenco delle figure

# Capitolo 1

# Introduzione

Il settore Automotive sta vivendo negli ultimi anni una trasformazione radicale dovuta all'affermarsi, in modo sempre più evidente, del mondo digitale. La nascita di servizi di Intelligenza Artificiale sempre più specializzati, così come l'offerta di tecnologie avanzate basate su Big Data e Internet of Things (IoT), spinge sempre più le aziende automobilistiche a modificare tutti i flussi di lavoro gestiti manualmente al fine di automatizzarli, garantendo efficienza operativa e basso indice di errore.

L'innesto della tecnologia nell'intera filiera industriale fornisce, inoltre, un miglioramento non indifferente anche a livello funzionale: algoritmi di machine learning sono, infatti, in grado di analizzare profondamente dati di marketing e proporre offerte in base alle preferenze o necessità dei clienti; allo stesso tempo alcuni sistemi predittivi sono in grado di anticipare guasti a veicoli e segnalarli tempestivamente, in modo da consentire interventi di manutenzione preventivi. Un chiaro esempio di questa evoluzione è rappresentato dai Software-Defined Vehicles (SDV), veicoli in cui le funzionalità principali sono controllate e migliorate attraverso aggiornamenti software. Questo approccio consente una maggiore personalizzazione, integrazione con servizi digitali e un'evoluzione continua delle prestazioni del veicolo, ridefinendo il concetto stesso di mobilità.

Sebbene la digitalizzazione mostri svariati vantaggi, allo stesso tempo molteplici sfide vanno prese in considerazione, relative per esempio alla sicurezza, alla privacy dei dati, agli alti costi di investimento e all'integrazione con altre tecnologie legacy. Il passaggio da procedure consolidate a nuovi modi di operare può inoltre trovare una forma di resistenza da parte dell'azienda, causata spesso dalla paura dell'ignoto. Al netto di tutto ciò, secondo le previsioni stimate dal Boston Consulting Group (BCG), il valore degli SDV supererà i 650 miliardi di dollari entro il 2030 [1]. Questo trend sottolinea l'importanza crescente dello sviluppo software nel settore automobilistico, poiché i veicoli stanno diventando sempre più dipendenti da applicazioni e sistemi digitali per migliorare funzionalità, sicurezza e connettività.

## 1.1 Contesto

Le aree nel settore Automotive coinvolte nella corsa alla digitalizzazione sono molteplici ed eterogenee, spaziando dalla produzione alla manutenzione predittiva, fino alla guida autonoma. Ad esempio, nella produzione si assiste all'integrazione di tecnologie avanzate come l'Industrial IoT (IIoT) e l'intelligenza artificiale, che permettono di ottimizzare le linee di assemblaggio e ridurre gli sprechi. La manutenzione predittiva, grazie all'analisi dei dati in tempo reale raccolti dai sensori installati sui veicoli, consente di prevenire guasti e ridurre i tempi di fermo. Infine, un settore in forte crescita è quello della guida autonoma, in cui la digitalizzazione gioca un ruolo centrale attraverso l'uso di machine learning, computer vision e big data per migliorare la sicurezza e la precisione della navigazione. Tuttavia, ognuna di queste aree può presentare, ancora oggi, processi e attività che, per motivi storici, etici o economici, sono ancora legate a pratiche artigianali o obsolete. In un ambiente simile, l'intelligenza artificiale trova ampie possibilità di sviluppo, arricchendo ulteriormente le proposte innovative che intendono evolvere o addirittura sostituire il modo operativo consolidato. L'esplorazione di queste procedure, insieme allo studio della Power Platform di Microsoft, ha gettato le fondamenta per la realizzazione di questo progetto nel contesto specifico del noleggio auto.

Il noleggio auto è un settore che sta vivendo un'espansione notevole negli ultimi anni, alimentata dall'emergere di nuovi modelli come il noleggio a lungo termine (NLT), il car sharing e l'abbonamento auto. Questa crescita è strettamente legata alla crescente domanda di soluzioni di mobilità flessibili, sia per scopi turistici che per esigenze professionali, rendendo il mercato una delle aree più dinamiche nel panorama globale. Considerando anche le implicazioni fiscali ed inflazionistiche, il noleggio rappresenta ad oggi un'alternativa all'acquisto in termini di minore immobilizzazione del capitale. Il rapido sviluppo del settore sta spingendo le aziende tradizionali a rivedere e innovare le loro operazioni per restare competitive. L'ingresso di nuove startup e tech companies, come Uber, Free2Move e Lynk & Co, sta introducendo modelli di business all'avanguardia e intensificando la concorrenza, costringendo le imprese storiche ad adattarsi a un mercato in continua evoluzione. Tutto ciò ha messo in evidenza l'esigenza di modernizzare i processi operativi per incrementare l'efficienza, ridurre i costi di gestione e migliorare l'esperienza complessiva del cliente. Sul piano tecnologico, per esempio, l'uso di app, IoT e intelligenza artificiale sta rivoluzionando l'esperienza di noleggio, con prenotazioni istantanee e gestione smart dei veicoli. L'integrazione di sistemi avanzati di analisi consente, inoltre, di migliorare l'utilizzo delle risorse e offrire un servizio più efficiente e personalizzato ai clienti. In questo contesto, l'innovazione digitale non è più un'opzione, ma un requisito essenziale per le aziende che desiderano non solo rimanere competitive, ma anche anticipare le tendenze del mercato e soddisfare le crescenti aspettative dei consumatori.

## 1.2 Un approccio Low-Code

Il crescente fabbisogno di sistemi informatici favorisce, per quanto concerne lo sviluppo software, l'utilizzo di metodologie Low-Code. L'approccio Low-Code consente, infatti, di realizzare programmi complessi in tempi decisamente minori rispetto agli approcci tradizionali, permettendo ai produttori software di essere competitivi sul mercato e pronti a soddisfare nuove richieste. La software factory WebRatio ha addirittura stimato un incremento della produttività di sviluppo del 700% rispetto ai metodi high-code, ovvero quelli di programmazione tradizionale, dimostrando quanto questo paradigma sia in grado di stravolgere i tempi di produzione [2]. Ma, approfondendo la tematica, in cosa consiste il Low-Code?

Una pubblicazione di IBM definisce questa strategia come "un approccio visivo allo sviluppo software" attraverso una minima scrittura di codice [3]. Le piattaforme che abbracciano tale metodologia sono spesso caratterizzate da interfacce grafiche user-friendly, le quali abilitano sviluppatori, anche con poca esperienza di programmazione, a creare soluzioni IT in modo semplice e a poterle consegnare rapidamente al cliente. Il Low-Code disaccoppia, dunque, le capacità di coding del programmatore dal processo di sviluppo di un'applicazione. Questo rappresenta un ulteriore vantaggio, poiché il numero di specialisti del settore non è sempre adeguato a consentire ad un'azienda di gestire efficacemente il proprio carico di lavoro. I benefici evidenziati con questa tecnica generano, a loro volta, un effetto domino di ulteriori vantaggi, tra i quali la diminuzione del payback period dell'investimento, la facile integrazione con altre tecnologie e sistemi aziendali e l'incentivazione alla sperimentazione e al rilascio rapido di nuove idee.

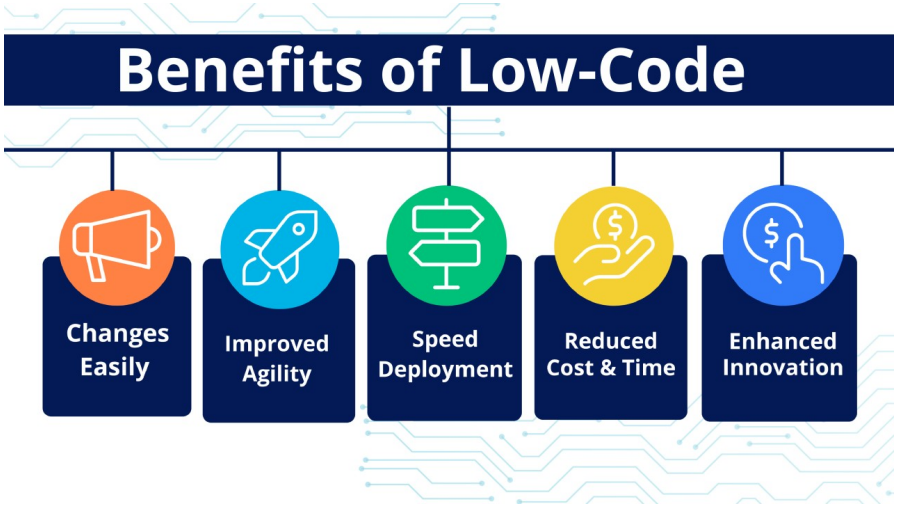


**Figura 1.1:** Benefici del Low-Code

In questo contesto si inserisce Cluster Reply, azienda del gruppo Reply specializzata in consulenza, system integration e sviluppo di soluzioni Low-Code basate su tecnologie Microsoft [4]. Questo progetto di Tesi è stato svolto, nello specifico, presso la Business Unit Automotive di **Cluster Reply DCX**, società focalizzata su delle progettualità che hanno come cuore pulsante Microsoft Dynamics 365 - la suite di applicazioni aziendali di Microsoft integrante sia funzionalità CRM che ERP - e la Microsoft Power Platform, applicativi per la creazione delle più eterogenee applicazioni di business (Web App, Mobile App, Chatbot, siti web e flussi di lavoro).

## 1.3 Cos'è un CRM

Dall'ottava edizione dell'Osservatorio CRM (una ricerca condotta dal 2015 da C-Direct Consulting e dal 2020 in stretta collaborazione con Engineering) emerge come il 54% delle aziende in Italia analizzi regolarmente i dati relativi ai propri clienti, mentre un altro 34% lo faccia saltuariamente [5]. La scienza della Customer Analytics risulta essere sempre più al centro delle attività presenti sul mercato, con diverse tecnologie utilizzate per monitorare i dati e prendere decisioni basate su di essi.

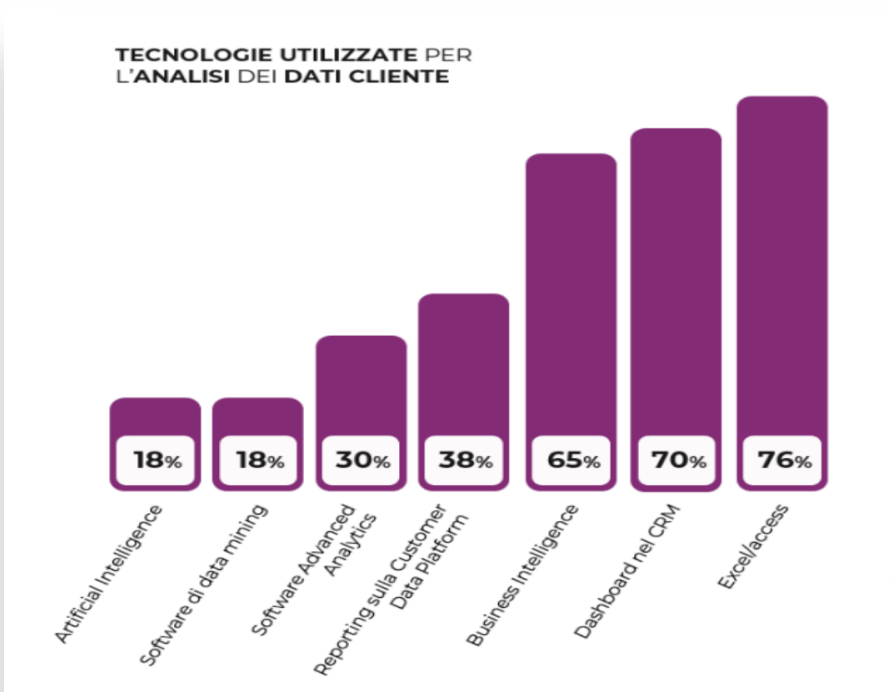


**Figura 1.2:** Tecnologie utilizzate per l'analisi dei dati cliente

Dalla Figura 1.2 si evince come Excel sia ancora lo strumento più adottato per l'analisi della Customer Base, seguito subito dopo dalle dashboard CRM. Il software CRM costituisce, insieme all'utilizzo dell'approccio Low-Code, una colonna portante di questo lavoro di Tesi, per cui verrà approfondito di seguito.

Il Customer Relationship Management è un sistema aziendale che consente di gestire le interazioni con i clienti acquisiti e potenziali. A prima vista, questa tecnologia sembrerebbe riconducibile a un semplice archivio dati, dove vengono memorizzati dati anagrafici dei clienti e dettagli relativi alle vendite. L'obiettivo principale di un CRM va, tuttavia, ben oltre la mera centralizzazione delle informazioni, trattandosi di uno strumento sofisticato in grado di:

- **Fidelizzare il cliente**, personalizzando e ottimizzando la gestione del rapporto;
- **Automatizzare processi**, incrementando l'efficienza e la redditività aziendale;
- **Favorire la condivisione di informazioni**, rendendole accessibili tra i vari team;
- **Integrare servizi AI**, assicurando una maggiore accuratezza delle previsioni e fornendo statische complesse relative ai trends di marketing, alle vendite e alle performance;
- **Fornire assistenza avanzata ai clienti** attraverso chatbot, meccanismi di analisi della soddisfazione e sistemi di ticketing;
- **Abbassare i costi**, garantendo meno lavoro amministrativo e ottimizzando le risorse.

Un software del genere può essere destinato ad aziende di qualsiasi dimensione, proprio grazie alle caratteristiche sopra elencate e rivelarsi estremamente utile in tutti i reparti aziendali. Questa sua versatilità la rende una tecnologia ormai matura, adottata in Italia dal 69% delle aziende, un dato che è cresciuto in modo costante negli ultimi anni e che si appresta a stabilizzarsi attorno a questa percentuale [5]. Tra gli esempi di CRM maggiormente diffusi, leader indiscussi sono sicuramente Salesforce e Microsoft Dynamics.

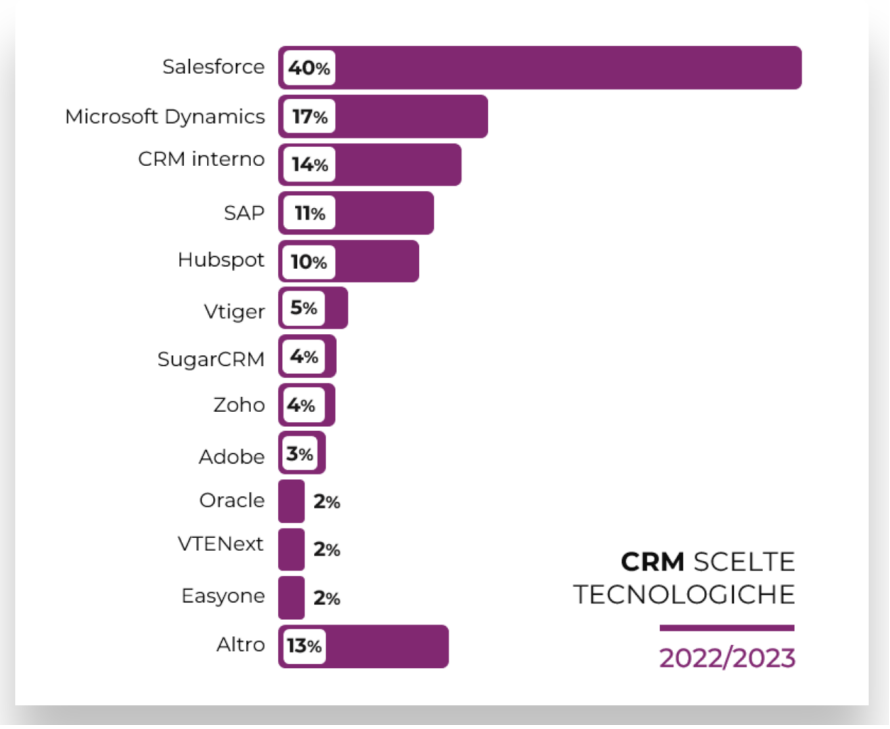


**Figura 1.3:** CRM: Scelte tecnologiche

## 1.4 Obiettivo della tesi

I vari passaggi che costituiscono un noleggio auto sono molteplici e diversificati. Tra le fasi più delicate di questo processo spiccano sicuramente i momenti del **check-in** e del **check-out**, rappresentando i punti in cui vengono svolte una serie di attività fondamentali. Durante il check-in vengono, per esempio, registrati i dati del cliente, verificata la documentazione e consegnate le chiavi del veicolo. Al check-out, invece, viene riconsegnata l'auto e chiuso il contratto di noleggio con l'addebito delle spese eventualmente derivanti da danni o extra chilometraggio. Entrando più nel dettaglio, un momento decisivo che accumuna entrambe le fasi e che gioca un ruolo determinante nell'intero processo è senz'altro quello in cui vengono verificati i danni al veicolo rispetto allo stato precedente.

Attualmente, questa operazione di ispezione è spesso effettuata manualmente dagli operatori o dai clienti stessi, attraverso ispezioni visive e fotografie. Tuttavia, tale approccio presenta diverse criticità: richiede un confronto con documentazione cartacea o database non sempre accessibili in tempo reale e risulta rallentato dalla necessità di compilare manualmente il modulo di ispezione del veicolo. Una gestione imprecisa della rilevazione dei danni può generare contestazioni tra l'azienda di noleggio e il cliente, con potenziali perdite economiche dovute a riparazioni

non attribuite correttamente o a tempi di fermo prolungati. Inoltre, l'aumento della mobilità condivisa e la diffusione di sistemi di noleggio self-service rendono ancora più necessaria un'automazione affidabile di questi controlli, cercando al contempo di non rallentare tali operazioni. Ciò che si vorrebbe ottenere è, piuttosto, l'accelerazione sia del processo aziendale che dei tempi di attesa per il cliente durante le fasi di check-in e check-out.

Alla luce di queste considerazioni, questo lavoro di Tesi si prefigge come obiettivo la progettazione e l'implementazione di un sistema informatico in grado di rilevare, in modo smart, eventuali danni ad autoveicoli durante le fasi di check-in e check-out di un noleggio. La soluzione proposta punta ad automatizzare il processo di ispezione tramite l'utilizzo di algoritmi di AI capaci di intercettare eventuali danni analizzando le foto scattate dall'utente. Il sistema è pensato, inoltre, per essere scalabile e configurabile, così da adattarsi alle diverse esigenze delle aziende, indipendentemente dal volume di veicoli gestiti o dalle specifiche modalità operative. Dal punto di vista delle performance, il sistema aspira a ottimizzare i tempi di gestione e ridurre il tasso di errore nei controlli, contribuendo di fatto a un processo più rapido e affidabile.

## 1.5 Struttura della tesi

Nei capitoli successivi verranno esposti i principali step di sviluppo che hanno accompagnato questo percorso di Tesi, seguendo la struttura illustrata di seguito:

- ***Capitolo 2*** **- Tecnologie utilizzate**. In questo capitolo verranno analizzate, in primis, le tecnologie Low-Code della Power Platform di Microsoft necessarie per lo sviluppo del progetto. Dopo una breve panoramica sulla piattaforma Dynamics 365 verranno, infine, descritti i servizi Azure utilizzati per il salvataggio delle foto, la gestione del web server e l'integrazione delle funzionalità di intelligenza artificiale.
- ***Capitolo 3*** **- Progettazione**. Questo capitolo comprenderà l'esposizione dei requisiti funzionali e non funzionali del sistema. Il focus di questa sezione si sposterà successivamente sullo studio di diverse alternative strutturali, analizzando pregi e difetti di ciascuna soluzione ed evidenziando come la scelta di determinati componenti influisca nella realizzazione del prodotto finale. Il capitolo terminerà infine con la presentazione dell'architettura definitiva, del modello dati e del prototipo dell'interfaccia per l'app mobile.
- ***Capitolo 4*** **- Implementazione**. Sulla base delle decisioni progettuali intraprese nel capitolo 3, verranno illustrate le varie fasi che hanno caratterizzato lo sviluppo del sistema. Saranno delineati aspetti tecnici e logici dell'applicazione CRM e di quella mobile. Sarà inoltre esplorato AI Studio,

mostrando in particolare il prompt utilizzato per la comunicazione con il modello di AI. In chiusura verrà presentato il Web Server che si interfaccia con l'intelligenza artificiale e il flusso Power Automate responsabile di connettere i vari componenti, automatizzando l'esecuzione delle richieste e la gestione dei risultati ottenuti.

- ***Capitolo 5* - Risultati e Valutazioni**. In questa sezione saranno fornite delle metriche di valutazione per analizzare le prestazioni del sistema sviluppato, confrontando i risultati ottenuti con i requisiti iniziali.
- ***Capitolo 6* - Conclusione**. Questo capitolo conclusivo riassumerà brevemente il prototipo realizzato sottolineando i suoi punti di forza e le sue vulnerabilità. A partire da queste saranno infine accennate possibili opportunità future di miglioramento.

# Capitolo 2

# Tecnologie utilizzate

## 2.1 Power Platform

La Power Platform è una suite di strumenti Low-Code, progettata per consentire una costruzione rapida di soluzioni aziendali personalizzate e complete [6]. Questa piattaforma, lanciata ufficialmente nel 2018 dal colosso del settore Microsoft, si è subito imposta come risorsa fondamentale per le imprese, supportando lo sviluppo di applicazioni, l'analisi dei dati, l'automazione di processi e la creazione di agenti virtuali. La sua impronta Low-Code è sicuramente una delle caratteristiche che ha contribuito al successo di tale tecnologia. La Power Platform permette, infatti, a chiunque in un'organizzazione di cooperare durante il processo di sviluppo, colmando di fatto il divario tra business e IT e, al contempo, generando soluzioni altamente orientate alle esigenze manageriali. Un altro punto di forza è sicuramente rappresentato dalla sua capacità di integrazione con altri servizi Microsoft (come Teams, Outlook, SharePoint e Azure) oltre che con software di terze parti, come GitHub.

La piattaforma presenta 4 componenti core, ciascuno dei quali è utilizzabile come tecnologia stand-alone oppure in sinergia con altri componenti per gestire specifiche esigenze aziendali:

- *Power Apps* - Strumento utilizzato per l'implementazione di applicazioni che possono essere eseguite su un browser o su un dispositivo mobile;
- *Power Automate* - Componente utilizzabile per automatizzare flussi di lavoro semplici o attività ripetitive con lo scopo di aumentare l'efficienza del singolo step o dell'intero processo;
- *Power Pages* - Software grafico utile per la generazione di siti web. Conosciuto inizialmente come 'Power Apps Portals', Power Pages è stato aggiornato e riorganizzato per rispondere a esigenze sempre più complesse di business;

- *Power BI* - Servizio di analisi aziendale capace di raccogliere tutti i dati di un'organizzazione e di elaborarli in report personalizzati sottoforma di grafici, diagrammi e dashboard. Questo componente offre, quindi, una visione chiara e facilmente comprensibile delle informazioni nascoste dietro i dati, rendendole più accessibili e intuitive;

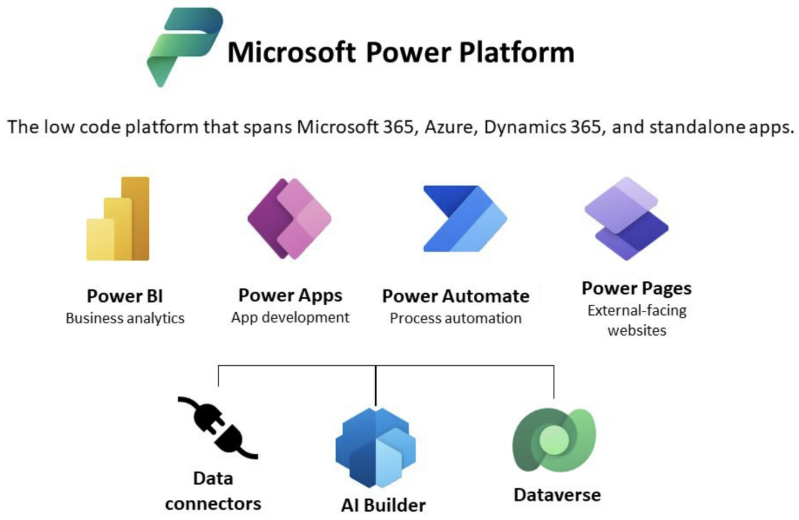


**Figura 2.1:** Struttura della Microsoft Power Platform

In seguito ho voluto approfondire alcuni dei servizi della Microsoft Power Platform, come Power Apps e Power Automate, in quanto rappresentano 2 delle tecnologie utilizzate all'interno della mia soluzione.

### 2.1.1 Power App

Power App è un tool di sviluppo Microsoft che consente la creazione rapida di software aziendali attraverso tecniche Low-Code. Le applicazioni generate tramite Power App si distinguono in **Canvas Apps** e **Model-Driven Apps**.

#### Canvas Apps

Una Canvas App è un tipo di applicazione generata a partire da una *blank canvas*, una tela bianca in cui lo sviluppatore, attraverso un'interfaccia drag-and-drop, trascina e rilascia componenti in base alle proprie necessità [7]. Una volta posizionati e formattati opportunamente gli elementi visivi, lo sviluppatore può concentrarsi

sulla logica dell'applicazione, attraverso un linguaggio dichiarativo basato su formule simili a quelle di Excel, chiamato **Power Fx**. Questo linguaggio open-source è attualmente disponibile soltanto all'interno delle Canvas App, sebbene Microsoft stia lavorando affinché possa essere integrato anche con tutti gli altri prodotti della Power Platform [8]. Le applicazioni Canvas offrono, dunque, massima flessibilità in termini di design ed esperienza utente, lasciando allo sviluppatore un controllo completo sull'aspetto e sul funzionamento dell'interfaccia. Esse, inoltre, sono capaci di connettersi facilmente a un'ampia gamma di servizi e database, sfruttando più di 1.200 **connettori built-in**, oltre alla possibilità di creare **connettori custom**.

### Model-Driven Apps

Le Model-Driven App rappresentano la seconda tipologia di applicazioni sviluppabili attraverso Power App. Esse, a differenza di quelle Canvas, si basano fortemente sul modello dati sottostante. Questa loro natura data-first rende le Model-Driven App più rigide e poco inclini alla personalizzazione dell'interfaccia grafica. Possiamo definire questi tipi di software come l'incarnazione dei principi teorici definiti dalla **Model-Driven Engineering** (in sigla **MDE**). L'idea di base di questo paradigma è che, attraverso la costruzione e l'uso di astrazioni per rappresentare i processi da automatizzare, il software prodotto sarà di qualità superiore rispetto a quello sviluppato con un linguaggio di programmazione generico. Viene, pertanto, enfatizzato l'utilizzo di modelli come artefatti principali della progettazione, essendo più facili da comprendere, verificare e simulare rispetto ai programmi informatici [9]. I dati, a partire dai quali vengono costruite le Model-Driven App, provengono da una piattaforma di archiviazione cloud chiamata **Dataverse**. Questa tecnologia, insieme ai connettori citati in precedenza, rappresenta uno degli strumenti forniti da Microsoft come supporto per l'intera collezione della Power Platform.

Due componenti disponibili per la gestione e visualizzazione dei dati all'interno di queste applicazioni sono i **Form** e le **View**. I Form rappresentano le pagine di dettaglio di singoli record. Attraverso queste sezioni è possibile determinare quali campi mostrare all'end user e la loro disposizione all'interno della schermata. I Form costituiscono, inoltre, l'elemento con il quale si interfaccia l'utente al momento della creazione e della modifica dei dati. Presentano un grado di personalizzazione più elevato rispetto alle View, permettendo di specificare campi obbligatori e regole di validazione in modo semplice e sicuro. D'altra parte le View sono griglie responsabili della visualizzazione di un insieme di record. Esse consentono la ricerca, l'ordinamento e il filtraggio di dati in modo efficiente, lasciando, tuttavia, meno possibilità configurazionali. Grazie a questi strumenti, le Model-Driven Apps consentono di costruire applicazioni robuste con una gestione ottimizzata delle informazioni e dei processi aziendali.

#### Canvas Apps e Model-Driven Apps a confronto

Canvas e Model-Driven App presentano peculiarità specifiche, ognuna delle quali le rende adatte a diversi scenari di sviluppo. Le prime vengono particolarmente indicate in contesti in cui la personalizzazione e l'aspetto estetico dell'interfaccia utente sono prioritari. Le Canvas App sono ideali, infatti, per digitalizzare attività semplici o specifiche, dove non è necessario lavorare con UI complesse. Questa tipologia di applicazioni consente, inoltre, una facile integrazione con fonti di dati diverse come SharePoint, SQL Server ed API esterne, non limitando lo sviluppatore all'utilizzo di Dataverse come unica sorgente. Con le Model-Driven app, invece, l'obiettivo principale è quello di standardizzare processi articolati e strutturati. Questo suo punto di forza le rende adatte alla realizzazione di applicativi aziendali avanzati come sistemi CRM e gestionali, in cui la singolarità delle schermate è sicuramente meno rilevante.

La scelta tra Canvas e Model-Driven App dipende, quindi, dal contesto di utilizzo, senza che ciò implichi necessariamente una preferenza esclusiva per una tipologia rispetto all'altra. Parecchie soluzioni prevedono, infatti, un mix di queste due tecnologie, le quali sono in grado di interagire tra di loro in modo frequente ed efficace.

### 2.1.2 Power Automate

Power Automate di Microsoft è uno strumento di automazione progettato per ottimizzare i processi aziendali, riducendo il tempo dedicato alle attività ripetitive e permettendo alle organizzazioni di focalizzarsi su obiettivi più strategici [10]. Che si tratti di inviare notifiche, raccogliere dati da più fonti o aggiornare sistemi, Power Automate consente agli utenti di creare applicazioni indipendenti altamente personalizzabili, che possono integrarsi con altre tecnologie per estenderne le funzionalità. Analogamente agli altri servizi all'interno della Power Platform, la sua interfaccia semplice e intuitiva rappresenta un motivo di interesse da parte delle aziende moderne che, cercando di restare al passo in un mercato in continua evoluzione, mirano a soddisfare le esigenze dei clienti in modo rapido e diretto.

La piattaforma di Power Automate è interamente progettata attorno al concetto chiave di **Flow**. Esso rappresenta un flusso di lavoro automatizzato, capace di attivarsi in risposta a uno (o più) eventi - chiamati *triggers* - e costituito da una sequenza logica di attività - dette *azioni* - in grado di sostituire compiti manuali e processi aziendali ripetitivi. Al fine di familiarizzare con la creazione di workflow, gli utenti Power Automate dispongono di un insieme di modelli predefiniti dai quali si può prendere spunto, adattandoli successivamente alle proprie necessità. La loro costruzione avviene in modo intuitivo, attraverso procedure guidate che permettono allo sviluppatore di scegliere il trigger che scatenerà il flusso (come la ricezione di una email o la compilazione di un campo di un form), definire le azioni

da eseguire (per esempio la creazione di un file Excel), configurare ciascun elemento con gli opportuni parametri di input e output e organizzarli nell'ordine desiderato. Seppur questo consenta la generazione di un'infinita varietà di flussi personalizzati, Power Automate riconduce ciascun Flow a una delle seguenti tipologie:

- *Flussi Automatici* - Flussi scatenati automaticamente ogni volta che un particolare evento accade all'interno di una specifica piattaforma, ad esempio l'invio di una notifica quando viene ricevuta una nuova email.
- *Flussi Istantanei* - Flussi che si attivano come risultato di un'azione diretta dell'utente, come ad esempio il clic di un pulsante specifico.
- *Flussi Pianificati* - Flussi che servono per portare a termine delle attività con cadenza periodica. Essi scattano, infatti, a intervalli regolari o a determinati istanti di tempo.

Un'ulteriore caratteristica interessante di Power Automate è costituita dalla funzione *Monitora*. Accedendo alla sezione in questione, è possibile, infatti, tracciare l'esecuzione dei propri flussi, ottenendo informazioni relative a quelli riusciti, a quelli in cui si sono presentati degli errori e a quelli annullati. Vengono fornite, inoltre, diverse metriche e dashboard di dettaglio, in modo da aiutare gli utenti a verificare l'efficacia delle loro automazioni e indicare dove intervenire per evitare che falliscano.

### 2.1.3 Dataverse

Dataverse è uno spazio di archiviazione sicuro basato sul cloud che le organizzazioni possono utilizzare per memorizzare i dati delle applicazioni aziendali [7]. Questo servizio, conosciuto fino all'ottobre 2020 come Common Data Service, assume un ruolo orizzontale per l'intera Power Platform, offrendo a tutte le tecnologie Microsoft che ne fanno parte la possibilità di salvare e recuperare le proprie informazioni quando necessario. Dataverse risulta essere, pertanto, un componente cruciale nell'ecosistema Microsoft, garantendo inoltre scalabilità elevata e raggiungibilità a livello globale.

Internamente i dati vengono gestiti in modo da assicurare un facile utilizzo dalle piattaforme Low-Code, rendendoli disponibili senza dover costruire complessi schemi di integrazione. Più precisamente, Dataverse fornisce un ambiente centralizzato per archiviare dati strutturati sotto forma di entità paragonabili a tabelle del modello relazionale.

Similmente alle tecnologie menzionate in precedenza, Dataverse presenta un'interfaccia semplice, grazie anche all'integrazione con Power Query, strumento Microsoft in grado di trasformare, pulire e modellare i dati provenienti da diverse fonti senza la necessità di scrivere lunghi script SQL.

In termini di sicurezza, Dataverse propone un modello flessibile, in cui gli utenti possono assumere permessi di accesso ai dati differenti, a seconda del ruolo e delle necessità specifiche, consentendo una gestione fine dei privilegi e delle informazioni sensibili.

### 2.1.4 Connettori

Al termine di questa sezione vengono illustrati brevemente dei componenti fondamentali per lo sviluppo di sistemi complessi in cui tanti servizi diversi tra loro interagiscono armoniosamente, trasferendosi informazioni. I connettori rappresentano, infatti, quegli elementi chiave in grado di connettere le piattaforme della Power Platform con servizi Microsoft o di terze parti. Esempio di connettori rilevanti già disponibili sono quelli che permettono l'integrazione di app (Canvas o Model-Driven) e flussi con Dropbox, Office 365, Teams, SharePoint, servizi Azure e molti altri ancora. Tra i connettori predefiniti si distinguono quelli standard, inclusi nella licenza base, da quelli premium, ovvero quelli che richiedono licenze aggiuntive. La vasta collezione di connettori built-in non è talvolta sufficiente a coprire il fabbisogno dello sviluppatore che intende integrare servizi meno comuni o di nicchia. A tal proposito, connettori custom sono stati introdotti per servizi non supportati nativamente.

L'utilizzo di connettori diventa, dunque, necessario per scambiare dati in tempo reale e migliorare l'efficienza operativa senza necessità di sviluppo complesso.

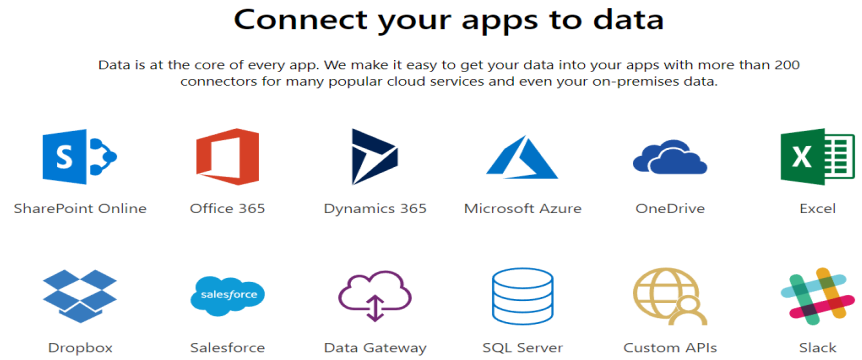


**Figura 2.2:** Servizi più comuni integrabili attraverso connettori

## 2.2 Dynamics 365

A partire dal 2016, Microsoft ha intrapreso un percorso strategico volto ad aumentare la **Business Agility**, ossia la capacità di un'organizzazione di adattarsi

al cambiamento e reinventare i propri processi [11]. Uno degli ostacoli principali che si contrapponeva a questa evoluzione era rappresentato dalla presenza, ancora radicata, di una serie di applicazioni aziendali mal integrate e gestite in modo isolato, rendendo difficile la condivisione delle informazioni. La necessità di una visione unificata sul business ha dato origine all'idea di avvicinare tutti questi software, facendoli, di fatto, convergere in un'unica piattaforma, così da ridurre il tempo di ricerca di un dato su sistemi diversi, portando a una migliore comprensione dell'azienda e accelerando il processo innovativo.

Dynamics 365 nasce, dunque, proprio con lo scopo di supportare le aziende moderne nel gestire i propri dati in modo centralizzato, concedendo loro un panorama a 360 gradi e, in tempo reale, del proprio ambiente lavorativo. Esso rappresenta una suite di applicazioni aziendali avanzate, basata su cloud e progettata per ottimizzare la gestione di vendite, marketing, assistenza clienti, operazioni e molto altro. Numerose aziende, indipendentemente dalle loro dimensioni, possono trarre vantaggio dall'uso di questo applicativo. Dynamics 365 fornisce, infatti, servizi per soddisfare le esigenze di qualsiasi realtà, che si tratti di PMI o multinazionali [12].


**Figura 2.3:** I moduli di Microsoft Dynamics 365

La piattaforma integra le funzionalità del **CRM (Customer Relationship Management)**, focalizzate sulla gestione e fidelizzazione dei clienti, con le soluzioni **ERP (Enterprise Resource Planning)**, indispensabili per il coordinamento

delle varie risorse operative e dell'intero ecosistema aziendale, dalla contabilità alla supply chain.

Pur essendo adesso una piattaforma basata sul cloud, Dynamics 365 ha origine come soluzione on-premise per la gestione di CRM e ERP. Ancora oggi, infatti, vengono offerte versioni installabili per alcuni moduli, permettendo alle aziende di scegliere l'opzione più adatta alle proprie esigenze. Nonostante questa possibilità, l'adozione di soluzioni installate localmente è sempre più in declino, poiché limita l'accesso ai benefici offerti dal cloud come aggiornamenti automatici e connessione immediata ad altri strumenti Microsoft. Per sfruttare, dunque, appieno le potenzialità della rete, sempre più organizzazioni si affidano al cloud, il quale assicura anche costi più bassi e consente loro di disinteressarsi della gestione e manutenzione hardware.

Oltre alle caratteristiche precedentemente descritte, desidero ora approfondire tre ulteriori aspetti che costituiscono i veri punti di forza di Dynamics 365:

- *Scalabilità* - Grazie alla sua architettura, Dynamics 365 consente di aggiungere o rimuovere funzionalità in base alla crescita del business, il che lo rende altamente flessibile. La piattaforma viene concepita come una serie di applicazioni plug-and-play; di fatto, l'utente può sentirsi libero di usare ciò che vuole, quando lo vuole e pagare soltanto in base all'effettivo consumo. Attraverso **Microsoft AppSource**, il marketplace ufficiale delle business app Dynamics, si possono infine acquistare estensioni e integrarle all'interno del proprio sistema grazie all'unicità del modello dati sottostante.

- *Produttività* - L'integrazione con Microsoft 365, Power BI, Power Automate, Power App e Azure rappresenta un significativo valore aggiunto per il sistema. La possibilità di gestire diversi aspetti di uno stesso compito in un unico ambiente non riduce soltanto il tempo di esecuzione del task stesso, ma abbatte sensibilmente il margine di errore e il rischio di duplicazione dei documenti. Dynamics 365 è inoltre accessibile su qualsiasi tipo di dispositivo, garantendo un'esperienza utente ancora più fluida e dinamica.

- *Intelligenza* - L'intelligenza Artificiale è un elemento trasversale che avvolge l'intero pacchetto di soluzioni Dynamics 365. Grazie agli algoritmi predittivi messi a disposizione dalla piattaforma, le aziende possono anticipare eventuali ritardi nelle consegne o prevedere futuri movimenti contabili, migliorando così la pianificazione operativa. Allo stesso tempo, algoritmi di apprendimento automatico (machine learning) possono imparare dall'esperienza e raffinarsi ogni giorno sempre di più, guidando in modo proattivo i dipendenti verso risultati di business ottimali.

In conclusione, Dynamics 365 si presenta come una soluzione potente e versatile per le aziende, in grado di affrontare le sfide legate all'integrazione dei dati, alla gestione della crescita e all'ottimizzazione dei processi.

## 2.3 Servizi Azure

Microsoft Azure è una piattaforma di cloud computing che include oltre 200 servizi per supportare il processo di sviluppo, distribuzione e mantenimento di applicazioni. Essa copre numerosi ambiti, anche molto diversi tra loro, come analisi dei dati, networking, archiviazione, sicurezza e soluzioni basate sull'intelligenza artificiale. La sua ampia diffusione nel mercato attuale è giustificata da una moltitudine di vantaggi che spaziano dall'elevata scalabilità dei suoi servizi alla semplicità di integrazione con software esistenti, dalla potenza computazionale disponibile alla riduzione dei costi di gestione, fino agli avanzati standard di sicurezza e conformità offerti, garantendo un'infrastruttura affidabile e performante.

Azure propone quattro principali modelli di cloud computing, permettendo così di soddisfare esigenze differenti:

- *Infrastructure as a Service (Iaas)* - Vengono fornite risorse all'utente come storage, rete e archiviazione. Questa modalità di erogazione consente alle aziende di mantenere esternamente l'infrastruttura IT, evitando così la manutenzione fisica di tali risorse (es. Azure VM o Azure Storage).

- *Platform as a Service (Paas)* - Con Paas le aziende beneficiano, oltre che dell'hardware, anche di piattaforme complete messe a disposizione da Azure per lo sviluppo, l'esecuzione e la distribuzione di applicazioni (es. Azure App Service).

- *Software as a Service (Saas)* - Modalità che include tutti quei servizi che offrono accesso al software nel cloud. Utilizzare un'applicazione tramite internet rimuove, di fatto, la necessità per le aziende di installare e mantenere software localmente (es. Azure DevOps).

- *Serverless* - Modello in cui il codice viene eseguito senza che l'utente debba gestire server o infrastruttura. Le risorse si attivano solo quando necessario (es. Azure Functions).

In Figura 2.4 viene illustrata una rappresentazione delle principali categorie di servizi offerti da Microsoft Azure. Per lo sviluppo di questo progetto di Tesi, sono stati analizzati e utilizzati servizi appartenenti a tre di queste macroaree: **Azure Storage**, **Azure Compute** e **Azure AI & ML**, che verranno approfonditi nei paragrafi successivi.

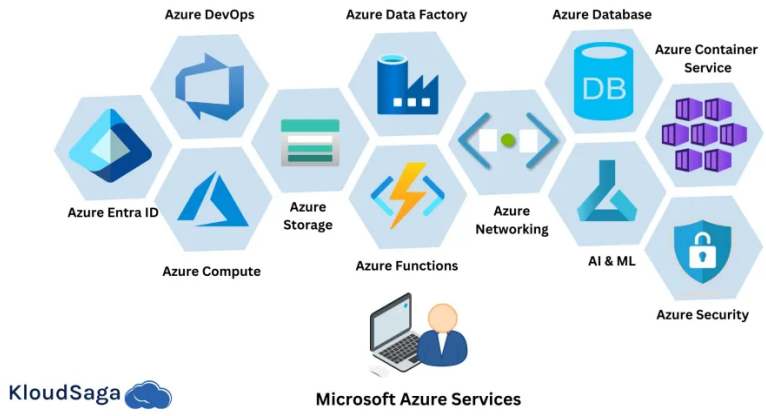


**Figura 2.4:** Panoramica delle principali famiglie di servizi Azure

### 2.3.1 Azure Storage Services

Azure Storage è la piattaforma Microsoft progettata per l'archiviazione sicura, scalabile e altamente disponibile di dati strutturati e non strutturati. Gli oggetti di Azure Storage sono accessibili attraverso un'API REST o mediante librerie client specifiche, integrabili facilmente con applicazioni sviluppate in .NET, Java, Python, JavaScript, C++ e Go. Il portale Azure fornisce, inoltre, un'interfaccia grafica intuitiva in cui gli utenti possono creare, caricare, scaricare e organizzare i dati, configurare le proprietà di archiviazione e monitorare l'utilizzo delle risorse.

Azure Storage offre quattro principali soluzioni di archiviazione tra cui: *Blob Storage* per la gestione di dati non strutturati, *Table Storage* per l'archiviazione di tabelle NoSQL e particolarmente adatto per scenari di Big Data, *File Storage* per la condivisione di file in rete e infine *Queue Storage* per quanto riguarda l'utilizzo di code, ideali in applicazioni distribuite che intendono scambiarsi messaggi in modo asincrono.


**Figura 2.5:** Tipologie di archiviazione in Azure Storage

#### Azure Blob Storage

Azure Blob Storage è uno dei servizi di archiviazione offerti da Azure Storage e consente il salvataggio di Blob (Binary Large Objects), ovvero particolari tipi di dati che possono essere archiviati in formato binario come video, immagini, documenti o qualsiasi altro contenuto non strutturato. Tramite l'interfaccia Azure è possibile visualizzare in modo semplice e immediato i contenitori generati, la loro gerarchia e i relativi contenuti. La piattaforma offre diverse opzioni di accesso ai dati, come Hot, Cool e Archive tiers, livelli che ottimizzano i costi e le prestazioni in base alla frequenza di utilizzo. Azure Blob Storage è dunque ideale per scenari che richiedono l'archiviazione di grandi volumi di dati, supportando anche operazioni di backup, disaster recovery e archiviazione a lungo termine.

### 2.3.2 Azure Compute Services

Azure Compute Services è una suite di servizi offerti da Microsoft Azure per eseguire carichi di lavoro e applicazioni nel cloud. Questa famiglia di servizi è particolarmente utile per creare e gestire server virtuali attraverso *Azure Virtual Machines*, orchestrare container grazie a *Azure Kubernetes*, eseguire codice serverless per mezzo di *Azure Functions* o distribuire applicazioni di qualsiasi dimensione con *Azure App*. Azure Compute offre, in generale, la possibilità di gestire applicazioni che richiedono risorse computazionali elastiche e flessibili nel cloud.

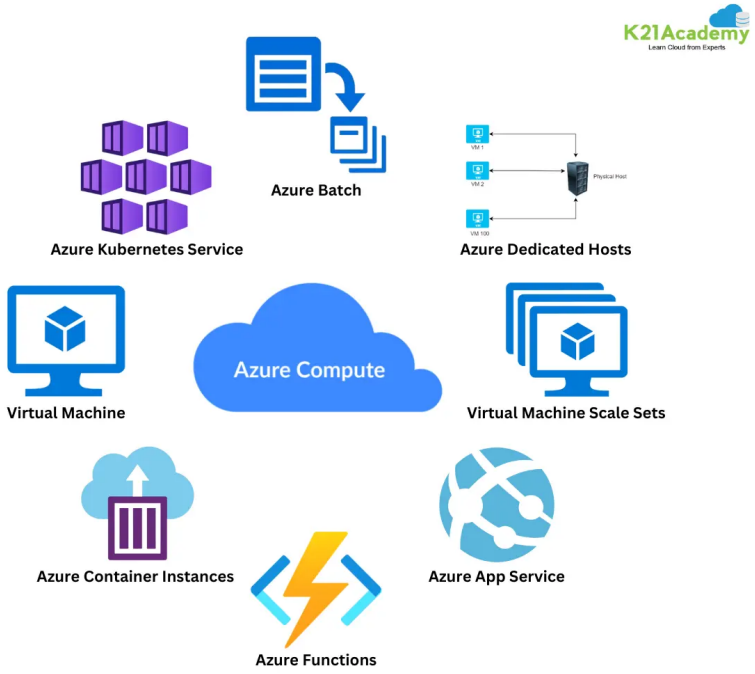


**Figura 2.6:** Principali servizi offerti da Azure Compute

### Azure App Service

Azure App Service è una soluzione PaaS offerta all'interno di Azure Compute, progettata per consentire agli sviluppatori di creare e distribuire velocemente web server e applicazioni web e/o mobile. Grazie a questa piattaforma, gli utenti possono implementare applicazioni senza doversi occupare della gestione dell'infrastruttura sottostante. Uno dei principali vantaggi offerti da Azure App è la scalabilità automatica, ovvero la capacità di aumentare e diminuire il numero di risorse coinvolte in base al traffico e assicurare, dunque, prestazioni elevate anche in presenza di picchi di richieste. Azure App offre, inoltre, la possibilità di distribuire applicazioni in modo contiguo, permettendo di rilasciare aggiornamenti senza interruzioni significative nel servizio. Questa funzionalità riduce al minimo i tempi di inattività, assicurando un'esperienza utente costante anche durante il rilascio di nuove versioni. Questa tecnologia fornisce, infine, strumenti di monitoraggio e diagnostica come file di log per identificare velocemente eventuali errori e dashboard per l'analisi delle prestazioni, meccanismi di sicurezza avanzata, connettività immediata con altri servizi Azure e supporto per l'integrazione con sistemi di gestione del codice, come GitHub e BitBucket.

### 2.3.3 Azure Cognitive Services

Durante lo svolgimento del mio lavoro di Tesi, l'intelligenza artificiale ha giocato un ruolo fondamentale, costituendo l'elemento chiave che conferisce valore alla soluzione proposta in ottica business. Con lo scopo di individuare un servizio che incontrasse le necessità progettuali sono state analizzate due delle tecnologie offerte da **Azure AI & ML**, come *Azure Cognitive Services* e *Azure OpenAI*.

Azure Cognitive Services è un pacchetto di API preconfigurate, capaci di integrare funzionalità di AI nelle applicazioni senza richiedere competenze avanzate di Machine Learning. Nato nel 2015 sotto il nome di *Project Oxford*, il prodotto inizialmente includeva un insieme di tecnologie intelligenti, offrendo soluzioni in grado di sfruttare al massimo il potenziale offerto dal Cognitive Computing, come il riconoscimento facciale, il riconoscimento vocale, la classificazione delle immagini e la comprensione del linguaggio naturale [13]. Ad oggi, in seguito all'ampliamento del prodotto, numerose aziende hanno iniziato ad adottare i servizi cognitivi di Microsoft all'interno dei loro software. Airbus, per esempio, utilizza il rilevamento delle anomalie per monitorare l'integrità degli aeromobili, mentre la BBC utilizza questi servizi per dialogare con i clienti attraverso un assistente vocale personalizzato [14]. Prima di approfondire la macrocategoria Vision, di interesse per il sistema sviluppato, è importante rispondere a una domanda fondamentale: "Cosa si intende per Cognitive Computing e quale obiettivo si propone di raggiungere?"

Il Cognitive Computing è un ramo dell'intelligenza artificiale che mira a simulare il modo in cui le persone pensano, apprendono e risolvono problemi. Oltre alle tecnologie precedentemente menzionate, questo ambito coinvolge anche l'elaborazione dei segnali, la realtà virtuale e le reti neurali, strumenti essenziali per riprodurre le capacità di ragionamento e la comprensione, tipiche dell'essere umano. Azure Cognitive Services sfrutta proprio questi principi per dar vita ai suoi numerosi servizi, applicabili in diversi contesti aziendali e catalogati secondo lo schema di Figura 2.7.

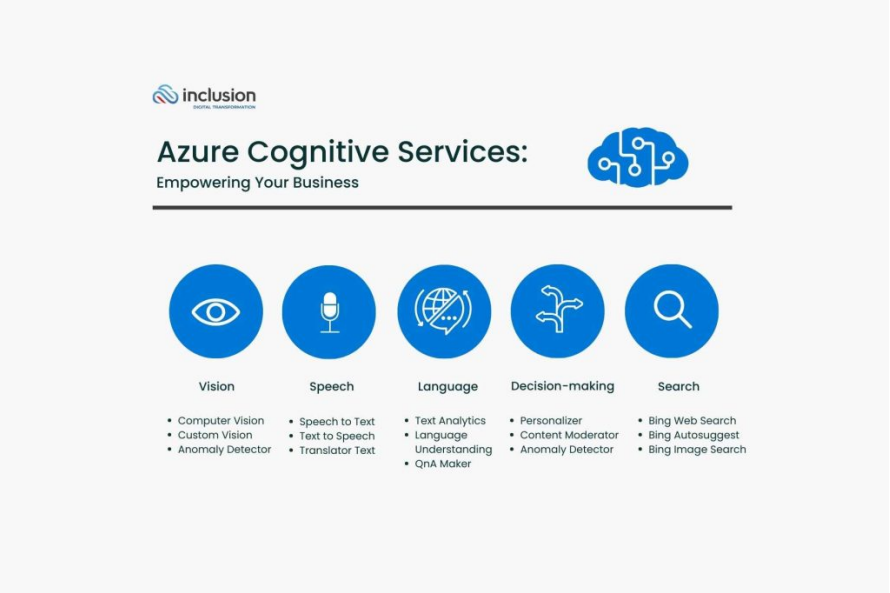


**Figura 2.7:** Le 5 categorie di servizi offerti da Azure Cognitive Services

### Azure Computer Vision

Azure Computer Vision è uno dei servizi offerti da Azure Cognitive Services nell'ambito della visione artificiale. Questa piattaforma mette a disposizione un insieme di algoritmi avanzati per l'elaborazione di immagini e video, offrendo all'utente la capacità di analizzarli ed estrarre informazioni attraverso API pronte all'uso. La tecnologia alla base sfrutta modelli di Machine Learning addestrati su una quantità massiva di immagini nel cloud. Gli algoritmi in gioco riconoscono patterns visivi specifici e li impiegano per interpretare il contenuto di nuove immagini. Una logica simile fa di Azure Computer Vision un applicativo che si presta molto bene a scenari in cui l'analisi riguarda oggetti comuni e generici, con un livello di personalizzazione limitato. Le sue funzionalità principali includono il riconoscimento di oggetti e scene, la lettura di un testo, l'analisi del contenuto visivo (come colore, volti e categorie) e la generazione di brevi didascalie a partire da immagini in ingresso. Questo servizio è ampiamente utilizzato, specialmente in settori come la gestione documentale e l'e-commerce. Azure Computer Vision è, inoltre, in grado di esaminare video, rilevando e analizzando movimenti all'interno di uno spazio, il che lo rende particolarmente utile anche in applicazioni di sorveglianza.

### Azure Custom Vision

Come Computer Vision, anche Azure Custom Vision fa parte delle piattaforme offerte da Microsoft nel contesto delle soluzioni cognitive. Il servizio, tuttavia, si

differenzia dal primo in quanto consente la generazione di modelli personalizzati di visione artificiale, piuttosto che limitarsi alla concessione di soli modelli preesistenti. Con Azure Custom Vision gli utenti hanno la possibilità di addestrare i propri modelli, utilizzando dataset di immagini forniti direttamente da loro stessi. Gli utenti possono, pertanto, etichettare le immagini del training set, assegnando loro il nome di una categoria o associando un tag agli oggetti presenti all'interno di esse. Gli utenti, inoltre, hanno la possibilità di addestrare il modello in base alle proprie esigenze, regolando i parametri e applicando tecniche di fine-tuning per ottimizzare le prestazioni in base allo specifico contesto di utilizzo. La creazione e l'addestramento di modelli è, dunque, semplice e richiede pochi passaggi, permettendo di migliorarli continuamente attraverso feedback e nuovi dati. È questa caratteristica a rendere la tecnologia particolarmente attraente per chi ha esigenze particolari, soprattutto per coloro che necessitano di uno strumento tarato per il proprio dominio di applicazione. Attraverso questo servizio è possibile sviluppare modelli sia per la classificazione delle immagini, sia per il riconoscimento degli oggetti al loro interno. Custom Vision consente, infine, di valutare l'accuratezza del modello, testandolo direttamente sulle stesse immagini utilizzate per l'addestramento. In questo modo, è possibile ottenere una prima indicazione delle sue prestazioni senza la necessità di caricare immagini di test aggiuntive.

### 2.3.4 Azure OpenAI

I servizi Azure che ricadono all'interno dei Cognitive Services sono soluzioni ampiamente utilizzate da imprese di ogni forma e dimensione, le quali si affidano, sempre con più convinzione, ai benefici dell'**intelligenza artificiale tradizionale**. Negli ultimi anni, tuttavia, una nuova tecnologia si sta affermando con velocità di adozione e diffusione nemmeno ipotizzabili fino a qualche anno fa. L'**intelligenza artificiale generativa** è il nuovo trend del momento e si discosta da quella tradizionale poiché non si limita ad analizzare e interpretare dati ma, basandosi su algoritmi avanzati di Deep Learning, è anche in grado di creare nuovi contenuti in modo innovativo, "imparando" dalle interazioni con l'uomo e con l'ambiente circostante e migliorando continuamente. Un esempio emblematico della sua diffusione è **ChatGPT**, un'applicazione basata su un modello GPT sviluppato da OpenAI, che ha raggiunto 100 milioni di utenti in appena due mesi, un tempo significativamente inferiore rispetto ad altre tecnologie di largo consumo (16 anni per l'iPhone, 75 anni per il primo telefono) [15]. Questo dato evidenzia l'impatto rivoluzionario dell'AI generativa e crea il contesto per la nascita di Azure OpenAI, il servizio che rappresenta la sinergia tra la piattaforma cloud leader di Microsoft, Azure, e le tecnologie di AI generativa sviluppate da OpenAI.

Azure OpenAI è una piattaforma che offre accesso a modelli avanzati di OpenAI, tra cui GPT, Codex e DALL-E per attività come la comprensione del linguaggio

naturale, l'analisi dei sentimenti e la traduzione linguistica in tempo reale. Al cuore del servizio ci sono i modelli GPT (Generative Pre-trained Transformers), reti neurali avanzate, basate sull'architettura Transformer. GPT-4, per esempio, è uno di questi modelli, pre-addestrato su un'enorme quantità di dati testuali e affinato per diverse applicazioni come chatbot e sintesi di documenti. Un'altra caratteristica di GPT-4 è la sua capacità di comprendere e generare codice in diversi linguaggi di programmazione a partire da istruzioni espresse in linguaggio naturale. Ciò rende il modello estremamente utile in contesti in cui si vuole fornire supporto allo sviluppo software, rendendo più veloce la codifica. Nel campo dell'analisi visiva, OpenAI propone GPT-4 Vision, modello multimodale in grado di elaborare sia testo che immagini. Gli utenti possono, pertanto, caricare immagini e ingaggiare conversazioni col modello per estrapolare informazioni rilevanti presenti all'interno del contenuto visivo. Questa tecnologia trova applicazione in contesti di qualsiasi tipo, dall'analisi di grafici e documenti, al controllo qualità, alla diagnosi automatizzata e persino all'assistenza in cucina, suggerendo ricette in base agli ingredienti presenti in un frigorifero. Azure OpenAI integra, inoltre, i modelli DALL-E, progettati per generare immagini a partire da descrizioni testuali, combinando concetti in modo creativo e per consentire la modifica di immagini esistenti tramite funzionalità di editing. La piattaforma fornisce, infine, numerosi altri modelli, ognuno con le proprie peculiarità, capaci di elaborare diversi tipi di input che vanno oltre il solo testo e le immagini.

In ottica futura, Azure OpenAI si candida a diventare uno dei servizi più innovativi dell'intero ecosistema Azure, evidenziando il ruolo sempre più centrale dell'intelligenza artificiale nella vita quotidiana e all'interno dei processi aziendali.

# Capitolo 3

# Progettazione

In questo capitolo si esplora la fase cruciale della Progettazione. Questo step riveste un'importanza strategica nella realizzazione di un prodotto capace di rispondere efficacemente alle esigenze di un pubblico specifico, garantendo al contempo il rispetto dei vincoli imposti dal contesto operativo.

Il capitolo comincia presentando lo stato dell'arte attuale, illustrando le diverse fasi che hanno portato alla decisione di focalizzarsi sull'analisi della fase di ispezione e all'individuazione delle problematiche da affrontare.

La sezione successiva si prefigge come obiettivo l'individuazione di caratteristiche che il sistema mira a raggiungere, definendo quindi le esigenze e le aspettative degli utenti in gioco e degli eventuali stakeholders. In questa fase iniziale si analizzano le problematiche alla base e, attraverso un'attenta valutazione del contesto e attività di brainstorming, si perviene alla formulazione di requisiti, catalogandoli formalmente in funzionali e non funzionali.

Una volta chiarite le specifiche del sistema, si passa all'effettiva fase di design in cui viene abbozzata una proposta architetturale di partenza. Dallo schema generato, indipendente dalle tecnologie e da eventuali vincoli implementativi, si comincia a ragionare dando uno sguardo più ampio alle tecnologie Microsoft disponibili. Questo passaggio rappresenta un momento decisivo, poiché richiede l'adozione di scelte mirate sulla base dei vincoli progettuali precedentemente riconosciuti e delle implicazioni tecniche derivanti dall'integrazione con Azure e piattaforme Low-Code.

A valle delle decisioni progettuali assunte, vengono elaborati e presentati diversi diagrammi che offrono una rappresentazione visiva della struttura del sistema. Attraverso il Component Diagram vengono illustrati i principali componenti software del sistema finale, le loro relazioni e dipendenze, fornendo una visione chiara dell'organizzazione interna. Viene, inoltre, delineato un modello dati che astrae le entità coinvolte, la connessione fra esse e le relative cardinalità. Il capitolo si conclude, infine, con la presentazione di mockup, prototipi delle schermate principali dell'app mobile.

## 3.1 Analisi dello stato dell'arte

Il servizio offerto dalle agenzie di autonoleggio consente ai clienti di affittare un veicolo per periodi di tempo che possono andare da poche ore fino a durate più estese. A partire, dunque, da una prenotazione, il cliente si reca al punto di ritiro ed esegue, insieme a un operatore, la fase di check-in costituita generalmente da alcune attività essenziali: verifica dei documenti del cliente, illustrazione dei termini di noleggio (inclusi aspetti come assicurazione e gestione del carburante) e ispezione del veicolo, con lo scopo di evidenziare eventuali danni preesistenti. Al termine di questo passaggio, si procede infine con la compilazione di un modulo che documenta i risultati dell'ispezione, la consegna delle chiavi e l'inizio dell'effettivo noleggio. Alla restituzione del veicolo, il processo segue una dinamica simile, con controlli speculari a quelli eseguiti al momento della consegna. Nel caso in cui vengano riscontrati nuovi danni, un chilometraggio eccedente o un livello di carburante inferiore rispetto a quanto stabilito, il cliente è soggetto ad addebiti secondo le tariffe aziendali. Un normale processo di autonoleggio si conclude, infine, con la chiusura del contratto e l'eventuale emissione della fattura finale. Per completezza, è importante sottolineare come tutto questo rappresenti l'iter classico di un **noleggio tradizionale**.

Negli ultimi anni, il **self-service** ha rivoluzionato numerosi settori, incluso quello dell'autonoleggio, trasformando profondamente le modalità di gestione e fruizione del servizio. I vantaggi di questa modalità sono molteplici e particolarmente evidenti, soprattutto dal punto di vista dei clienti. Nel modello tradizionale, le operazioni di ritiro e rilascio del veicolo sono vincolate alla disponibilità dell'operatore e agli orari di apertura del punto di noleggio. Il cliente, quindi, deve attenersi a tempi prestabiliti, un aspetto che potrebbe rendere il servizio meno flessibile. Con il self-service, invece, il cliente può ritirare e riconsegnare il veicolo in aree dedicate, senza essere vincolato alla presenza fisica di un operatore. Pur essendo prevista una data e un'ora di rilascio, l'assenza di personale in loco elimina l'impossibilità di restituire il mezzo in caso di ritardo. Parallelamente, anche le aziende del settore stanno riconoscendo il potenziale di questa innovazione, adottando sempre più frequentemente soluzioni di noleggio self-service. Questo cambiamento non solo risponde alle nuove esigenze dei consumatori, sempre più orientati verso esperienze digitali rapide ed efficienti, ma rappresenta anche un'opportunità per le imprese di ridurre i costi di gestione e migliorare l'esperienza complessiva del servizio. L'uso di contratti digitali e firme elettroniche, per esempio, riduce i costi di stampa, archiviazione e gestione amministrativa della documentazione. Allo stesso modo, un servizio self-service consente di limitare la necessità di uffici fisici presso le sedi di noleggio, portando a un risparmio su affitti e utenze.

In uno scenario del genere, l'attività dell'operatore non viene eliminata, ma soltanto spostata nel tempo: l'ispezione del veicolo viene infatti effettuata in

background dall'operatore, prima del check-in e dopo il check-out del cliente. Le modalità con cui avviene la verifica dei danni sono le stesse in entrambe le tipologie di servizio. Ciò che cambia è, invece, la gestione anticipata di tutta la documentazione necessaria e la consegna, completamente digitalizzata, delle chiavi tramite cassette di sicurezza smart, totem automatizzati o tecnologie keyless come Bluetooth o NFC, nel caso di veicoli predisposti.

### Processo di approfondimento

Per analizzare in modo più approfondito le dinamiche che caratterizzano le principali fasi di un processo di noleggio auto e individuarne le criticità più ricorrenti, è stata condotta un'attività di ricerca basata su esperienze pregresse maturate in prima persona come utente del servizio, integrata da un'analisi documentale di fonti pubblicamente disponibili (siti web, report informativi, guide operative) relative al funzionamento di specifici modelli di business nel settore. Ulteriori spunti di riflessione sono emersi grazie al contributo della BU Automotive di Cluster DCX, che ha messo a disposizione documentazione tecnica relativa a processi consolidati in ambito automotive. Pur non riferendosi direttamente al contesto dell'autonoleggio, tali materiali hanno offerto esempi di problematiche e soluzioni applicabili anche in scenari affini, caratterizzati da attività analoghe in termini di interazione tra utente e operatore e verifica dello stato di beni mobili. L'insieme di queste fonti mi ha permesso di estrapolare ulteriori informazioni che vengono presentate di seguito:

- La fase di ispezione è estremamente delicata, poiché richiede un controllo accurato da parte dell'operatore. Tuttavia, la stanchezza, soprattutto nelle ore finali del turno, può influire negativamente sulla concentrazione, rendendo più probabili sviste o valutazioni imprecise. La distrazione e la superficialità sono, infine, fattori dell'essere umano che possono compromettere l'accuratezza dei controlli, aumentando il rischio di errori e riducendo l'affidabilità complessiva del servizio offerto.

- Le fasce orarie di maggiore affluenza, come la mattina presto e il tardo pomeriggio, vedono un numero elevato di check-in e check-out. Questo può portare a tempi di attesa variabili tra i 20 e i 50 minuti nei momenti di picco, un aspetto poco ottimale per l'esperienza cliente.

- Un elemento di rallentamento è la consultazione della documentazione da parte dell'operatore, necessaria per categorizzare eventuali danni e calcolare i costi da addebitare. La necessità di incrociare informazioni provenienti da fonti diverse, come policy aziendali, listini prezzi per le riparazioni e standard di valutazione del danno, aumenta il rischio di discrepanze e interpretazioni soggettive, portando a possibili contestazioni da parte dei clienti. Questa

dispersione di informazioni allunga i tempi operativi e rende il processo più macchinoso e meno standardizzabile.

- La compilazione del modulo di ispezione del veicolo è un'operazione ripetitiva che rallenta ulteriormente il processo, senza apportare un valore aggiunto significativo rispetto alla semplice registrazione digitale dei danni.

- Ogni azienda di autonoleggio adotta procedure leggermente diverse, con variazioni nei dettagli operativi del check-in e del check-out. Questa eterogeneità complica l'uniformazione dei processi e può rendere difficile l'implementazione di soluzioni digitali standardizzate.

## 3.2 Requisiti

Terminata questa prima fase di studio, si è iniziato a riflettere sulle tipologie di problematiche individuate. Da queste emerge chiaramente la necessità di automatizzare il momento di ispezione del veicolo, rappresentando il vero collo di bottiglia dei flussi di check-in e check-out. Le tecnologie odierne rappresentano l'opportunità di miglioramento ricercata, offrendo soluzioni digitali capaci di affrontare le criticità presenti in questa specifica fase. L'ispezione manuale rappresenta, infatti, un'operazione troppo lenta per l'azienda rispetto al resto delle attività citate, contribuendo ad allungare i tempi di attesa dei clienti e incidendo negativamente sull'esperienza utente complessiva. Il totale affidamento alla capacità dell'operatore di intercettare eventuali danni costituisce, inoltre, il secondo grande punto critico, non essendo attualmente presenti strumenti di supporto adeguati per assistere l'addetto in questa operazione. Ciò che è importante evidenziare è che l'affidabilità delle ispezioni potrebbe essere migliorata semplicemente dedicando più tempo a questa fase. Per garantire un'ispezione accettabile del veicolo, sarebbero necessari almeno 3 minuti per ogni controllo più altri 10 per la compilazione del modulo di ispezione, tempo che, tuttavia, non è spesso disponibile a causa dell'elevato numero di clienti da gestire. La necessità di servire un alto volume di utenti in tempi ridotti porta, infatti, a controlli sommari, aumentando il rischio di errori nella rilevazione dei danni. Questi errori possono tradursi in perdite economiche significative per l'azienda, a seconda dell'entità del danno non rilevato. Nonostante ciò, allungare i tempi di ispezione non rappresenterebbe una soluzione ottimale, poiché andrebbe a contrastare l'esigenza di rapidità nel servizio, aggravando il problema iniziale anziché risolverlo. Di conseguenza, diventa fondamentale individuare un approccio che garantisca controlli accurati senza compromettere la fluidità delle operazioni. Infine, la mancanza di un sistema centralizzato in cui raccogliere e condividere i dati relativi a clienti, veicoli, noleggi e danni rappresenta un ostacolo significativo per l'efficienza operativa. L'implementazione di una piattaforma unica permetterebbe

all'azienda di migliorare la gestione e la condivisione delle informazioni tra i vari team.

Per affrontare le criticità emerse dall'analisi dell'attuale stato dell'arte, il progetto ha portato allo sviluppo di un sistema in grado di automatizzare l'ispezione dei veicoli durante le fasi di check-in e check-out, sfruttando le potenzialità della Power Platform di Microsoft e tecnologie avanzate come i modelli di AI offerti da Azure.

L'intero percorso progettuale è stato guidato dalle problematiche individuate in fase di analisi, che hanno permesso di delineare con chiarezza le necessità del sistema. Ogni decisione tecnica e implementativa è stata quindi presa sulla base di questi spunti iniziali, traducendoli in funzionalità concrete capaci di rispondere efficacemente alle esigenze operative dell'autonoleggio. Per garantire una soluzione robusta e ben strutturata, queste caratteristiche sono state analizzate, organizzate e formalizzate all'interno di un processo metodico di definizione dei requisiti. Questo passaggio è fondamentale per tradurre le esigenze operative in specifiche tecniche chiare e attuabili. In particolare, i requisiti individuati sono stati catalogati in **requisiti funzionali** e **requisiti non funzionali**. I primi delineano i comportamenti del sistema, le funzionalità esposte e le interazioni tra gli attori in gioco. I requisiti non funzionali, invece, si concentrano su aspetti sia qualitativi che quantitativi del sistema, affrontando attributi come sicurezza, performance e scalabilità. Di seguito vengono presentati i risultati di questa fase di analisi:

### Requisiti Funzionali

- *Gestione dei clienti* - Il sistema deve permettere all'azienda di monitorare le informazioni relative ai clienti, offrendo la possibilità di modificare o cancellare i dati di ciascuno di essi.

- *Gestione dei veicoli* - Il sistema deve essere in grado di gestire le informazioni riguardanti il parco veicoli dell'azienda, consentendo la creazione, modifica o cancellazione di un veicolo.

- *Gestione dei noleggi* - Il sistema deve consentire all'azienda la possibilità di registrare un nuovo noleggio, eliminarlo o modificarlo secondo le proprie esigenze.

- *Gestione delle immagini* - Il sistema deve fornire la possibilità ai clienti di scattare delle immagini e di poterle caricare sulla piattaforma. Si deve, inoltre, concedere all'azienda la possibilità di reperire tali immagini in qualsiasi momento, tenendo traccia del noleggio dal quale deriva la foto e la fase precisa a cui l'immagine è stata associata (check-in o check-out).

- *Capacità di analisi delle immagini* - Si tratta della funzionalità core del sistema, in grado di fornire valore alla proposta di Tesi. Il servizio deve essere in grado

di esaminare le immagini scattate dagli utenti, riconoscere il tipo di soggetto nella foto e indagare, nel caso esso sia un veicolo, sull'eventuale presenza di danni.

- *Gestione dei danni* - Il sistema deve consentire all'azienda di rintracciare, in qualsiasi momento, i risultati dell'analisi. Questo permette di prendere decisioni basate sulle segnalazioni evidenziate e sul livello di accuratezza con cui il sistema ha identificato il danno.

**Requisiti Non Funzionali**

- *Performance*
  - *Velocità* - Il sistema deve velocizzare la fase di ispezione, riducendo i tempi di attesa per i clienti e migliorando l'efficienza complessiva dell'operazione di verifica e compilazione modulo di almeno il 50%.
  - *Affidabilità* - Il sistema deve garantire un grado di affidabilità sufficientemente elevato. Maggiore sarà il livello raggiunto, maggiore sarà il suo poteziale d'uso. Il sistema intende, infatti, essere uno strumento di supporto per l'addetto alle ispezioni, contribuendo alla riduzione del tasso di errori e all'aumento di precisione nell'identificazione dei danni. È importante sottolineare, tuttavia, che la soluzione proposta non mira a sostituire completamente il ruolo dell'operatore, ma a potenziarlo, rendendo i suoi controlli più efficaci.

- *Potenziale Commerciale* - La soluzione sviluppata non deve presentare specificità per un particolare caso d'uso. Ciò che si vuole ottenere è, piuttosto, una tecnologia versatile e personalizzabile a piacere, che garantisca configurabilità, modularità e punti, quindi, a raggiungere obiettivi di business elevati. Il sistema deve, inoltre, essere facilmente integrabile con applicazioni diverse qualora già presenti.

- *Scalabilità* - Il sistema deve essere progettato per gestire una grande quantità di richieste di analisi nell'unità di tempo. Allo stesso modo, il software deve essere capace di memorizzare una grande quantità di dati e di renderli accessibili a livello globale, garantendo la disponibilità delle informazioni in qualsiasi momento e da qualsiasi luogo.

- *Centralizzazione delle informazioni* - Il sistema deve fungere da punto di riferimento unico per tutti i team coinvolti, centralizzando le informazioni provenienti da fonti diverse e garantendo un ambiente condiviso e integrato. Questo consentirà a tutti i membri del team di accedere a dati aggiornati e

coerenti, migliorando la collaborazione e la comunicazione tra i diversi reparti e assicurando un flusso di lavoro più fluido ed efficiente.

- *Sicurezza* - Il sistema deve implementare robuste misure di sicurezza per proteggere i dati personali e aziendali e garantire l'accesso ai soli utenti autorizzati.

## 3.3 Strategie di Design dell'Architettura

Partendo da un modello architetturale iniziale, vengono approfondite due tematiche che hanno richiesto un'attenta valutazione durante la fase di design. Ogni aspetto racchiude al suo interno diverse proposte progettuali, di cui ciascuna rappresenta un possibile percorso per giungere a un'architettura finale coerente e funzionale. Le differenze tra le varie opzioni risiedono principalmente nell'approccio seguito: alcune strategie mirano ad abbattere i costi, altre a fornire scalabilità e performance elevate, altre ancora privilegiano la semplicità e l'utilizzo di codice ridotto. In generale, ogni decisione è stata guidata dalla volontà di rispettare i requisiti raccolti durante la fase di analisi, tenendo presente le tecnologie Low-Code di Cluster DCX come punto di partenza e optando spesso per approcci più sofisticati e complessi al fine di gestire il maggior numero possibile di criticità. L'idea è stata quella di non limitarsi alle funzionalità offerte dalla Power Platform, ma di esplorare anche i pro e i contro emergenti dall'integrazione con software di terze parti, ampliando così le possibilità di sviluppo.

Il primo argomento in cui è stato necessario addentrarsi riguarda il meccanismo di **salvataggio delle foto**. L'operazione di archiviazione è stata delegata generalmente a Dataverse, incaricato di memorizzare i dati testuali dell'applicazione essendo il servizio Microsoft di riferimento per la Power Platform. Le immagini, tuttavia, rappresentano un tipo di dato particolarmente oneroso in termini di spazio e gestione, il che ha portato a chiedersi se Dataverse fosse davvero adatto per questo compito o valesse la pena analizzare servizi alternativi, ottimizzati per l'archiviazione di file multimediali.

L'argomento successivo analizza, invece, il comportamento un'architettura che prevedesse la **presenza di un Web Server** come parte integrante del sistema. Anche questa scelta riveste un ruolo cruciale nell'orientare l'evoluzione del prodotto finale, influenzando tanto la struttura quanto il comportamento complessivo della soluzione. Sebbene arricchisca la complessità generale del sistema, i vantaggi generati dalla sua introduzione aprono interessanti prospettive e rendono questa possibilità particolarmente attraente.

L'analisi di tutte queste alternative ha permesso, dunque, di definire l'architettura ottimale, bilanciando benefici funzionali e compromessi tecnici derivanti dalla scelta di un componente o di un servizio piuttosto che un altro.

### 3.3.1 Fondamenti Architetturali

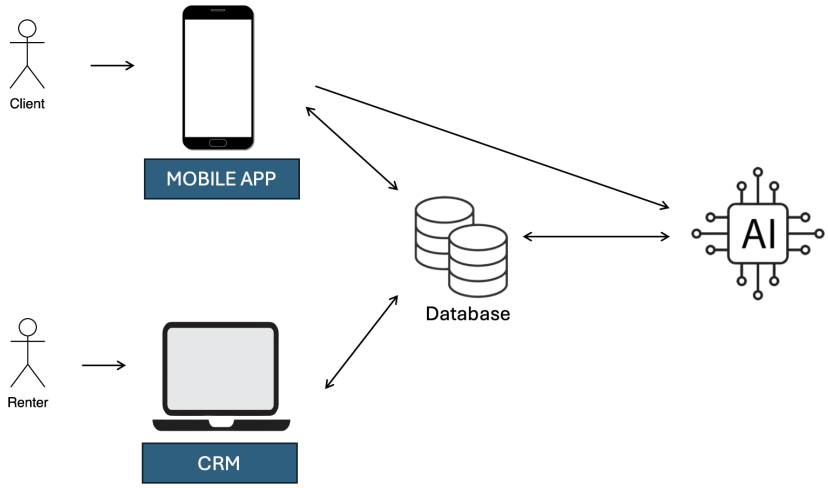


**Figura 3.1:** Diagramma Architetturale di base del sistema

In Figura 3.1 è illustrata la struttura iniziale del sistema così come concepita nella fase preliminare di progettazione. Questo schema rappresenta una versione semplificata dell'architettura finale, priva di dettagli specifici, ma utile per comprendere le scelte progettuali descritte nel paragrafo successivo. Tramite questo diagramma vengono evidenziati quindi i componenti che costituiscono lo scheletro della piattaforma, escludendo da questa rappresentazione elementi secondari come specifiche tecnologie o ottimizzazioni mirate a migliorare aspetti qualitativi del sistema.

Osservando la figura emerge subito una chiara distinzione tra due tipologie di attori: da un lato, i Client, ovvero gli utenti che desiderano usufruire del servizio di autonoleggio; dall'altro, i Renter, tutti coloro che, all'interno dell'azienda di autonoleggio, sono incaricati di gestire i noleggi o che, in generale, hanno accesso alle informazioni relative ad essi. Per i Client è stata prevista un'app mobile, poiché la cattura delle immagini del veicolo rappresenta la funzionalità core del processo, e l'utilizzo di un dispositivo mobile si rivela la soluzione più naturale ed efficace. Per i Renter, invece, un'applicazione CRM garantisce quella centralizzazione delle informazioni che costituisce uno dei punti chiave dell'analisi. Attraverso questo tipo di software i Renter avrebbero, infatti, la possibilità di monitorare agevolmente informazioni di vario tipo come quelle relative ai noleggi, ai clienti e ai veicoli.

Sia l'app mobile sia il CRM richiedono un database per l'archiviazione e il recupero delle informazioni necessarie al funzionamento del sistema. Per l'analisi delle immagini si è immaginato, invece, un modulo di intelligenza artificiale, il

quale, attivato al termine del caricamento delle foto da parte del Client, elabora le immagini e trasmette i risultati al database.

In questo momento non vengono considerati componenti aggiuntivi diversi da quelli appena presentati, poiché già questi risultano adeguati a soddisfare i requisiti di base. Nelle sezioni successive, tuttavia, l'architettura proposta verrà riesaminata alla luce delle tecnologie Microsoft disponibili, valutando eventuali ottimizzazioni e l'introduzione di nuovi componenti per migliorare le prestazioni e la configurabilità del sistema.

### 3.3.2 Salvataggio foto

Quando si tratta di archiviare immagini all'interno di un sistema basato su tecnologie Microsoft, Azure Blob Storage rappresenta spesso la scelta più vantaggiosa rispetto a Dataverse, sia in termini di costi che di scalabilità.

Analizzando più da vicino i piani tariffari del modello di pagamento a consumo (pay-as-you-go), vengono riportati in figura i prezzi relativi all'archiviazione dei dati, dai quali emerge chiaramente quanto sia estremamente oneroso salvare un elevato volume di immagini su Dataverse rispetto a una soluzione basata su Blob Storage. A parità di spazio utilizzato, il costo di archiviazione su Dataverse è circa 133 volte superiore rispetto a Blob Storage nella configurazione di accesso frequente, arrivando fino a 1200 volte maggiore nella modalità archivio. Tuttavia, è importante considerare che i due servizi sono progettati per scopi diversi: Azure Blob Storage è pensato per la gestione di grandi volumi di dati non strutturati, come immagini e documenti, mentre Dataverse è ottimizzato per la gestione di dati relazionali e transazionali, offrendo funzionalità avanzate per il modellamento dei dati e l'integrazione con altre applicazioni Microsoft.

**Prezzi per l'archiviazione dei dati con pagamento in base al consumo**

Tutti i prezzi si intendono per GB al mese.

| Prezzi per l'archiviazione dei dati con pagamento in base al consumo | Premium | Accesso frequente | Accesso sporadico | Accesso saltuario | Archivio |
|---|---|---|---|---|---|
| Primo 50 terabyte (TB)/mese | $0,15 per GB | $0,018 per GB | $0,01 per GB | $0,0036 per GB | $0,002 per GB |
| Successivo 450 TB/mese | $0,15 per GB | $0,0173 per GB | $0,01 per GB | $0,0036 per GB | $0,002 per GB |
| Oltre 500 TB/mese | $0,15 per GB | $0,0166 per GB | $0,01 per GB | $0,0036 per GB | $0,002 per GB |

**Figura 3.2:** Azure Blob Storage - Prezzi per l'archiviazione pay-as-you-go

| Dataverse Capacity/environment | Additional Increment | Pricing |
|---|---|---|
| Dataverse Database capacity | 1 GB | $48/month |
| Dataverse File capacity | 1 GB | $2.40/month |
| Dataverse Log capacity | 1 GB | $12/month |

**Figura 3.3:** Dataverse - Prezzi per l'archiviazione pay-as-you-go

Oltre al fattore economico, altre differenze rilevanti riguardano le performance, le funzionalità offerte e la scalabilità. Azure Blob Storage offre, infatti, una maggiore efficienza nella gestione dei file, permettendo l'accesso diretto tramite URL, e funzionalità avanzate come il versionamento e la replica geografica, rappresentando una tecnologia sicura e flessibile. Di contro, Dataverse presenta prestazioni inferiori per scenari di archiviazione massiva o accesso frequente a file multimediali.

Alla luce delle caratteristiche dei due servizi, dai costi e dalle performance osservate si è quindi pensato di includere Azure Blob Storage all'interno del progetto. Più precisamente si è deciso di salvare i metadati delle immagini su Dataverse in quanto informazioni strutturate, mentre di lasciare a Azure Blob Storage il compito di gestire il contenuto fisico delle immagini. Questo approccio consente quindi, oltre all'abbattimento dei costi, anche la riduzione del carico sulle risorse di Dataverse, evitando di appesantire il database con dati binari che non rientrano nelle sue funzionalità principali.

### 3.3.3 Presenza di un Web Server

Una volta individuata la tecnologia di archiviazione per le immagini più adatta alle necessità del sistema, alcune delle domande a cui è stato interessante rispondere sono state: *Che tipo di infrastruttura è necessario progettare affinchè le immagini memorizzate su Azure Blob Storage possano essere analizzate dal servizio di AI in modo automatico? Quale o quali componenti è opportuno introdurre per eseguire eventuali operazioni di pre-processing sulle immagini?* E infine, *è preferibile adottare un tipo di comunicazione sincrona o asincrona tra i vari servizi?*

Per rispondere a tutte queste domande, la strategia accolta è stata la stessa seguita per le altre scelte progettuali: Pensare a delle tecnologie che rispettassero i vincoli del contesto ma che, al contempo, fossero soprattutto in grado di soddisfare i requisiti del sistema. Power Automate è emerso come risposta ideale alla prima domanda. La sua capacità di creare flussi automatizzati sposa bene l'idea di interconnettere due componenti in modo efficiente, senza dover gestire manualmente ogni singola fase dell'interazione. Una volta terminato il caricamento delle foto su Azure Blob Storage, è sufficiente che l'utente segnali il verificarsi di tale evento per avviare il flusso in grado di leggere le immagini e inviarle direttamente al modello AI. L'architettura così definita sarebbe pienamente coerente e ben strutturata,

capace di assolvere alle specifiche funzionali per le quali è stata progettata. Ma dal punto di vista puramente qualitativo, il sistema aderisce agli standard prefissati durante l'analisi? L'introduzione di un componente intermedio come un web server nasce proprio da questa esigenza. In questo paragrafo si approfondiscono i vantaggi e gli svantaggi di entrambe le architetture, considerando aspetti come complessità progettuale, costi, facilità di manutenzione e flessibilità.

Con la prima tipologia di architettura **senza web server**, il numero di componenti da gestire e configurare è banalmente inferiore. Questo si traduce in semplicità strutturale e facilità di manutenzione: essendo tutto il lavoro centralizzato in Power Automate, infatti, esisteranno meno rischi di desincronizzazione tra i vari componenti. Altri vantaggi sono costituiti dalla presenza di costi più contenuti e da una velocità di sviluppo superiore, non dovendo ideare e scrivere codice backend. La flessibilità e l'integrazione sono, tuttavia, punti a sfavore di questa soluzione. È evidente, infatti, che l'uso di Power Automate imponga dei limiti alla logica di business rispetto a un server dedicato. Allo stesso modo, l'assenza di un web server incide negativamente sulla scalabilità del sistema. Se il volume delle richieste aumenta, i connettori Power Automate potrebbero rappresentare il collo di bottiglia dell'intero ecosistema software, causando un significativo rallentamento delle prestazioni complessive.

La seconda tipologia di architettura prevede, invece, la **presenza di un Web Server** come punto di contatto tra l'automazione Power Automate e il servizio di AI. L'introduzione di questo componente aumenta inevitabilmente la complessità della soluzione, introducendo nuove necessità operative, come il monitoraggio dello stato del server, la gestione delle dipendenze del codice e l'esecuzione periodica di aggiornamenti di sicurezza. Sorgono, inoltre, costi ulteriori dovuti all'hosting su servizi come Azure App Service o VM dedicate. Nonostante gli svantaggi evidenziati, il Web Server permette, tuttavia, di implementare logiche sofisticate come il pre-processing delle immagini, la validazione dei dati e la gestione avanzata degli errori. Il ruolo del web server è, inoltre, decisivo per la realizzazione di comunicazioni asincrone. Con la prima opzione architetturale, essendo Power Automate responsabile dell'invocazione del servizio di AI, il flusso è costretto ad aspettare il risultato dell'elaborazione prima di procedere con i passi successivi. Con l'introduzione del web server, invece, si ha la possibilità di disaccoppiare le operazioni svolte dall'attore che effettua la richiesta da quelle eseguite dal servizio chiamato. In questo modo, il chiamante può invocare il servizio senza rimanere in attesa della risposta, ottimizzando così l'uso delle risorse. In pratica, quando Power Automate interagisce con il server, quest'ultimo può rispondere immediatamente con una conferma di ricezione (ad esempio, restituendo un Job ID). A questo punto, Power Automate può proseguire con altre attività, mentre il web server continua l'elaborazione in background. Potrebbe essere il web server stesso, infatti, a fungere da buffer intermedio e occuparsi della gestione del risultato

e della memorizzazione su Dataverse, definendo un comportamento più efficiente dell'intero sistema. Infine, questa seconda tipologia di architettura permette di ottenere integrazioni personalizzate, facilitando la connessione tra Power Automate e sistemi o servizi non direttamente supportati dalla piattaforma Microsoft.

Al netto di queste considerazioni si è dunque optato per l'architettura più complessa, nonostante richieda tempi di sviluppo più lunghi e costi operativi superiori. Questa scelta è stata guidata dalla volontà di massimizzare il valore di business, estendendo le potenzialità del sistema e garantendo una struttura solida e scalabile, in grado di rispondere a esigenze future di natura diversa.

## 3.4 Evoluzione finale dell'architettura

Dalle decisioni progettuali discusse precedentemente emerge in modo del tutto naturale un'architettura software articolata, che vede l'integrazione di tecnologie eterogenee sotto tanti punti di vista. Questa complessità si traduce in un ecosistema che combina applicativi realizzati tramite piattaforme Low-Code con un server web implementato tramite un linguaggio di programmazione di alto livello. Il progetto prevede, inoltre, la creazione sia di un'app Canvas sia di un'app Model-Driven, offrendo un mix di flessibilità operativa e personalizzazione per soddisfare diverse esigenze degli utenti finali. È, al contempo, pianificato l'utilizzo di due soluzioni di archiviazione differenti: Dataverse, ideale per la gestione strutturata dei dati aziendali, e Azure Blob Storage, utilizzato per l'archiviazione scalabile degli oggetti non strutturati come le immagini. L'architettura definitiva del sistema include, dunque, una moltitudine di componenti software che interagiscono fra di loro in modo coordinato per raggiungere gli obiettivi prefissati durante la fase di analisi. Nei paragrafi successivi ci si immerge in profondità nei vari aspetti tecnici del sistema, con ampio spazio per diagrammi raffiguranti diverse prospettive che caratterizzano questo lavoro di Tesi.

Viene inizialmente presentata la struttura definitiva del progetto, entrando nel dettaglio dell'organizzazione interna e spiegando attentamente le connessioni tra i vari componenti. L'obiettivo è fornire una visione chiara e completa del funzionamento del sistema, descrivendo come le diverse tecnologie coinvolte collaborino per supportare un tipico caso d'uso.

Oltre alla vista d'insieme dei componenti, viene poi sviscerato il modello dati, ovvero la rappresentazione logica e concettuale delle informazioni gestite dal software. Durante questa fase si stabiliscono le entità coinvolte, i campi che le costituiscono e le relazioni tra di esse. Un modello dati ben progettato fornisce un riferimento per la scrittura del codice, consente agli sviluppatori di pianificare con precisione l'implementazione del sistema e aiuta nella manutenzione a lungo termine.

Un ulteriore aspetto del sistema che ha visto un'attenta fase di progettazione è quello delle interfacce. Esse rappresentano l'oggetto più vicino con il quale si confronta l'utente finale, per cui la realizzazione di prototipi accessibili e intuitivi determina non solo una facile interazione con il software, ma contribuisce a migliorare l'esperienza complessiva. Saranno quindi esplorati i principi che muovono il design delle UI, con particolare attenzione all'usabilità, all'accessibilità e alla coerenza visiva, per offrire un prodotto che risponda pienamente alle aspettative del target di riferimento.

### 3.4.1 System Design

Il sistema presenta l'architettura mostrata in Figura 3.4.

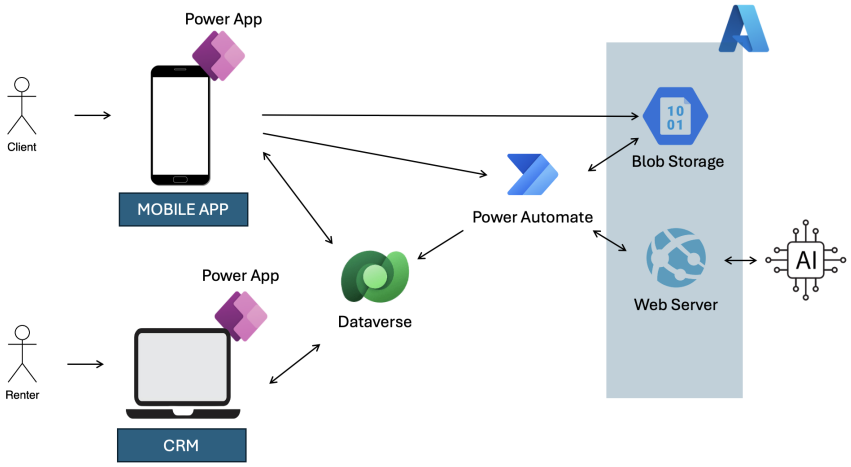


**Figura 3.4:** Diagramma Architetturale del sistema

Partendo da sinistra ritroviamo nuovamente gli attori Client e Renter. I Client, attraverso l'app mobile, hanno la possibilità di scattare delle foto al veicolo noleggiato e caricarle direttamente su Azure Blob Storage al momento del ritiro e della riconsegna. L'applicazione mobile è realizzata come una *Canvas App*, scelta che consente di offrire un'interfaccia chiara e diretta all'utente, non intrinsecamente vincolata al modello dati sottostante. Questo approccio conferisce, inoltre, un grado di configurabilità maggiore, permettendo di adattare facilmente l'applicazione alle mutevoli esigenze aziendali. I Renter, invece, hanno accesso al CRM, sviluppato come una *Model-Driven App*. Tramite questo Back-Office, i Renter possono avere accesso a una vasta gamma di informazioni relative ai clienti, ai veicoli e ai noleggi, che siano già terminati, in corso o futuri, attraverso l'interazione con Form e View.

Entrambe le Power App comunicano con Dataverse per memorizzare e reperire le informazioni necessarie, ciascuna con privilegi di accesso differenti. Mentre i

Client possono consultare esclusivamente i dati legati ai propri noleggi, i Renter dispongono di una visione completa e centralizzata dell'intero sistema. L'insieme costituito dalle due applicazioni, dal database e dal servizio Azure Blob Storage può essere considerato come un primo grande sottosistema autonomo, in grado di gestire efficacemente tutta la parte operativa legata al noleggio.

Spostandosi sulla parte destra della figura, si nota, invece, la presenza di due servizi Azure come Blob Storage e App Service. Per quanto riguarda il servizio di archiviazione, il suo unico scopo è memorizzare sul cloud le immagini caricate dai clienti in contenitori dedicati che rappresentano il noleggio associato. La connessione con Power App e Power Automate è resa agevole grazie alla disponibilità dei connettori built-in offerti da Microsoft per scenari di questo genere, in cui è richiesta una forte integrazione tra tecnologie differenti. Azure App Service è, invece, il servizio utilizzato per la distribuzione del web server, il quale costituisce il secondo sottosistema presente all'interno di tale architettura, fungendo da vero e proprio backend del sistema. Il server espone un'API per l'analisi delle immagini di veicoli, realizzata tramite l'integrazione con un servizio di AI. Per valutare l'intelligenza artificiale più adatta al caso in esame, è stato necessario interagire direttamente con essa, implementando alcune parti, piuttosto che limitarsi a una valutazione teorica dei vantaggi e svantaggi di ogni opzione. Per questo motivo, l'analisi sui servizi di AI considerati viene approfondita nel capitolo successivo.

Al centro del diagramma troviamo, infine, Power Automate. Questa tecnologia è stata la chiave per interconnettere i due sottosistemi attraverso la creazione di un *flusso istantaneo*. Il flusso generato scatta automaticamente nel momento in cui il Client preme uno specifico pulsante nell'app mobile. A partire da questo momento, il flusso legge le immagini dal contenitore Azure corrispondente e invoca, tramite HTTP, l'API del web server, passando in input le immagini recuperate. L'automazione Power Automate si conclude, infine, con l'ottenimento del risultato dell'analisi il quale, una volta convertito in un formato opportuno, viene memorizzato sotto forma di contenuto strutturato su Dataverse.

L'integrazione fluida tra le Power Apps, il backend su Azure App Service, e l'automazione tramite Power Automate garantisce un flusso end-to-end ottimizzato. Tale approccio pone solide basi per le fasi successive del progetto, rendendo il sistema facilmente estendibile e adattabile a futuri scenari operativi e requisiti funzionali. La modularità ottenuta attraverso la suddivisione in sottosistemi autonomi permette, infine, non solo una chiara separazione delle responsabilità tra le diverse parti, ma anche una maggiore flessibilità nell'eventuale aggiornamento o sostituzione di singoli elementi, senza compromettere l'intero sistema.

### 3.4.2 Model Design

In seguito alla presentazione dei componenti fondamentali, viene illustrato, in questo paragrafo, il modello dati del sistema, attraverso il supporto di un diagramma UML.

**UML (Unified Modeling Language)** viene definito formalmente come un linguaggio standard per la modellazione visiva di sistemi software orientati agli oggetti. Attraverso i suoi diagrammi statici e dinamici, UML fornisce degli strumenti per rappresentare graficamente l'architettura e il comportamento di una determinata soluzione. In questa sezione, l'obiettivo è, più precisamente, quello di analizzare la gerarchia delle classi che compongono il sistema, le dipendenze e le relazioni tra di esse. A tal fine, il **Class Diagram** è sicuramente la tipologia di schema più indicata, mettendo a disposizione una serie di convenzioni utili per la rappresentazione del concetto di classe, di ereditarietà, di composizione o di tanti altri aspetti che possono costituire un modello dati.

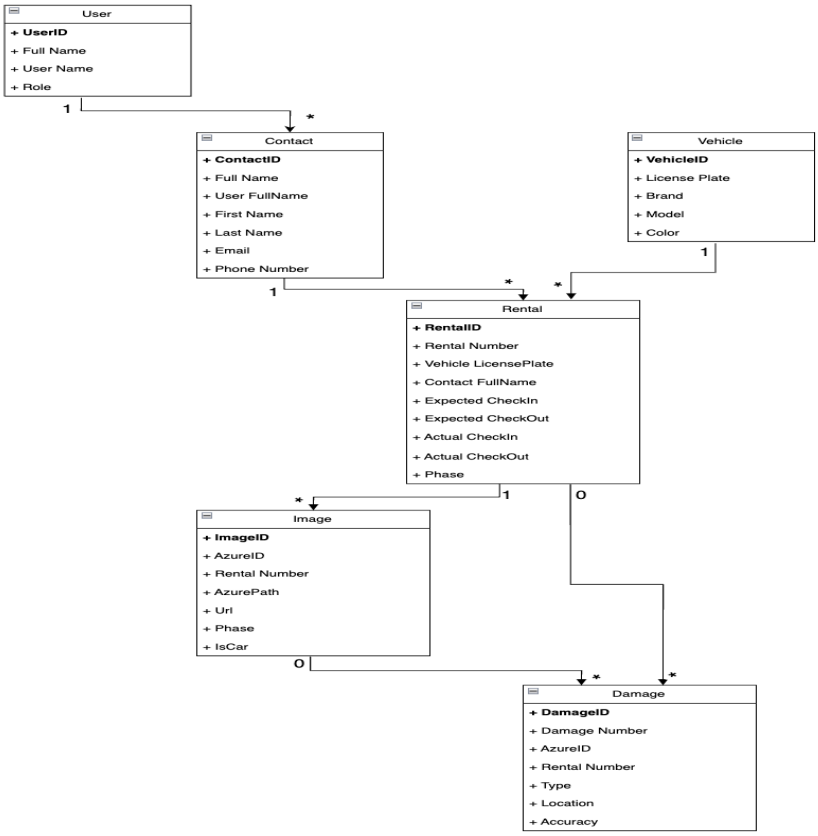


**Figura 3.5:** Diagramma delle Classi

Come mostrato dal diagramma in Figura 3.5, sono state individuate 5 classi principali che astraggono i concetti fondamentali di Veicolo, Contatto, Noleggio, Immagine e Danno. Per ogni entità vengono specificati i suoi attributi e le relazioni tra di esse con relativa cardinalità.

Al centro del Class Diagram si trova Rental, entità che rappresenta il cuore del sistema e che è legata alle classi Contact e Vehicle attraverso delle relazioni molti a uno (n-1). Ciò consente di gestire scenari in cui un cliente effettua più noleggi nel tempo o un veicolo viene utilizzato in più noleggi differenti. Il processo di noleggio, in particolare, si articola in due fasi principali: check-in e check-out, ognuna delle quali può essere documentata con una o più immagini del veicolo. Le immagini, a loro volta, possono contenere un elenco di possibili danni, motivo per cui si è adottata una struttura ad albero per rappresentare questa rete di relazioni, come illustrato in figura. Un ulteriore legame collega direttamente la classe Rental alla classe Damage. Questo accorgimento tecnico semplifica la gestione dei danni, consentendo al Renter di visualizzarne immediatamente la lista a partire direttamente dal *Form* del noleggio. Poiché i form delle Model-Driven app non consentono di navigare le relazioni oltre un livello, questo collegamento diretto non è solo una scelta progettuale, ma una necessità per aggirare tale vincolo di prodotto.

Oltre al blocco delle 5 entità menzionate, è presente in cima un'ulteriore classe User, fondamentale per il sistema sviluppato. Per consentire a un utente di accedere a una Power App, è infatti necessario registrarlo all'interno della tabella User, una delle tabelle predefinite di Dataverse. Un aspetto interessante di questo modello è che, sebbene ogni User sia solitamente associato a un singolo record Contact, la connessione tra le due entità avviene tramite una relazione 1-a-molti. Questa scelta è una peculiarità di Dataverse, che non supporta le relazioni 1-1 e obbliga dunque l'implementazione di una configurazione simile.

Un'ultima osservazione riguarda la presenza dell'entità Contact. Se il concetto di utente viene modellato attraverso l'entità User, perché è stata inclusa questa classe ulteriore? La risposta risiede nella visione futura e nella riutilizzabilità del modello dati. Attualmente, sia i Renter che i Client sono considerati *systemuser*, ognuno con una licenza dedicata per accedere alle proprie applicazioni. In futuro, una prospettiva invitante potrebbe essere quella di rendere necessario, per i soli Renter, il possesso di una tale licenza, consentendo di fatto all'azienda di evitare l'acquisto di una licenza per ogni singolo cliente. Questa scelta comporterebbe, tuttavia, l'adozione di una nuova tecnologia in grado di sostituire la Canvas App, come può essere, per esempio, una Power Page. In questo modo non sarebbe più necessario considerare i clienti come systemuser ma basterebbe mantenere tutte le loro informazioni anagrafiche all'interno dell'entità Contact.

### 3.4.3 User Interface Design tramite Figma per l'app mobile

I prototipi sono approssimazioni tangibili del comportamento e dell'aspetto di un sistema. Essi sono solitamente caratterizzati da diverse proprietà, le quali vengono presentate di seguito:

- *Scopo* - Un prototipo può essere utile per valutare un'idea, una funzionalità o l'interazione utente-sistema.
- *Copertura* - Un prototipo può approssimare l'intero sistema oppure concentrarsi su un singolo componente.
- *Fedeltà* - Esistono prototipi di bassa, media o alta fedeltà, i quali si distinguono in base al livello di somiglianza raggiunto rispetto al prodotto finale, sia in termini di aspetto visivo che di funzionalità.
- *Durabilità* - Questa caratteristica riguarda la solidità e la longevità del prototipo, non solo fisicamente ma anche dal punto di vista dell'affidabilità delle prestazioni nel tempo.
- *Utilizzo* - I prototipi possono essere statici, dinamici o interattivi, a seconda del livello di coinvolgimento che offrono agli utenti.
- *Completezza Funzionale* - I prototipi si distinguono anche in base al numero di funzionalità di sistema simulate e al livello di dettaglio raggiunto. In base a questa caratteristica, è possibile distinguere tra prototipi orizzontali, verticali e diagonali.

Una volta esaminati i diagrammi rappresentanti l'architettura e il modello dati del sistema, vengono, infine, presentati in Figura 3.6 i mockup dell'app mobile, realizzati attraverso il tool di progettazione UI/UX **Figma**. Questi modelli rappresentano una versione statica, di media fedeltà, di un'interfaccia utente. Pur essendo bozze semplificate della UI finale, permettono di anticipare il design e l'esperienza d'uso, fornendo un riferimento visivo per le successive fasi di sviluppo. Per tale motivo, una progettazione accurata della struttura e del contenuto di una pagina o di una schermata risulta essenziale non solo per migliorare l'esperienza dell'utente, ma anche per facilitare il lavoro degli sviluppatori. Essendo un prodotto digitale e statico, il mockup Figma è facile da mantenere, consentendo l'apporto di modifiche in modo rapido in base ai feedback ricevuti, senza costi elevati o tempi lunghi di revisione. Dal punto di vista della copertura e della completezza funzionale, infine, il mockup che si è scelto di sviluppare è un tipo di prototipo abbastanza verticale che modella le funzionalità della sola app mobile. Questo perché, essendo il CRM realizzato tramite Model-Driven App, non è possibile

personalizzare l'interfaccia grafica oltre un certo limite, il che rende superfluo un qualsiasi tipo di modellazione visiva.

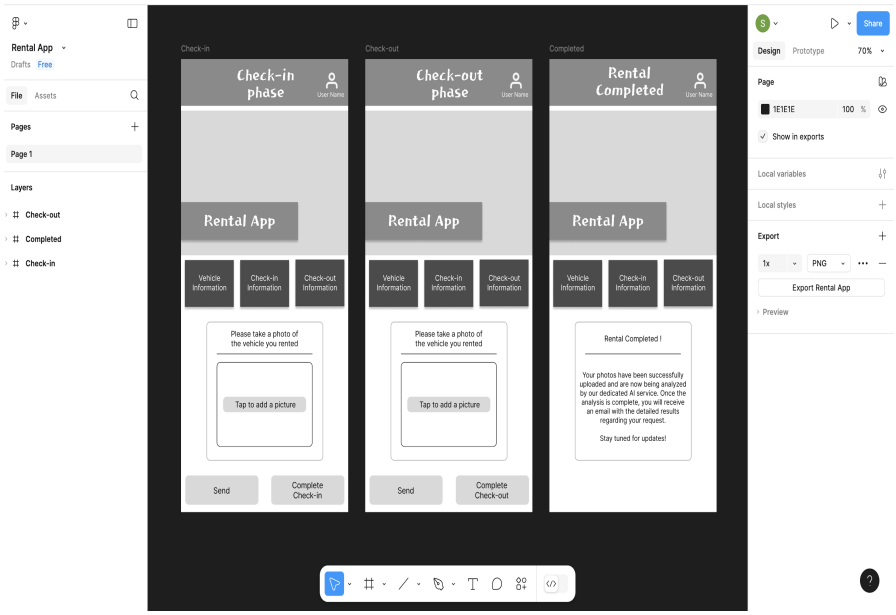


**Figura 3.6:** Mockup Figma dell'app mobile

Le interfacce progettate rappresentano la stessa schermata dell'app mobile, adattata alle diverse fasi del noleggio del cliente.

Nell'interfaccia a sinistra, l'utente che intende effettuare l'operazione di check-in ha la possibilità di scattare delle foto al veicolo e di caricarle, una per volta, sulla piattaforma tramite il tasto 'Send'. Una volta caricate tutte le foto, è possibile procedere al secondo step del noleggio attraverso il tasto 'Complete Check-In'. L'interfaccia centrale è quella relativa alla fase di check-out. Anche qui, per mantenere una coerenza tra le varie fasi, si è deciso di lasciare la stessa struttura visiva: un tasto, in basso a sinistra, permette di caricare le foto del veicolo al termine del noleggio, mentre un tasto in basso a destra consente di procedere con l'ultima fase. L'ultima interfaccia è quella relativa alla conclusione del noleggio ed ha un ruolo puramente informativo: qui l'utente viene avvisato della terminazione del processo e che riceverà via email i risultati dell'ispezione del veicolo. Le interfacce mantengono, infine, una parte comune per garantire uniformità tra di esse e aderenza agli standard di design. Si trovano, nella parte centrale, le informazioni essenziali del noleggio come la targa del veicolo e le date e gli orari previsti di check-in e check-out. Nella parte superiore della schermata si trovano, invece, ulteriori dettagli come il titolo dell'applicazione e funzionalità che completano l'esperienza utente.

# Capitolo 4

# Implementazione

Attraverso le tecnologie presentate nel capitolo 2, è stata realizzata una piattaforma che rappresenta un esempio di come Power Platform ed Azure possano incontrarsi per dar vita a software estremamente versatili o, allo stesso tempo, specializzati per specifici scenari. Sulla base della progettazione effettuata, si è scelto di procedere concentrandosi inizialmente sull'implementazione dei due sottosistemi principali, per poi passare alla loro integrazione attraverso Power Automate. Nei paragrafi successivi vengono esplorate non solo le funzionalità di ciascun componente, ma anche gli aspetti tecnici sottostanti, le sfide incontrate e le soluzioni adottate.

Ciascun'attività svolta durante questo lavoro è stata preceduta da un momento di formazione necessario per entrare in contatto con la tecnologia di interesse. Questa fase è stata supportata dalla supervisione del Project Manager e dallo studio della documentazione ufficiale Microsoft. Una volta configurato l'ambiente di lavoro, l'implementazione è iniziata con la creazione delle tabelle Dataverse, adottando il modello dati individuato come punto di partenza. Successivamente, il focus si è spostato su Power App e sulla realizzazione delle due applicazioni Canvas e Model-Driven, definendone le interfacce grafiche e approfondendo i meccanismi dietro le quinte che regolano l'interazione con i software di archiviazione utilizzati. Completata questa prima fase, ci si è immersi nella sfera AI studiando come i servizi disponibili su Azure possano essere sfruttati per l'analisi ed elaborazione di contenuti visivi. Vengono dunque esaminati, in relazione alla loro capacità di rilevare danni nelle immagini di automobili, sia i Servizi Cognitivi come Azure Computer Vision e Azure Custom Vision, sia quelli basati su AI generativa come Azure OpenAI. Una volta scelto il modello opportuno si è passati alla sua configurazione e all'automatizzazione delle richieste attraverso codice Python. È stato, dunque, implementato il web server responsabile della comunicazione con il modello di AI e distribuito sul cloud attraverso Azure App Service. Dopo lo sviluppo di questi due macro-componenti, il processo si è concluso, infine, con l'implementazione del flusso Power Automate e il salvataggio su Dataverse dei risultati ottenuti.

## 4.1 Setup del progetto

Per la realizzazione di questo progetto è stato necessario, inizialmente, configurare e gestire due ambienti principali: Microsoft Power Platform e Microsoft Azure.

Gli **Ambienti Power Platform** sono spazi di lavoro isolati che permettono di gestire applicazioni, flussi di lavoro, connessioni ai dati e altre risorse all'interno dell'ecosistema Microsoft Power Platform. Ogni ambiente ha la propria configurazione, autorizzazioni e risorse, garantendo così una netta separazione e un maggiore controllo tra diversi progetti o fasi di sviluppo. Per lo svolgimento di questo lavoro di tesi, è stato utilizzato, attraverso il portale **Make.PowerApps.com**, un ambiente Sandbox (ovvero di sviluppo) già predisposto da Cluster DCX Reply, all'interno del quale è stata creata la soluzione *RentalSystem*.

La creazione di una **Soluzione** è fondamentale nei contesti di applicazioni aziendali complesse, dove è necessario un controllo granulare sulle modifiche e sull'accesso ai dati. Tramite le soluzioni si possono, infatti, organizzare i componenti dell'applicazione in modo modulare, gestire le versioni e facilitarne la distribuzione tra ambienti diversi (ad esempio tra sviluppo, test e produzione).

All'interno della Soluzione RentalSystem sono stati sviluppati tutti i componenti della Power Platform necessari per il sistema. Come mostrato in Figura 4.1, al suo interno sono presenti le applicazioni Canvas e Model-Driven, le tabelle Dataverse, il flusso Power Automate e tanti altri componenti di supporto come connettori, option set, script JavaScript e ruoli di sicurezza.

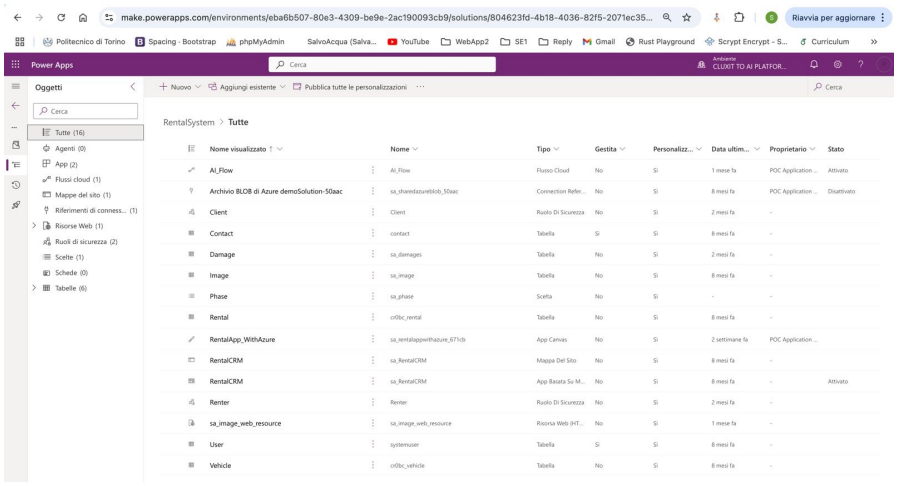


**Figura 4.1:** Componenti principali della soluzione RentalSystem

Oltre alla configurazione dell'ambiente Power Platform, è stato necessario predisporre un'infrastruttura su Microsoft Azure per ospitare i servizi di archiviazione,

calcolo e intelligenza artificiale necessari al funzionamento del sistema. In particolare, tutte le risorse Azure sono state raggruppate all'interno del Resource Group *dcxb1-us-pocrntl-rg-p-010*.

Un **Resource Group** rappresenta un contenitore logico che permette di organizzare, gestire e monitorare le risorse in modo centralizzato, facilitando l'amministrazione e l'ottimizzazione dei costi. Grazie a questa struttura, è possibile mantenere un alto livello di sicurezza e controllo, permettendo solo agli utenti autorizzati di visualizzare, modificare o distribuire le risorse all'interno dell'ambiente Azure.

L'accesso al Resource Group è stato ottenuto grazie all'assegnazione di specifici permessi da parte di Cluster DCX Reply, tramite il sistema di gestione delle identità e degli accessi di Azure (Azure Role-Based Access Control - RBAC). Grazie a questa configurazione, è stato possibile interagire con le risorse del Resource Group direttamente dal portale di Azure, dove sono state create e configurate le componenti essenziali per il sistema.

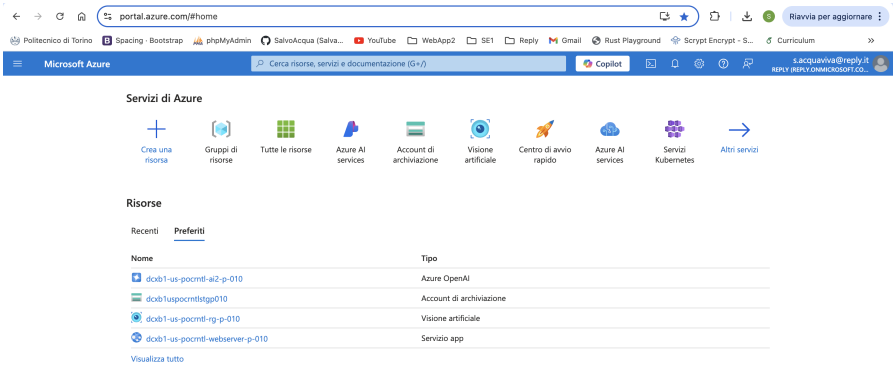


**Figura 4.2:** Configurazione delle risorse Azure

La gestione strutturata attraverso Solutions in Power Apps e Resource Groups in Azure ha garantito un'architettura scalabile, modulare e sicura, ottimizzando i processi di sviluppo e distribuzione del progetto.

## 4.2 Creazione delle tabelle Dataverse

All'interno della soluzione *RentalSystem* ogni entità individuata nel Class Diagram del Capitolo 3 ha trovato una corrispondente rappresentazione in una tabella Dataverse. Alcune di queste tabelle, come Vehicle, Rental, Image e Damage, sono state create appositamente per questa progettualità; altre invece, come User e Contact, sono tabelle predefinite, già disponibili all'interno della piattaforma.

Per le tabelle standard non è stato necessario effettuare modifiche significative poiché i concetti di utente (User) e contatto (Contact) rispecchiano fedelmente le

necessità del sistema, evitando così l'esigenza di ridefinire strutture già esistenti. La maggior parte dei campi previsti in fase di progettazione si sono, infatti, rivelati già presenti nelle tabelle predefinite, riducendo il bisogno di personalizzazioni. L'utilizzo delle tabelle standard garantisce, inoltre, diversi vantaggi, tra cui maggiore interoperabilità con altre soluzioni Dataverse e riduzione dei tempi di sviluppo e manutenzione. Ciò, tuttavia, non esclude la possibilità di personalizzarle per adattarle alle esigenze specifiche dell'applicazione. Nella tabella Contact, ad esempio, è stato introdotto il campo User FullName come campo in vincolo di chiave esterna con la tabella User, al fine di mantenere il collegamento diretto presente nel modello dati progettato.

Per quanto riguarda le tabelle custom, invece, ogni colonna è stata definita manualmente, scegliendo per ciascun attributo il tipo di dato più adeguato in base alla sua funzione e alla natura dei dati da gestire.

Dopo aver gestito la creazione delle tabelle, è stato definito un vincolo di calcolo per il campo Phase della tabella Rental. A differenza dei campi statici, questo valore viene calcolato automaticamente in base alla presenza o meno dei timestamp Actual CheckIn e Actual CheckOut per ciascun record di noleggio. Nello specifico, Phase assume il valore 2 se per il noleggio non è ancora stato effettuato il check-in, 1 se invece è presente il timestamp di check-in ma assente quello di check-out, 3 se entrambi i timestamp sono presenti. L'assegnazione di questi valori è spiegata più nel dettaglio nel paragrafo dedicato all'implementazione dell'app mobile.

Un vincolo imposto da Dataverse è la presenza di una **Colonna Primaria** per ogni tabella, un campo testuale obbligatorio che identifica il record in modo leggibile. La ragione dietro questa scelta progettuale è che, mentre l'identificativo **(GUID)** è utile per referenziare i record in modo univoco ma in modo non leggibile, la colonna primaria consente di avere un nome significativo per ogni record.

In Dataverse, i dati sono organizzati attraverso un'ampia gamma di tipi di dati, che permettono di modellare le informazioni in modo flessibile e adattabile a diversi scenari aziendali. Questo insieme include campi testuali e numerici per memorizzare stringhe e valori quantitativi, campi di scelta per selezionare opzioni predefinite, campi data/ora per gestire informazioni temporali e campi booleani per rappresentare valori vero/falso. Inoltre, Dataverse supporta tipi di dati avanzati come file e immagini, utili per archiviare documenti e contenuti multimediali direttamente nelle entità.

Tra questi, un ruolo fondamentale è svolto dai campi **Lookup**, che permettono di creare relazioni tra tabelle collegando un record a un altro. Anche se tecnicamente memorizza l'identificativo (GUID) del record correlato, il Lookup appare all'utente come un riferimento al record associato. Ciò significa che, quando una Power Apps utilizza un campo Lookup, può recuperare direttamente l'entità collegata ed accedere ai suoi campi. Ad esempio, se la tabella Rental ha un campo Lookup verso la tabella Vehicle, si può accedere a *Rental.Vehicle.LicensePlate* per ottenere la

targa del veicolo associato al noleggio. A livello di visualizzazione all'interno delle view e dei form delle Model-Driven App, invece, ciò che viene mostrato all'utente non è il GUID ma la colonna primaria del record associato.

Un campo Lookup funziona, dunque, in modo simile a una chiave esterna in un database relazionale tradizionale. Una differenza sostanziale, tuttavia, è che, attraverso le Lookup è possibile implementare soltanto relazioni del tipo 1-n o n-n. In altri termini, Dataverse non prevede nativamente le relazioni 1-1, poiché progettato per gestire dati in contesti più dinamici e scalabili, dove questo tipo di associazione è raramente utilizzato e spesso evitabile. Ciò è stato anticipato durante la descrizione della relazione tra la tabella User e la tabella Contact nel paragrafo relativo al modello dati del sistema. Per bypassare questo vincolo in modo tale da avvicinarsi il più possibile a un tipo di relazione 1-1 classica, è stato aggiunto un ulteriore vincolo di unicità al campo 'User FullName' della tabella Contact. In questo modo sarà impossibile per un utente avere più contatti associati, seppur la relazione rimanga a livello teorico una relazione 1-n.

## 4.3 Front End

### 4.3.1 App CRM

Una volta definite le tabelle e le relative relazioni, il passo successivo è stato configurare il rendering dei dati all'interno della Model-Driven App per realizzare un CRM capace di supportare tutte le funzionalità individuate in fase di analisi. Le Model-Driven App sono applicazioni strettamente legate ai dati, la cui struttura riflette fedelmente l'organizzazione delle entità nel modello dati sottostante. Questa caratteristica è il motivo per cui è possibile configurare tali applicazioni senza la necessità di ricorrere ad ulteriori piattaforme, basandosi piuttosto interamente sulla definizione di form e view delle tabelle all'interno di Dataverse. Questi elementi consentono di personalizzare la visualizzazione e l'interazione con i dati a seconda delle necessità dell'utente.

Tramite il portale *Make.PowerApps.com*, sono state inizialmente sviluppate le *Public View* per ciascun'entità, selezionando i campi da mostrare, formattando opportunamente alcune tipologie di dati (come i timestamp) e predisponendo filtri specifici. Nel caso dei noleggi sono, inoltre, state previste diverse Public View, in modo tale da fornire al Renter un accesso rapido ai noleggi terminati, in corso e futuri. Oltre alle Public View, sono state definite ulteriori tipologie di viste note come *Lookup View* e *Subgrid*. Le Lookup View sono viste utilizzate per visualizzare un insieme limitato di campi di un'entità quando si deve selezionare un record correlato all'interno di un altro record. Le Subgrid, invece, sono componenti che permettono di visualizzare, in una griglia interattiva, un elenco di record correlati a un'entità principale. Vengono utilizzate per rappresentare relazioni uno-a-molti o

molti-a-molti, facilitando la navigazione e l'accesso rapido ai dati associati. Tramite le Subgrid è stato possibile, per esempio, mostrare tutti i noleggi legati a un singolo contatto o tutte le immagini relative a un particolare noleggio.

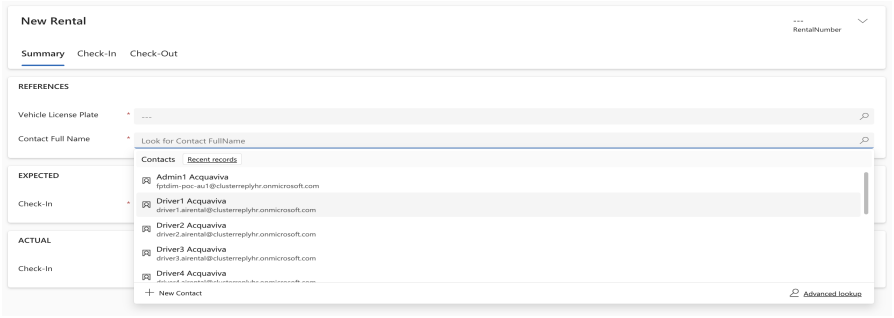


**Figura 4.3:** Esempio di Lookup View

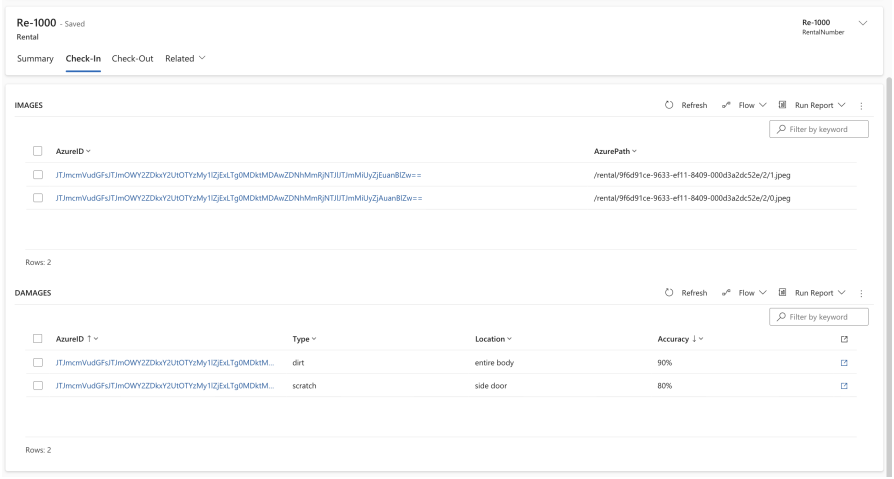


**Figura 4.4:** Esempio di Subgrid

Attraverso i Form è stato possibile definire come gli utenti interagiscono con i record delle entità, determinando quali campi devono essere visualizzati, in quale ordine e con quali opzioni di input. Un form può possedere al suo interno diversi componenti come i *tab* per organizzare i dati in schede diverse e le *sezioni* per raggruppare a loro volta i campi correlati. Oltre a questi componenti di Layout, esistono anche tante altre tipologie di elementi come caselle di controllo, menu dropdown e calendari. Sono, dunque, stati configurati form specifici per ogni entità, per permettere agli utenti di inserire, modificare e visualizzare le informazioni in modo semplice e intuitivo.

Nei form delle Model-Driven App è possibile, inoltre, applicare una personalizzazione avanzata utilizzando degli script per estendere la funzionalità e l'interattività dell'interfaccia utente. Un esempio in cui è stato necessario intervenire attraverso codice Javascript è costituito dal Form delle immagini. Affinché sia possibile visualizzare del contenuto multimediale, come per esempio le foto dei veicoli caricate su Azure Blob Storage, occorre infatti interagire con il DOM tramite l'**execution context**, ovvero il contesto in cui il codice Javascript viene eseguito. Tramite esso, lo sviluppatore può accedere e modificare la struttura della pagina intervenendo puntualmente su uno specifico elemento HTML, come un tag <img>, e modificando dinamicamente l'attributo src per specificare l'URL dell'immagine online.

A seguire le immagini delle principali schermate del CRM sviluppato.

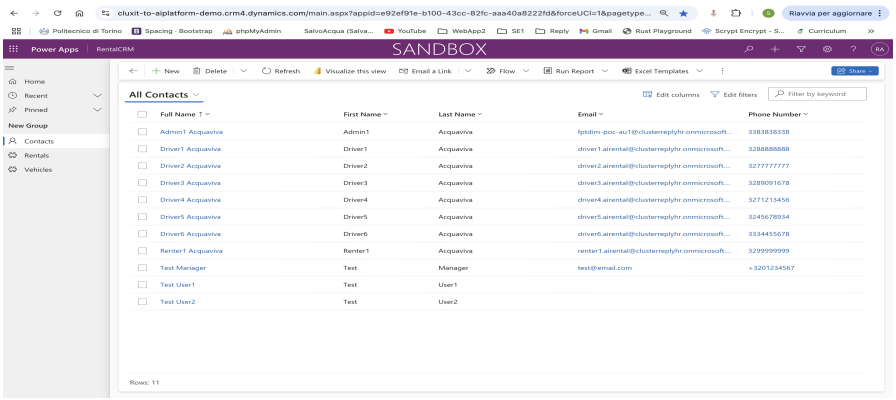


**Figura 4.5:** CRM: Schermata Contatti

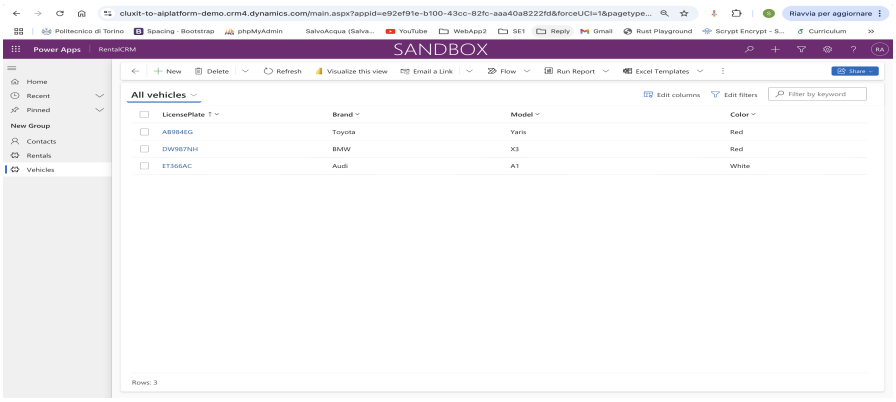


**Figura 4.6:** CRM: Schermata Veicoli

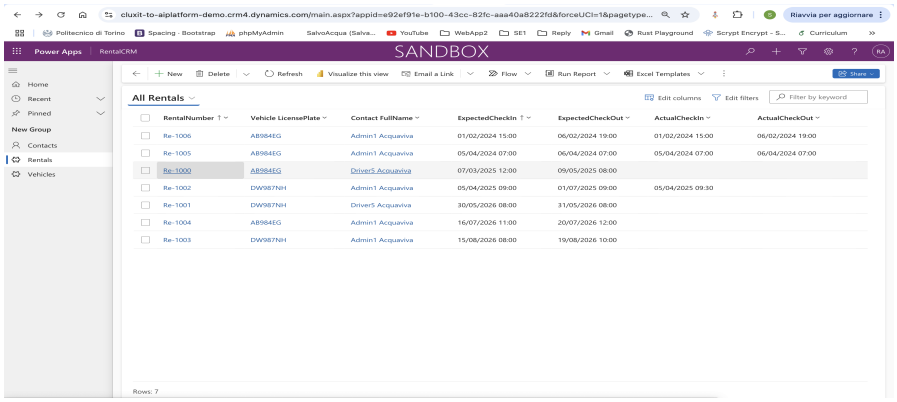


**Figura 4.7:** CRM: Schermata Noleggi

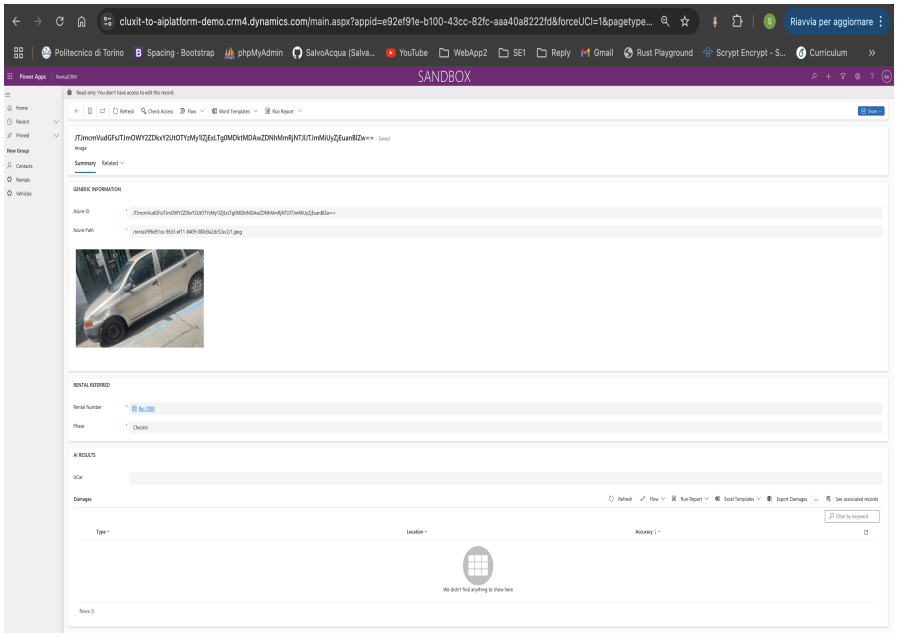


**Figura 4.8:** CRM: Schermata Immagine pre AI-analysis

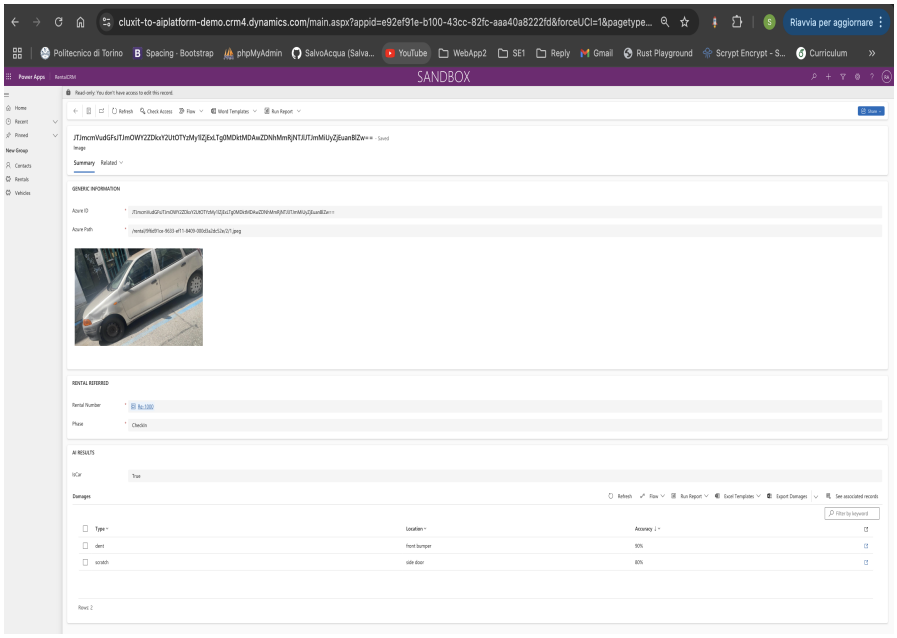


**Figura 4.9:** CRM: Schermata Immagine post AI-analysis

### 4.3.2 App Mobile

Sempre tramite il portale *Make.PowerApps.com* è stata sviluppata l'app mobile destinata al cliente. L'applicazione, concepita come Canvas App, è stata generata a partire da un'area di lavoro completamente vuota, consentendo così un pieno controllo sulla disposizione degli elementi e sull'architettura dell'esperienza utente. A supporto di questa libertà progettuale, l'editor visuale di Power Apps, basato sul paradigma **WYSIWYG (What You See Is What You Get)**, permette di modellare l'interfaccia in tempo reale, semplificando l'organizzazione degli elementi grafici e la configurazione delle loro proprietà senza la necessità di scrivere codice manualmente.

Data la natura dell'applicazione e gli obiettivi funzionali prefissati, lo sviluppo è stato interamente realizzato all'interno di un'unica schermata. Il primo passo implementativo è stato, quindi, suddividere l'interfaccia in sezioni funzionali ben definite, ciascuna delle quali ospitante una serie di componenti interattivi (noti anche come *control*). Tali componenti sono stati organizzati secondo un opportuno **layering di visualizzazione**, ovvero un sistema di sovrapposizione e gerarchizzazione che consente di gestire la loro disposizione e visibilità in relazione agli altri elementi presenti nell'app. Ogni control possiede proprietà configurabili dinamicamente attraverso un menù contestuale, che permette di personalizzare aspetti come le dimensioni, lo stile grafico, il contenuto testuale e il comportamento interattivo.

Determinante per una gestione avanzata delle proprietà dei componenti è l'utilizzo di **Power FX**, un linguaggio dichiarativo sviluppato appositamente per la piattaforma Power Apps. Ispirandosi ai principi di Excel, Power Fx fornisce una sintassi semplice e accessibile, basata su formule, per manipolare la struttura dell'applicazione, definire comportamenti dinamici e controllare l'interazione tra più schermate.

Nel caso di questo lavoro, Power Fx si è rivelato particolarmente utile innanzitutto per l'inizializzazione e la gestione di variabili locali e globali, fondamentali per il controllo dello stato dell'app e delle interazioni utente. La principale differenza tra le due tipologie di variabili risiede nel loro ambito e persistenza: mentre le variabili globali sono accessibili da qualsiasi schermata e mantengono il valore finché l'app è in esecuzione, quelle locali hanno sicuramente uno scope più ristretto, rimanendo disponibili e persistenti solo all'interno della schermata in cui vengono dichiarate. Nonostante l'applicazione sia attualmente costituita da una sola schermata, l'uso strategico di entrambe le tipologie di variabili garantisce una gestione più chiara e strutturata dello stato dell'app. In prospettiva futura, è buona norma di programmazione prevedere un'organizzazione delle variabili che faciliti eventuali sviluppi successivi, come l'aggiunta di nuove schermate o funzionalità, senza compromettere la manutenibilità del codice. Grazie a Power Fx è stato possibile, inoltre, gestire la visibilità dei componenti in modo dinamico, rendendoli visibili o nascosti in base al verificarsi di condizioni specifiche o eventi predefiniti. Attraverso questo linguaggio sono, infine, state definite tutte le logiche di processo che coinvolgono direttamente l'app mobile, illustrate in Figura 4.10, 4.11 e 4.12. Tali immagini mostrano il codice di tre distinti *Event Handlers*.

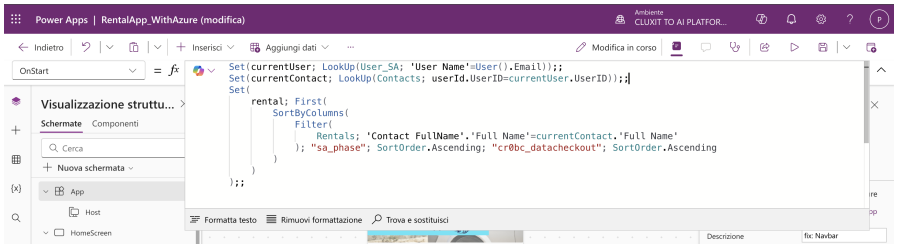


**Figura 4.10:** Logica associata alla scelta del noleggio

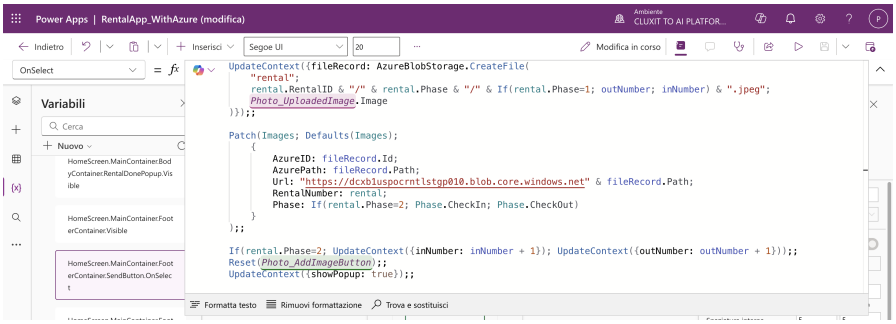


**Figura 4.11:** Logica associata al pulsante Send

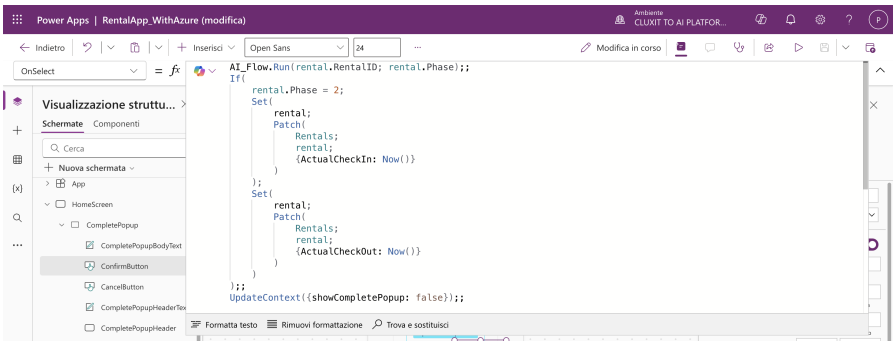


**Figura 4.12:** Logica associata all'avanzamento di fase per uno specifico noleggio

La prima immagine mostra una porzione di codice responsabile dell'estrazione del noleggio corretto per l'utente autenticato. Poiché un cliente può avere più noleggi a suo nome distribuiti nel tempo, è stato necessario definire una logica per determinare quale noleggio visualizzare al momento dell'apertura dell'applicazione. Poiché il focus di questa tesi non è la gestione complessa dei noleggi, si è optato per una soluzione semplificata che presuppone l'impossibilità da parte dell'utente di avere noleggi sovrapposti temporaneamente. Sebbene questa ipotesi risulti ragionevole, va notato che si tratta di uno scenario non ancora gestito dal sistema POC sviluppato. A partire da questa assunzione, è stato deciso di mostrare all'utente il noleggio attualmente in corso (ovvero quello per cui è stato fatto il check-in ma non ancora il check-out). Se l'utente non ha noleggi attivi, il sistema mostrerà il noleggio futuro con la data di check-out più recente. In assenza di noleggi futuri, verrà mostrato il noleggio completato con la data di check-out più antica. La logica alla base di questa selezione utilizza diverse funzioni come *LookUp*, per effettuare delle query su campi vincolati da chiavi esterne, e *Set*, per assegnare alla variabile globale specificata come primo parametro il valore del secondo parametro. Una volta memorizzato il contatto associato al systemuser all'interno della variabile 'currentContact', sono infine state utilizzate le funzioni *Filter* per selezionare i

record dell'entità Rental in base al campo 'Contact FullName', *SortByColumns* per ordinare il risultato secondo il criterio precedentemente descritto e *First* per estrapolare solo il primo record. La presenza di una funzione di ordinamento giustifica perché sia stato assegnato al campo Phase il valore minore per la fase di 'Check-out' e maggiore per la fase 'Complete'.

Nella seconda immagine è mostrato, invece, il codice eseguito ogni volta che l'utente preme il pulsante 'Send'. Questo pulsante viene utilizzato durante le fasi di check-in e check-out per caricare l'immagine, memorizzata all'interno del componente *Photo_UploadImage*, direttamente su Azure Blob Storage attraverso il supporto dell'omonimo connettore. Più precisamente, sfruttando proprio uno dei metodi di questo componente come *CreateFile*, viene generato un blob all'interno del contenitore 'rental' contenente l'immagine fornita dall'utente. Il file viene salvato con una nomenclatura strutturata nel formato:

<RentalID>/<RentalPhase>/<Count>.jpeg

dove <RentalID> identifica univocamente il noleggio associato, <RentalPhase> ne specifica la fase e <Count> rappresenta un intero che tiene traccia del numero di immagini caricate per quel noleggio nella fase specificata. Attraverso questa convenzione sulla struttura dei nomi, Azure Blob Storage riesce automaticamente a suddividere tali file in directory virtuali, organizzando così i dati in modo più efficiente e migliorando di fatto la reperibilità delle immagini archiviate. Il codice che segue si occupa, innanzitutto, della creazione di un record sulla tabella Image per ogni immagine caricata nel cloud. Grazie alla funzione *Patch* è, infatti, possibile creare un nuovo record quando il secondo parametro assume il valore Defaults(Image), oppure modificare un record esistente se viene fornita una condizione di ricerca che ne consenta l'identificazione. Successivamente viene aggiornato il contatore opportuno attraverso *UpdateContext*, funzione analoga a *Set* che permette bensì di aggiornare variabili locali piuttosto che globali. Il gestore dell'evento termina, infine, con il reset del componente di caricamento immagini, consentendo così all'utente di acquisirne o selezionarne una nuova, e l'aggiornamento della variabile 'showPopup', utilizzata per indicare all'utente, attraverso una notifica, il corretto caricamento dell'immagine sulla piattaforma.

La terza immagine illustra, infine, la logica che regola l'avanzamento di fase di un noleggio. Alla pressione del pulsante 'Complete' viene mostrato a schermo un popup che richiede all'utente la conferma per procedere con la fase successiva. Se il cliente conferma l'operazione, vengono eseguite due operazioni principali:

- *Attivazione del flusso di elaborazione delle immagini* - Il metodo *Run* viene utilizzato per lanciare, in modo asincrono, il flusso Power Automate responsabile dell'analisi automatizzata delle immagini. Tale metodo consente, inoltre, di specificare i parametri che il flusso si aspetta di ricevere per l'elaborazione.

Internamente, Power Apps registra automaticamente un connettore interno per consentire l'integrazione con Power Automate senza alcuna configurazione aggiuntiva. L'esecuzione in background del flusso rappresenta un vantaggio significativo in termini di esperienza utente poichè evita che l'utente debba attendere il completamento dell'elaborazione delle immagini prima di procedere con lo step successivo. La sincronia potrebbe, infatti, costituire un vero e proprio collo di bottiglia per il sistema, specialmente in scenari in cui il servizio di AI si trova a dover gestire un elevato numero di richieste simultanee.

- *Aggiornamento dello stato del noleggio su Dataverse* - Grazie all'utilizzo combinato delle funzioni *Patch* e *Set*, è possibile aggiornare rispettivamente la fase del record su Dataverse e la variabile globale associata al noleggio. Più precisamente, essendo la fase un campo calcolato a partire dai valori di 'Actual CheckIn' e 'Actual CheckOut', l'operazione non modifica direttamente la fase; ciò che viene impostato è il timestamp corrispondente al momento effettivo del check-in o check-out, determinando così l'aggiornamento automatico della fase.

L'event handler termina, infine, con l'aggiornamento della variabile 'showCompletePopup' che gestisce la chiusura del popup di conferma e consente così la transizione all'interfaccia relativa alla fase successiva.

Il paragrafo si conclude con le immagini della schermata definitiva dell'app mobile. L'interfaccia mostrata rappresenta i diversi stati attraversati da un noleggio nel suo ciclo di vita. Il suo sviluppo ha seguito fedelmente i mockup Figma progettati in fase di design, garantendo così un'aderenza ottimale tra la progettazione concettuale e l'implementazione finale.

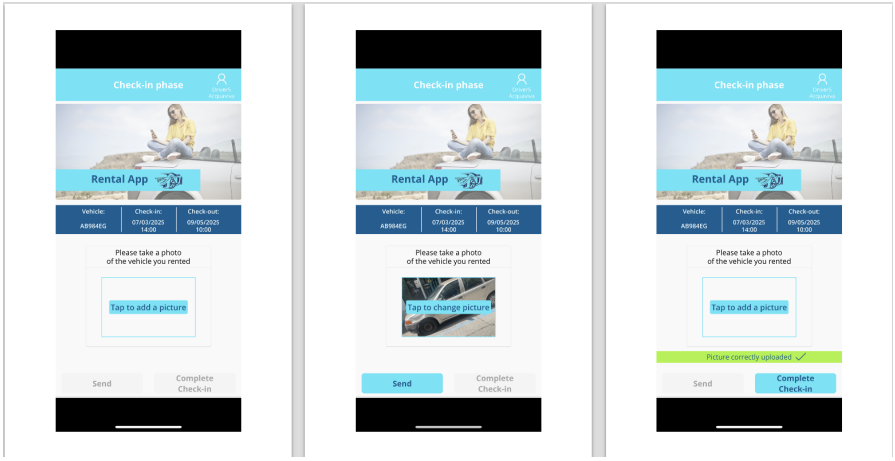


**Figura 4.13:** Mobile App: Fasi principali di un noleggio

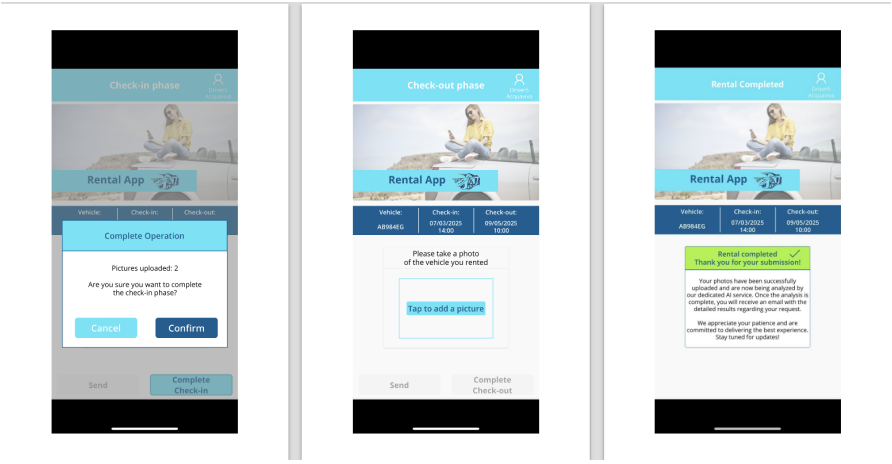


**Figura 4.14:** Mobile App: Fasi principali di un noleggio (2)

## 4.4 Analisi dei servizi Azure AI disponibili

Nel capitolo 2 sono state presentate alcune tecnologie di AI offerte da Azure per l'analisi di contenuti visivi. Sotto la sfera di *Azure AI & ML* rientrano, infatti, un insieme di servizi in grado di fornire funzionalità e algoritmi eterogenei adatti per qualsiasi scenario. In questo paragrafo, l'obiettivo è quello di mostrare come tali strumenti potrebbero essere sfruttati all'interno della mia soluzione, esaminando le performance ottenute in seguito a dei test preliminari, per poi giungere alla scelta della piattaforma definitiva.

I primi servizi presi in considerazione sono stati quelli appartenenti alla famiglia dei Servizi Cognitivi, come Computer Vision e Custom Vision. Questi servizi, basati su algoritmi di Machine Learning, condividono la capacità di analizzare immagini, ma si distinguono per il diverso grado di personalizzazione e addestramento.

### 4.4.1 Azure Computer Vision

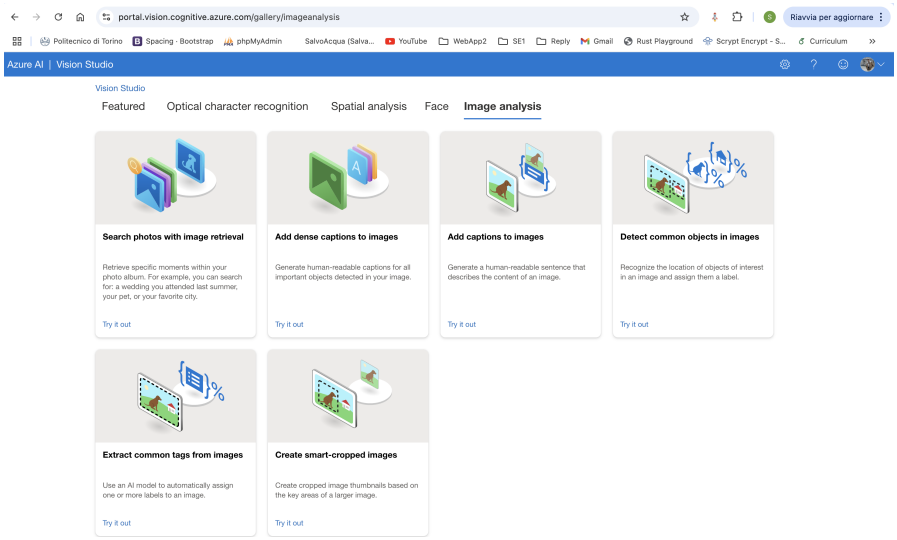


**Figura 4.15:** Modelli offerti da Computer Vision per l'analisi delle immagini

Tra i tre servizi analizzati, Azure Computer Vision è senza dubbio il più semplice da utilizzare. L'utente ha, infatti, a disposizione una serie di modelli pre-addestrati pronti all'uso, con possibilità di personalizzazioni limitate. Come illustrato nella schermata del portale **Vision Studio** in Figura 4.15, è possibile selezionare la categoria di funzionalità di interesse per accedere ai diversi modelli offerti dal servizio. Tra questi, si distinguono:

- *Search photos with image retrieval* - Modello capace di individuare un'immagine all'interno di un dataset a partire da una descrizione testuale del suo contenuto;
- *Add dense captions to images* - Modello progettato per generare didascalie in base agli oggetti rilevati all'interno dell'immagine;
- *Add captions to image* - Modello in grado di produrre una breve frase descrittiva del contenuto di un immagine;
- *Detect common objects in images* - Modello che riconosce gli oggetti presenti in un immagine, ne identifica la posizione e assegna loro un'etichetta;
- *Extract common tags from images* - Modello di classificazione in grado di associare una o più etichette all'intera immagine sulla base del suo contenuto;
- *Create smart-cropped images* - Modello che genera miniature ritagliate automaticamente in base alle aree chiave dell'immagine originale.

Nonostante alcuni di questi modelli siano in grado di analizzare il contenuto di un'immagine e di restituire un output testuale sulla base di ciò che viene rilevato, essi non risultano adatti al sistema sviluppato in questo lavoro. Pur essendo teoricamente capaci di eseguire il compito richiesto, la loro efficacia è solo nominale. Il problema principale risiede nella loro natura generalista: gli algoritmi di Azure Computer Vision sono stati addestrati per riconoscere oggetti semplici o per distinguere categorie generiche come, per esempio, un cane da un gatto. Tuttavia, quando si tratta di ispezioni automatizzate di veicoli, lo scenario diventa significativamente più complesso, poiché il grado di dettaglio richiesto è molto più elevato. Non è sufficiente distinguere solo se l'immagine analizzata sia una vettura o meno, ma è necessario rilevare eventuali danni come graffi, ammaccature, deformazioni o rotture. Tale livello di finezza è pertanto irraggiungibile tramite Computer Vision ed è da ricercare in servizi differenti.

### 4.4.2 Azure Custom Vision

Custom Vision rappresenta una tecnologia particolarmente interessante, che si presta bene a scenari specifici come l'ispezione dei veicoli. Questo servizio assicura quell'elevato grado di personalizzazione e flessibilità richiesto per questo contesto, a differenza di altri servizi Azure che non dispongono delle stesse capacità di adattamento. Tramite Custom Vision, l'utente può, infatti, creare modelli su misura per i propri casi d'uso grazie a meccanismi di addestramento personalizzabili. Consentendo il caricamento di immagini specifiche come training set, il modello è pertanto più abile nell'apprendere e riconoscere dettagli rilevanti, come eventuali imperfezioni nei veicoli. Questo perché il processo di apprendimento si basa sull'analisi di esempi concreti, i quali riflettono fedelmente le condizioni reali con cui il modello dovrà confrontarsi durante le effettive predizioni. Un ulteriore vantaggio è costituito dal maggiore controllo sul processo di addestramento. È, infatti, possibile migliorare il modello nel tempo, aggiungendo nuove immagini o perfezionando i parametri per ottenere risultati sempre più precisi. Infine, Custom Vision non si limita a riconoscere oggetti all'interno di un'immagine, ma fornisce anche informazioni aggiuntive. Oltre a identificare gli oggetti presenti, il modello è, infatti, capace di evidenziare le aree in cui questi si trovano e, per ciascuno di essi, indica il livello di confidenza stimato, migliorando così l'interpretabilità delle predizioni.

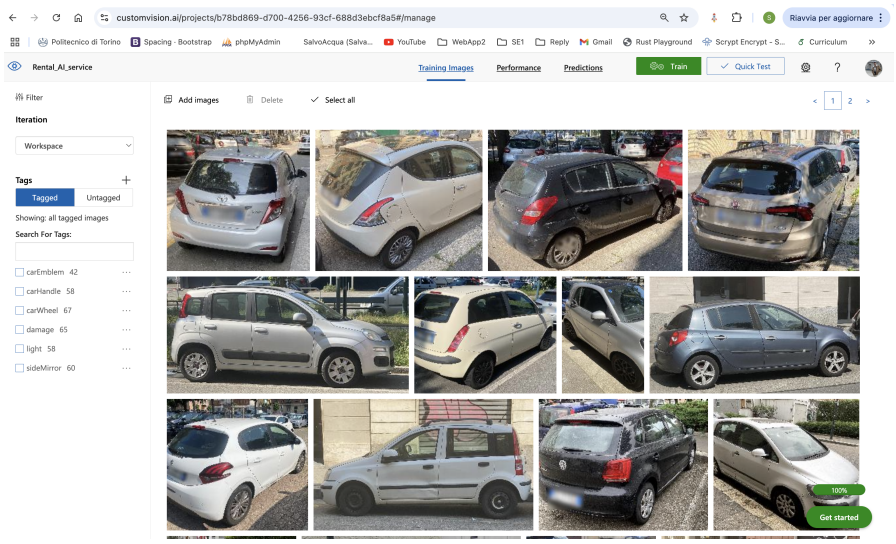


**Figura 4.16:** Portale Web di Azure Custom Vision

Entrando più nel dettaglio, per avviare lo sviluppo di un modello personalizzato è stato utilizzato il portale web **Custom Vision** con lo scopo iniziale di familiarizzare con il servizio e condurre una prima analisi delle potenzialità del modello costruito. In Figura 4.16 è illustrata l'interfaccia grafica di tale piattaforma. Essa si presenta come una pagina molto intuitiva, mostrando le principali funzionalità offerte da Custom Vision nella parte superiore della schermata mentre, nelle sezioni sottostanti, vengono presentati i dati e i parametri fondamentali su cui si basa il modello di AI.

Durante la fase di setup, è stato necessario selezionare il dominio più vicino al contesto di interesse e definire una serie di etichette (o tag) per identificare gli elementi che il sistema dovrebbe riconoscere all'interno delle immagini. Successivamente, il processo seguito per la realizzazione di questo modello non è stato caratterizzato da un susseguirsi di operazioni in modo strettamente sequenziale. L'idea è stata quella, infatti, di ripetere i vari step implementativi una serie di volte, cercando di costruire un modello efficiente attraverso raffinamenti successivi. Ogni iterazione è stata, tuttavia, caratterizzata da 5 fasi:

- *Raccolta delle immagini* - Questa fase preliminare ricopre un ruolo molto importante. La scelta delle immagini che costituiscono il training set è, infatti, decisiva per la qualità del modello finale. Questi dati devono essere rappresentativi di quelli che il sistema riceverà in produzione e, pertanto, dovrebbero coprire quanti più scenari diversi possibili. Poiché le fotografie caricate dagli utenti non possono essere ricondotte a condizioni fisse, ma variano per luminosità, angolazione, livello di zoom e altri fattori, è stato necessario includere immagini che presentassero prospettive diverse, modelli di veicoli eterogenei e differenti tipologie di danni.

- *Caricamento delle immagini* - Una volta raccolte, le immagini vengono caricate all'interno della piattaforma e suddivise in due categorie: tagged (ovvero quelle già classificate e utilizzabili dal modello come training o validation set) e untagged (quelle non ancora classificate dall'utente e quindi prive di qualsiasi etichetta).

- *Annotazione e Classificazione* - In questa fase, l'utente, attraverso strumenti grafici di selezione, ha la possibilità di evidenziare determinate aree dell'immagine e assegnare, a ciascuna di esse, una delle etichette personalizzate previste durante il setup del progetto. Questo passaggio cruciale per la successiva fase di addestramento permette al modello di apprendere le caratteristiche distintive di ciascun elemento.

- *Addestramento* - Dopo aver etichettato un numero sufficiente di immagini, viene avviata la procedura di training. Quì l'utente ha la possibilità di configurare alcuni dei parametri disponibili per ottimizzare il processo di apprendimento, dopodiché il servizio elabora i dati e genera il modello.

- *Test e Valutazioni* - Al termine dell'addestramento, il modello viene valutato utilizzando un validation set, ovvero un sottoinsieme di immagini non incluse nel training set, ma già etichettate. Il sistema confronta i risultati ottenuti con le etichette assegnate dall'utente, verificando la correttezza delle predizioni, la posizione degli oggetti riconosciuti e la categoria assegnata. Le performance vengono, quindi, sintetizzate attraverso dashboard interattive, che forniscono metriche dettagliate per valutare l'efficacia del modello.

In Figura 4.17 è mostrata un'immagine di input al modello, evidenziando il processo di classificazione descritto in precedenza. È innanzitutto importante notare la scelta delle etichette: nonostante il modello avesse come obiettivo il riconoscimento dei soli danni e la loro distinzione per tipologia, sono stati volutamente inclusi ulteriori tag per studiare il comportamento di tale strumento con elementi più semplici da individuare in quanto caratterizzati da pattern ricorrenti e facilmente distinguibili. Altro aspetto rilevante è sicuramente la selezione dell'area da evidenziare. Affinché si ottenga un modello ragionevolmente preciso, infatti, è fondamentale segnalare una porzione di immagine attraverso delle linee che delimitino bene i confini di ogni singolo oggetto. Gli algoritmi che ricevono le immagini classificate analizzano a fondo ogni singolo pixel e, se l'area evidenziata non include esattamente l'oggetto, trascurando determinati dettagli decisivi per il suo riconoscimento, il modello finale potrebbe non rispecchiare gli standard di qualità prefissati.

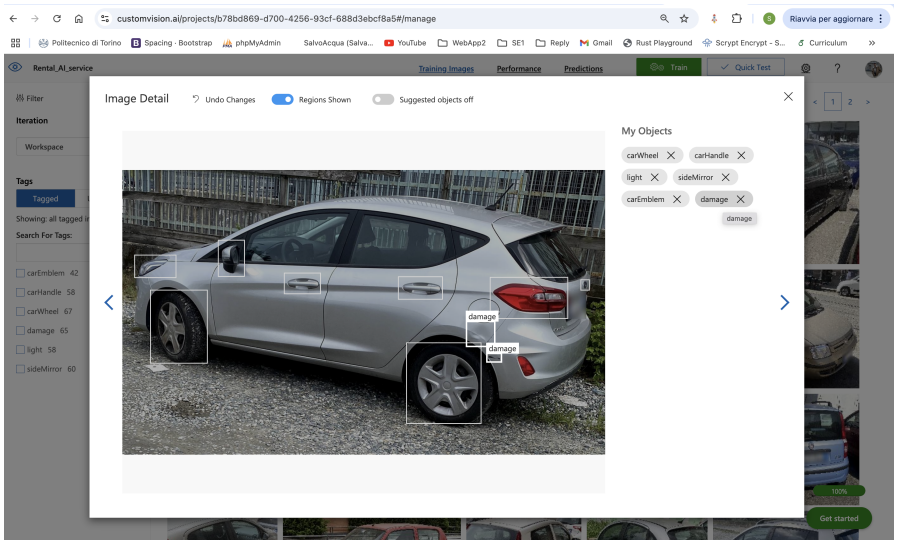


**Figura 4.17:** Classificazione di un'immagine su Azure Custom Vision

Al termine di ogni iterazione sono state poi esaminate le prestazioni raggiunte dal modello e, nel caso di bassa accuratezza e precisione, sono stati ripetuti i passaggi precedenti caricando nuove immagini e regolando opportunamente i parametri di training. I cicli eseguiti si sono basati, dunque, su dataset sempre più ampi sino ad arrivare a circa 50 immagini utilizzate come insieme di addestramento. I risultati riportati di seguito fanno riferimento alla quinta e ultima iterazione dove si sono ottenute le migliori performance.

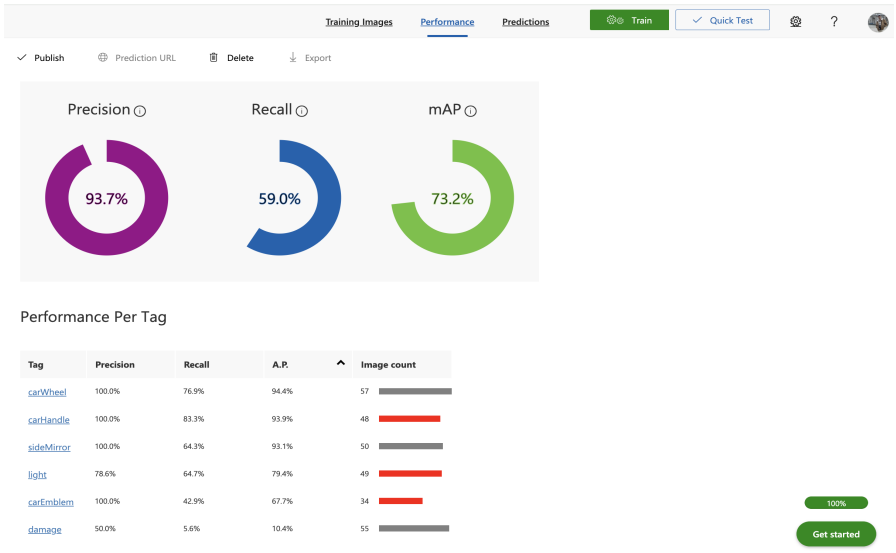


**Figura 4.18:** Performance ottenute con Azure Custom Vision

La dashboard illustrata espone sia le prestazioni generali del modello, sia quelle specifiche per ciascuna etichetta, attraverso metriche classiche di Data Science come la Precisione, il Richiamo e la Precisione Media. Queste tre misure vengono definite come:

$$\textbf{Precisione} = \frac{TP_C}{TP_C + FP_C}$$

$$\textbf{Richiamo} = \frac{TP_C}{TP_C + FN_C}$$

$$\textbf{Precisione Media} = \int_0^1 P(R)\, dR$$

Dove:

$\mathbf{TP_C}$ = Numero di oggetti correttamente assegnati al tag C

$\mathbf{FP_C}$ = Numero di oggetti erroneamente assegnati al tag C

$\mathbf{FN_C}$ = Numero di oggetti C che non sono stati riconosciuti come tali

In altre parole, la precisione indica quanto spesso una previsione positiva è effettivamente corretta, mentre il richiamo descrive quanto bene il modello trovi tutte le reali istanze della classe. La precisione media è infine una misura che esprime la qualità del modello tenendo conto di entrambe le metriche precedenti e coincidente con l'area sottesa dalla curva Precisione-Richiamo.

Dai grafici mostrati si nota come il modello possieda in generale un ottimo grado di precisione (93.7%) e, allo stesso tempo, un discreto indice di richiamo (59.0%). Questi risultati sono, tuttavia, dei valori d'insieme che vengono calcolati ignorando le singole classi. Approfondendo l'analisi, infatti, è evidente come il modello si comporti estremamente bene nel riconoscimento di alcune parti di automobili come la maniglia o le ruote, ma piuttosto male per quanto riguarda la discriminazione di danni. Il 50% in precisione e solo il 5.6% in richiamo evidenziano questa problematica. In particolare, la Figura 4.19 mostra, per esempio, come l'immagine etichettata in Figura 4.17 venga effettivamente esaminata dal modello, individuando correttamente tutte le parti del veicolo tranne i danni sulla parte posteriore.

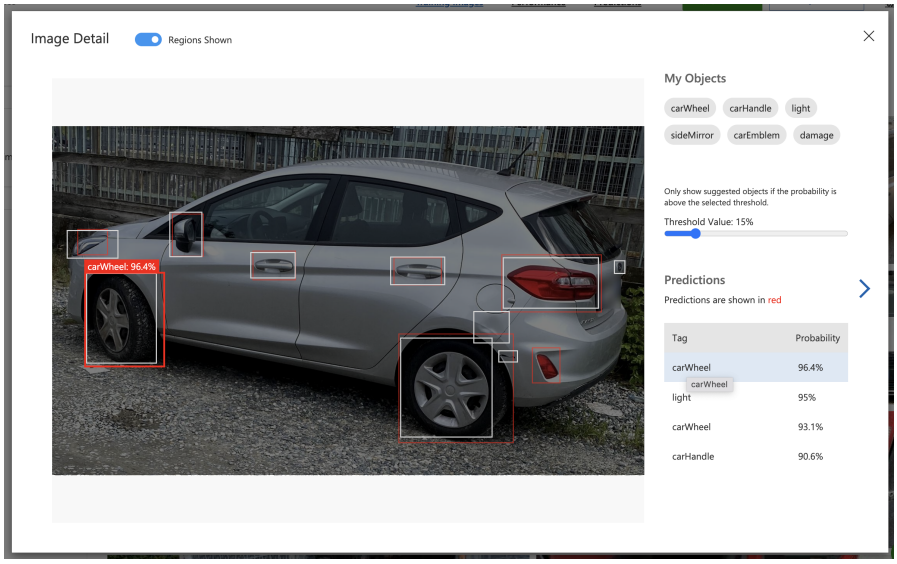


**Figura 4.19:** Esempio di output del modello Custom Vision realizzato

La spiegazione dietro questo divario nei risultati tra le diverse classi è riconducibile alla variegata natura dei danni. Mentre le maniglie hanno una struttura costante, le ruote presentano sempre lo stesso colore e la loro posizione è fissa rispetto agli altri elementi, per i danni non è così. Un danno è una classe estremamente più complessa da identificare e, per farlo, sono necessari dataset con migliaia di immagini ad alta risoluzione. Questo vincolo rende significativamente più complesso l'impiego di Custom Vision nel progetto, non avendo a disposizione una mole simile di dati classificati. Custom Vision nasce, infatti, come servizio ottimizzato per riconoscere velocemente grandi differenze tra immagini, identificando schemi macroscopici attraverso l'utilizzo di piccoli dataset. La sua capacità di Feature Extraction limitata non rende, dunque, questa piattaforma adatta per intercettare dettagli minimi.

### 4.4.3 Azure OpenAI

L'AI tradizionale basata su modelli di Computer Vision è spesso impiegata per il riconoscimento di immagini e la rilevazione di anomalie. Tuttavia, per compiti che richiedono l'identificazione di differenze minime, come piccoli danni o crepe, si scelgono solitamente soluzioni diverse, coinvolgendo spesso algoritmi di AI generativa.

L'AI generativa, in particolare i modelli di Vision-Language, è addestrata su dataset enormi e ha una capacità più sofisticata di comprendere contesti visivi complessi. Uno dei suoi punti di forza è rappresentato, appunto, dal ragionamento visivo avanzato: mentre Custom Vision si basa sul riconoscimento di schemi

predefiniti, modelli come quelli di OpenAI possono descrivere, confrontare e inferire informazioni dall'immagine, spiegando perché un'area sembra danneggiata. Tutto queste rende il modello non solo uno strumento di rilevazione, ma anche di interpretazione e approfondimento. Il sistema permette, infine, di identificare dinamicamente la tipologia di danno, senza doverla vincolare a un insieme fisso di etichette predefinite.

Cercando di sfruttare queste caratteristiche, si è sperimentata l'integrazione con uno dei suoi modelli, **GPT4 - Vision**. Chiedendo, attraverso un prompt opportuno, di analizzare le immagini fornite in ingresso, i risultati emersi dai primi test sono stati nettamente migliori rispetto a quelli rilevati con Azure Custom Vision. Si è, dunque, scelto Azure OpenAI per giungere a delle elaborazioni grafiche complete, veloci e affidabili. Nei paragrafi successivi viene mostrato nel dettaglio il processo implementativo che ha portato all'integrazione di tale modello nel sistema sviluppato.

## 4.5 Prompting, AI Studio e API Python

Una volta identificato il servizio più adatto per il riconoscimento dei danni sulle immagini dei veicoli, lo sviluppo si è focalizzato sull'interazione con esso, con particolare attenzione verso il modello GPT-4 Vision.

Il primo passo è stato la configurazione di un'istanza di OpenAI Service all'interno del portale Azure. Questo ha richiesto la registrazione del servizio nella sottoscrizione Azure aziendale, scegliendo la regione di distribuzione più adatta e il livello di pricing appropriato. Una volta creata la risorsa, è stato possibile accedere al catalogo dei modelli pre-addestrati disponibili e scegliere quello più adatto al caso d'uso. In particolare, è stato selezionato GPT-4 Turbo con Vision, in quanto l'unico a consentire l'elaborazione di immagini oltre al testo.

Dopo queste due attività iniziali, l'interazione con il modello è stata gestita attraverso **Azure AI Studio**, un ambiente che facilita l'esplorazione e la sperimentazione con i modelli AI. Tramite la sua interfaccia web è stato possibile:

- *Creare l'endpoint* - È stato configurato un endpoint personalizzato per consentire al modello di ricevere richieste. Questo endpoint rappresenta un punto di accesso specifico all'interno del servizio AI, definendo l'URL tramite cui l'applicazione può interagire con il modello.

- *Generare una chiave API* - È stata generata una chiave API per autenticare le richieste inviate al servizio. La chiave consente, dunque, di monitorare l'uso del servizio permettendo ai soli utenti o sistemi autorizzati di poter ottenere risposte dal modello.

- *Definire un prompt* - Un prompt è un input testuale fornito al modello per generare una risposta o un'azione specifica. Il prompt può essere una domanda, una frase o un insieme di istruzioni che indirizza il modello a produrre una risposta che soddisfi una determinata esigenza.

- *Testare e perfezionare il modello* - Una volta definito il prompt, sono stati eseguiti dei test con diverse immagini per analizzare la qualità delle risposte del modello. In seguito, il prompt è stato modificato e affinato più volte per ottimizzare le risposte e raggiungere il risultato desiderato. Questo processo di iterazione ha contribuito a migliorare le prestazioni del modello nell'elaborazione delle immagini.

- *Generare codice Python* - AI Studio offre, inoltre, una funzionalità che consente di generare codice Python preconfigurato per facilitare l'integrazione con il modello AI. Tale codice include l'inizializzazione di variabili come l'endpoint e l'API Key, e la configurazione dell'header e del payload della richiesta HTTP. Questa funzionalità semplifica l'attività di coding, consentendo agli sviluppatori di avere uno snippet funzionante e di poterlo adattare rapidamente alle proprie esigenze, senza dover necessariamente partire da zero.

Tra le diverse attività, prevalentemente di configurazione, quella che incide maggiormente sulla qualità del prodotto finale e che richiede un livello di competenza tecnica più elevato è la definizione del prompt. L'ottimizzazione del prompting sta assumendo un ruolo sempre più centrale negli ultimi anni, poiché determinante per ottenere risposte pertinenti e accurate dai modelli di intelligenza artificiale. Esiste, infatti, una connessione diretta tra la qualità dell'input fornito e l'output generato dal modello. L'ingegneria del prompting (Prompt Engineering) si occupa di affinare questa arte, combinando approcci empirici e metodologie strutturate. Tra le strategie emerse da questi studi, una possibile soluzione è illustrata in Figura 4.20, dove viene mostrato un approccio possibile basato su cinque passi per lo sviluppo di prompt efficaci.


**Figura 4.20:** Regole per la costruzione di prompt efficaci

In generale, si è osservato che le risposte migliori si ottengono quando il prompt è chiaro e conciso, evitando ambiguità e fornendo indicazioni precise sul risultato desiderato. Un prompt ben formulato si caratterizza tipicamente per la presenza di quattro componenti fondamentali:

- *Ruolo* - Definire esplicitamente il ruolo che il modello deve assumere è essenziale per ottenere risposte coerenti con il contesto. L'AI può operare, ad esempio, come insegnante, consulente o sviluppatore. Senza una chiara assegnazione di ruolo, l'interpretazione della richiesta potrebbe risultare generica o imprecisa.

- *Contesto* - Questa sezione consente al modello di comprendere la situazione in cui si inserisce la richiesta. Fornisce informazioni sugli attori coinvolti, sullo scenario di riferimento e sulle motivazioni alla base della richiesta, migliorando così la pertinenza della risposta generata.

- *Obiettivo* - Qui viene specificato in modo chiaro ciò che si vuole ottenere dal modello. L'output atteso può assumere, infatti, diverse forme, come un riassunto tecnico, una lista di azioni o una spiegazione semplificata. Per obiettivi complessi, è spesso utile scomporre il problema in sotto-task più semplici e intuitivi, eventualmente supportandoli con esempi.

- *Dettagli* - Questa componente consente di calibrare la risposta del modello. È possibile indicare il pubblico di riferimento, il formato desiderato o eventuali

vincoli da rispettare. Maggiore è il livello di dettaglio fornito, più la risposta sarà personalizzata.

Infine, è cruciale valutare attentamente le risposte generate e non fermarsi alla prima formulazione del prompt. Il processo di iterazione e affinamento è determinante per ottenere risultati più precisi e allineati agli obiettivi.

In base a questi principi, il prompt utilizzato nelle richieste HTTP verso Azure OpenAI è strutturato secondo il formato illustrato in Figura 4.21.

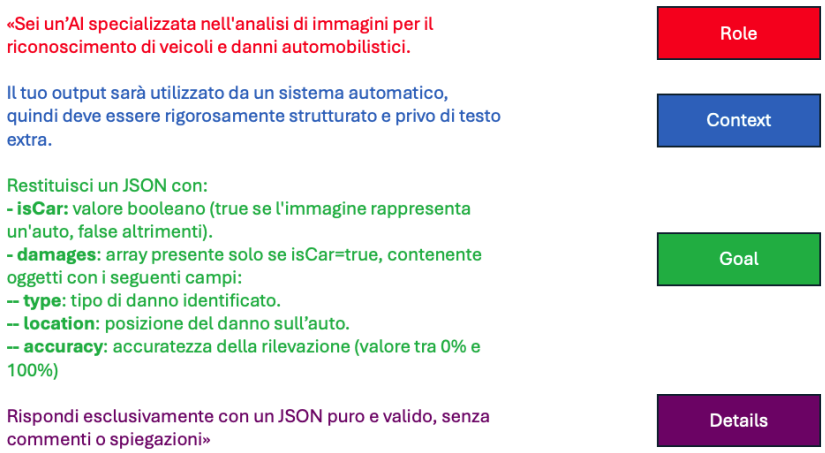


**Figura 4.21:** Prompt utilizzato nelle richieste HTTP

## 4.6 Realizzazione e Deployment del web server

### Implementazione

Dopo aver configurato l'istanza di OpenAI e analizzato i risultati ottenuti variando il prompt in ingresso, il passo successivo è stato caratterizzato dallo sviluppo di un web server, componente in grado di facilitare e standardizzare l'interazione con il servizio Azure. Attraverso il web server è possibile, infatti, effettuare un primo pre-processing delle richieste, filtrando quelle configurate erroneamente e, allo stesso tempo, gestire le risposte "grezze" derivanti da OpenAI. I vantaggi derivanti dall'introduzione di questo componente sono stati ampiamente trattati nel Capitolo 3; questo paragrafo si concentra, quindi, sugli aspetti tecnici della sua realizzazione, illustrando le tecnologie e le metodologie adottate.

Per lo sviluppo del web server è stata utilizzata la versione 3.12 di Python, in combinazione con **Flask**, un microframework open-source utilizzato nella creazione di web app. Il codice è stato gestito e versionato tramite **GitHub**, garantendo un controllo efficace sulle modifiche e una tracciabilità chiara dello sviluppo.

Al fine di garantire una struttura modulare e facilmente manutenibile, l'organizzazione interna di questo componente è stata definita secondo un'architettura a cartelle, ciascuna con una specifica responsabilità:

- *Routes* - Contiene i file in cui vengono definite le rotte dell'applicazione, attualmente limitate a un'unica endpoint.
- *Controllers* - Gestisce la logica di validazione delle richieste e l'elaborazione delle risposte.
- *Services* - Implementa la logica di business così come le interazioni con sistemi esterni (nel caso di questo progetto, l'invocazione dell'API esposta da OpenAI).
- *Exceptions* - Contiene la definizione di tutti gli errori ed eccezioni personalizzate generabili dal sistema.

L'adozione di un'architettura modulare non solo semplifica la gestione delle modifiche, ma rende anche l'applicazione facilmente estendibile, permettendo di aggiungere nuovi endpoint o funzionalità in modo rapido ed efficiente. Il file *server.py* funge, infine, da punto di ingresso per l'applicazione: si occupa di avviare il server Flask e gestire le configurazioni necessarie all'ambiente di esecuzione.

Entrando più nel dettaglio, vengono riportate in seguito due porzioni di codice estratte dai file *ai_controller.py* e *ai_service.py*.

Nel primo script viene mostrato, in particolare, il comportamento della funzione *analyze_image*. Questa funzione, invocata dalla rotta */api/ai*, gestisce la richiesta di analisi di un'immagine a partire da un URL specificato come query parameter. Verificata la presenza di tale parametro, la procedura scarica l'immagine da internet attraverso la funzione *download_image* e, in caso di successo, converte il risultato (che sia il percorso del file scaricato o il file binario stesso) in formato **base64**. Tale passaggio è necessario per includere correttamente l'immagine nel payload della richiesta HTTP. Il JSON restituito da OpenAI viene, quindi, inserito direttamente all'interno della risposta fornita dal web server con codice di uscita 200 - OK.

Il controller sviluppato, oltre ad automatizzare la normale interazione con OpenAI, fornisce inoltre una gestione avanzata degli errori, essendo in grado di intercettare una qualsiasi eccezione lanciata durante l'intero processo e, in base al tipo, fornire una risposta adeguata e un codice di errore appropriato. Tra le varie eccezioni si distinguono:

- *MissingParameterException* - Si verifica quando l'URL richiesto non è presente nella richiesta.
- *DownloadImageException* - Si verifica quando la richiesta HTTP per scaricare l'immagine restituisce un codice di stato diverso da 200.

- *APIRequestException* - Si verifica quando si presenta un problema di rete o un errore imprevisto durante il download dell'immagine.
- *MissingConfigurationException* - Si verifica quando ci sono problemi con la configurazione dell'applicazione.
- *MultipleRetriesException* - Si verifica quando viene superato il limite massimo di tentativi di analisi per la stessa richiesta.
- *Exception* - Si riferisce a errori generici che non possono essere identificati nei casi precedenti.

Nel caso si verifichi, dunque, uno qualsiasi di questi errori durante l'elaborazione della richiesta, la risposta fornita dal web server sarà del tipo:

```
{
  "status": "error",
  "code": "CODICE_ERRORE",
  "message": "MESSAGGIO_ERRORE"
}
```

**Listing 4.1:** Esempio di risposta in caso di errore

Il secondo script mostra, invece, il codice responsabile dell'effettiva invocazione dell'API esposta da OpenAI. Una volta configurati correttamente l'header e il payload della richiesta con il prompt e l'immagine in formato base64, il sistema contatta l'endpoint configurato nel paragrafo precedente, inviando una richiesta POST. Attraverso la funzione *extract_json*, l'output testuale viene convertito in un oggetto JSON prima di essere restituito al controller chiamante. Al netto del comportamento standard, è stata prevista una logica di gestione del risultato fornito da OpenAI basata sul principio dei **tentativi multipli**. È stato osservato, infatti, come, nonostante il prompt sia abbastanza chiaro nel richiedere al modello di AI esclusivamente un risultato in formato JSON, il servizio OpenAI restituisce in rarissime occasioni informazioni aggiuntive, come del testo all'inizio o alla fine della risposta. Per ovviare a questo problema, il contatore *retry_count*, insieme al ciclo while, gestisce questa problematica rieseguendo la stessa richiesta per un numero massimo di volte. Se la probabilità di ottenere una risposta non conforme alle regole specificate è inizialmente bassa, attraverso questo meccanismo la probabilità di ricevere risposte errate consecutive diventa ancora più ridotta.

```
def analyze_image():

    try:
        image_url = request.args.get('image_url')  # None if image_url is absent

        if not image_url:
            raise MissingParameterException(f"Parameter image_url is missing!", 400)

        image_url = image_url.strip('"')

        image_data = download_image(image_url)

        if isinstance(image_data, str):  # image_data is a path
            with open(image_data, 'rb') as file:
                encoded_image = base64.b64encode(file.read()).decode('ascii')
        else:  # image_data has a binary format
            encoded_image = base64.b64encode(image_data).decode('ascii')

        result = analyze_image_base64(encoded_image)

        return result, 200

    except MissingParameterException as e:
        return jsonify({"status": "error", "code": "PARAMETER_ERROR", "message": str(e.message)}), e.status_code
    except DownloadImageException as e:
        return jsonify({"status": "error", "code": "DOWNLOAD_ERROR", "message": str(e.message)}), e.status_code
    except APIRequestException as e:
        return jsonify({"status": "error", "code": "API_REQUEST_ERROR", "message": str(e.message)}), e.status_code
    except MissingConfigurationException as e:
        return jsonify({"status": "error", "code": "CONFIG_ERROR", "message": str(e.message)}), e.status_code
    except MultipleRetriesException as e:
        return jsonify({"status": "error", "code": "MULTIPLE_RETRIES_ERROR", "message": str(e.message)}), e.status_code
    except Exception as e:
        return jsonify({"status": "error", "code": "UNKNOWN_ERROR", "message": str(e)}), 500
```

**Listing 4.2:** Validazione delle richieste e gestione delle risposte

```
while retry_count < MAX_RETRIES:
        try:
            response = requests.post(ENDPOINT, headers=headers, json=payload)
            response.raise_for_status()  # Will raise an HTTPError if the HTTP request returned an unsuccessful status code

            # Get the response json content
            raw_json_content = response.json()["choices"][0]["message"]["content"]
            parsed_json_content = extract_json(raw_json_content)

            print(f"retry_count: ", retry_count, flush=True)
            return parsed_json_content

        except (requests.RequestException, JSONParsingException):
            retry_count += 1
            if retry_count >= MAX_RETRIES:
                raise MultipleRetriesException("Failed to get a valid JSON after multiple retries", 500)
```

**Listing 4.3:** Invocazione dell'API di OpenAI con tentativi multipli

### Distribuzione

Al termine della fase di implementazione, sono stati eseguiti dei primi test manuali utilizzando **Postman**, uno strumento che permette di inviare richieste HTTP e analizzare le risposte delle API in modo semplice e intuitivo. Con Postman è stato possibile simulare diverse condizioni di input e verificare il comportamento dell'applicazione, assicurandosi che la rotta prevista rispondesse nel modo in cui ci si aspettava. Una volta completati i test, il web server sviluppato è stato distribuito utilizzando il servizio Azure App Service.

Per ottenere tale risultato sono, inizialmente, state eseguite delle operazioni di setup come la creazione della risorsa Azure, la scelta del linguaggio di sviluppo e la configurazione delle variabili d'ambiente necessarie al corretto funzionamento del sistema. Una volta completata questa fase, sono disponibili per lo sviluppatore diverse opzioni di deployment in base alle proprie esigenze, ciascuna riconducibile a una di due metodologie di distribuzione: **Metodologie Manuali** e **Metodologie Contigue**. Con la prima strategia l'utente carica manualmente i file dell'applicazione senza un'integrazione automatizzata; con la seconda, invece, la distribuzione del codice avviene in modo automatico ogni volta che questo viene aggiornato.

Nel caso di questo progetto è stato utilizzato **GitHub Actions** come strumento di CI/CD per effettuare il build e il deploy dell'applicazione direttamente dal repository GitHub. Questo sistema, infatti, permette di definire dei workflow

automatici che vengono attivati in risposta a eventi specifici, come commit, push o pull request. I workflow sono definiti tramite file YAML e consistono in una serie di passaggi che consentono di testare (CI - Continuous Integration), compilare e distribuire (CD - Continuous Deployment) il codice in maniera completamente automatica, migliorando l'efficienza e riducendo il rischio di errori manuali durante il processo.

In seguito vengono illustrati gli step presenti all'interno del file YAML automaticamente generato.

```yaml
name: Build and deploy Python app to Azure Web App - dcxb1-us-pocrntl-webserver-p-010

on:
  push:
    branches:
      - main
  workflow_dispatch:

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v4

      - name: Set up Python version
        uses: actions/setup-python@v5
        with:
          python-version: '3.12'

      - name: Create and start virtual environment
        run: |
          python -m venv venv
          source venv/bin/activate

      - name: Install dependencies
        run: pip install -r requirements.txt

      # Optional: Add step to run tests here (PyTest, Django test suites, etc.)

      - name: Zip artifact for deployment
        run: zip release.zip ./* -r

      - name: Upload artifact for deployment jobs
        uses: actions/upload-artifact@v4
        with:
          name: python-app
```

```
38            path: |
39              release.zip
40              !venv/
41
42    deploy:
43      runs-on: ubuntu-latest
44      needs: build
45      environment:
46        name: 'Production'
47        url: ${{ steps.deploy-to-webapp.outputs.webapp-url }}
48
49      steps:
50        - name: Download artifact from build job
51          uses: actions/download-artifact@v4
52          with:
53            name: python-app
54
55        - name: Unzip artifact for deployment
56          run: unzip release.zip
57
58
59        - name: 'Deploy to Azure Web App'
60          uses: azure/webapps-deploy@v3
61          id: deploy-to-webapp
62          with:
63            app-name: 'dcxb1-us-pocrntl-webserver-p-010'
64            slot-name: 'Production'
65            publish-profile: ${{ secrets.
     AZUREAPPSERVICE_PUBLISHPROFILE_87236286AFDA49E6BFE4E37A00CDD0F5
      }}
```

**Listing 4.4:** Deploy su Azure App Service con GitHub Actions

Il file è caratterizzato dalla presenza di tre sezioni principali:

Nella prima sezione viene specificata la modalità di **attivazione del workflow**. In particolare, questo è stato configurato per scattare ad ogni *git push* sul branch *main* o ogni volta che il workflow stesso viene eseguito manualmente dall'interfaccia di GitHub Actions tramite l'azione workflow_dispatch.

Nella seconda sezione viene descritto il **Job di Build**. Il processo inizia specificando il sistema operativo della macchina host (Ubuntu-latest), quindi procede con il download del codice sorgente dal repository GitHub. Successivamente, viene impostata la versione di Python 3.12 e viene creato un ambiente virtuale, necessario per garantire un'installazione isolata delle dipendenze. Dopo l'attivazione dell'ambiente virtuale, il sistema installa le librerie richieste dal file requirements.txt. Il job termina con la compressione del codice dell'applicazione in un file ZIP e il caricamento di tale artefatto su GitHub Actions Artifacts, un'area di storage temporanea utilizzata per il trasferimento di file tra i job.

La terza e ultima sezione, costituita dal **Job di Deploy**, ha il compito, infine, di scaricare l'artefatto creato precedentemente, decomprimerlo e caricarlo su Azure App Service. Tramite questa sezione, ciò che si ottiene è dunque la distribuzione del codice sul cloud Azure ogni qual volta si esegue un *push* sul branch *main*.

Il paragrafo termina con un esempio di interazione, effettuata tramite Postman, con il web server distribuito. Nella figura sottostante è, dunque, mostrato l'endpoint contattato, il valore passato come parametro d'ingresso, il tempo necessario per l'intera elaborazione e la risposta ottenuta.

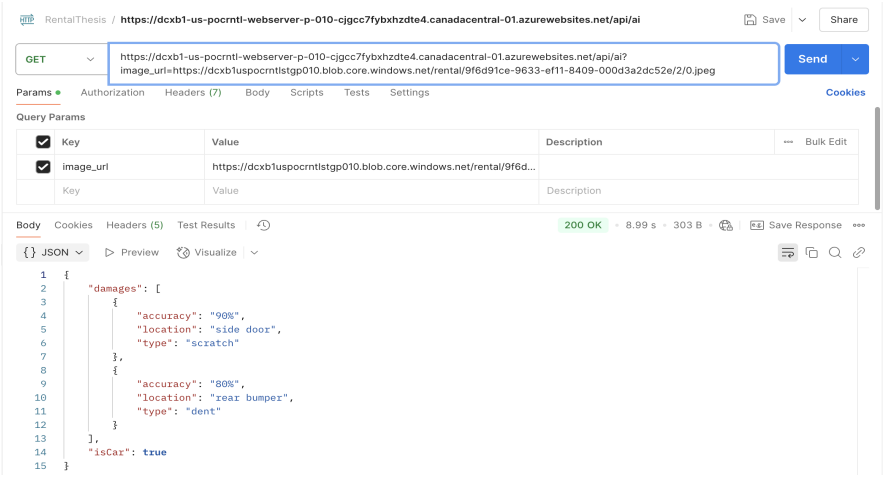


**Figura 4.22:** Esempio di interazione tramite Postman col web server distribuito

## 4.7 System Integration

Una volta che il cliente ha completato il caricamento delle foto sulla piattaforma ed è soddisfatto del numero di immagini fornite, ha la possibilità di notificare questa scelta al sistema tramite un apposito pulsante sull'app mobile. Questa azione non solo aggiorna lo stato del noleggio portandolo alla fase successiva, ma attiva anche il processo di elaborazione delle immagini su Azure OpenAI. L'ultimo step nello sviluppo del progetto ha quindi riguardato l'implementazione del meccanismo di system integration, che preleva le immagini dalla relativa cartella su Azure Blob Storage, le invia al modello AI per l'analisi e salva i risultati su Dataverse.

Per realizzare questo flusso automatizzato, è stato sviluppato, attraverso il portale **Make.PowerAutomate.com**, un flusso istantaneo attivato da Power Apps, capace di ricevere come parametri la GUID del noleggio e la fase di riferimento delle immagini. Una volta attivato il trigger, il flusso interroga Azure Blob Storage tramite un connettore dedicato, accedendo alla cartella corretta e raccogliendo i metadati delle immagini memorizzate. Successivamente, per ogni

Blob, il sistema genera una richiesta HTTP al web server ospitato su Azure App Service, inviando l'URL del file come query parameter. Il servizio di AI analizza quindi l'immagine e restituisce una risposta in formato JSON, che viene validata attraverso un modulo specifico del flusso. Questo modulo assicura che l'output rispetti uno schema predefinito, facilitando l'estrazione delle informazioni rilevanti. Tra i dati estratti figurano il campo *isCar*, che indica se l'immagine rappresenta un'automobile, e l'array *damages*, che elenca eventuali danni rilevati, specificandone il tipo, la posizione e il livello di accuratezza. Attraverso una query al database, il flusso ottiene inoltre l'ImageID corrispondente alla risorsa analizzata, utile per correlare l'output dell'analisi con l'immagine specifica memorizzata nel sistema. L'automazione termina, infine, con la gestione dei risultati:

- Se il modello ha rilevato danni, il flusso avvia un processo iterativo, creando un nuovo record nella tabella Damage per ciascun danno individuato.

- Parallelamente, il record corrispondente nella tabella Image viene aggiornato, impostando il valore booleano di *isCar* per classificare correttamente il contenuto dell'immagine.

In Figura 4.23 viene mostrato il workflow sviluppato per orchestrare l'intero processo.

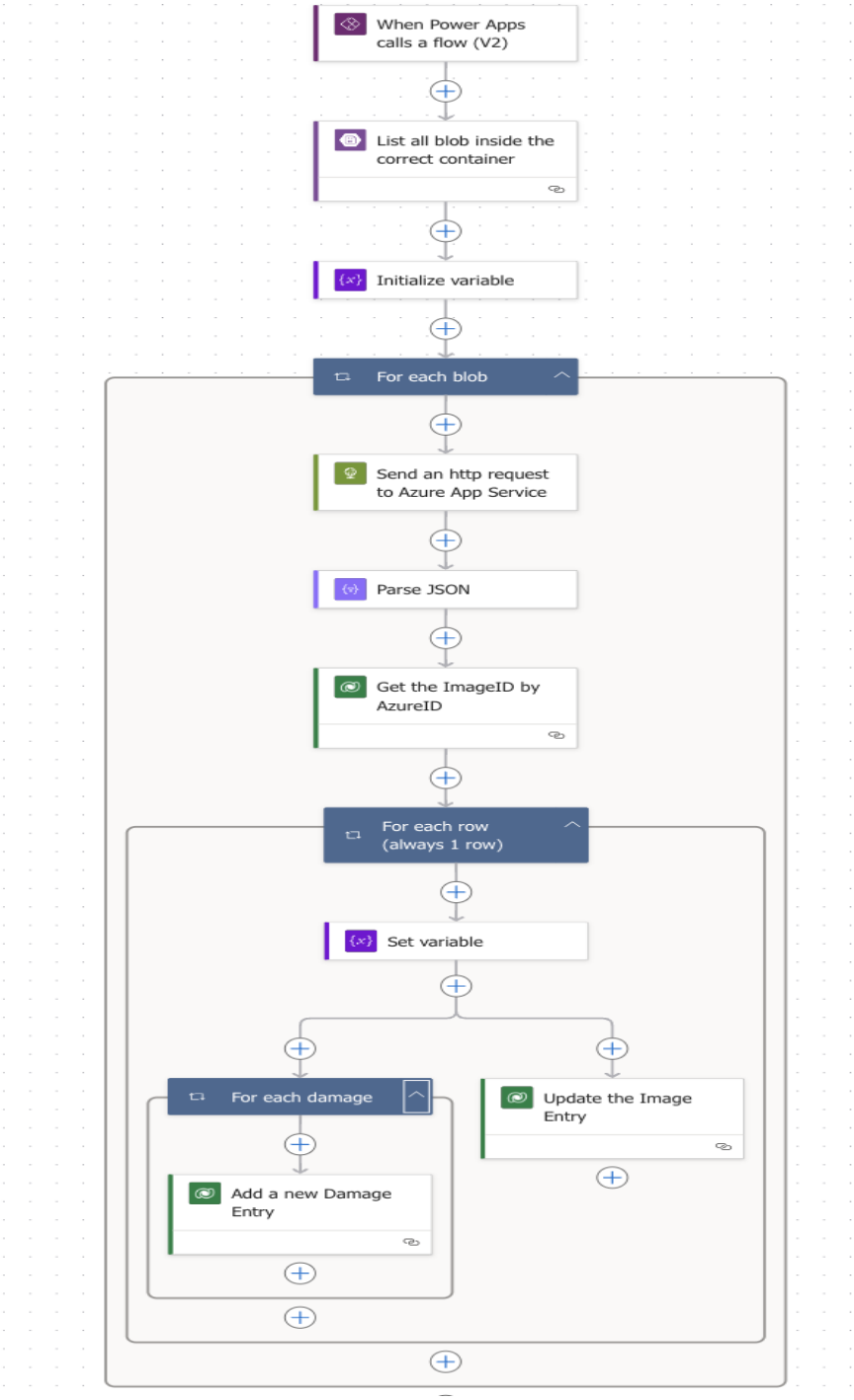


**Figura 4.23:** Workflow realizzato tramite Power Automate

Per il flusso mostrato in figura è importante sottolineare un aspetto chiave relativo al tipo di comunicazione realizzata tra i vari componenti. Nel capitolo 3 è stato evidenziato come l'introduzione di un web server abbia portato diversi vantaggi, tra cui la capacità di disaccoppiare i tempi di invocazione del servizio AI da quelli della sua effettiva elaborazione. In questo modello, il web server dovrebbe essere responsabile del salvataggio dei risultati su Dataverse, permettendo al sistema di evolvere indipendentemente dall'esito della computazione cloud.

L'implementazione attuale, tuttavia, sembra discostarsi da questo principio, adottando un approccio sincrono: il flusso attende, infatti, l'esito dell'AI prima di procedere. A prima vista, ciò potrebbe sembrare una scelta in contrasto con l'obiettivo di ridurre il vincolo temporale tra richiesta e elaborazione. Tuttavia, in questo scenario specifico, l'assenza di operazioni intermedie rende irrilevante il disaccoppiamento del salvataggio: indipendentemente dal componente incaricato (web server o flusso), l'operazione di scrittura su Dataverse può avvenire solo dopo la risposta dell'AI. Il vantaggio di un approccio asincrono emergerebbe solo in presenza di altre attività da svolgere nell'attesa, come notifiche, aggiornamenti di stato o ulteriori elaborazioni parallele. In tal caso, delegare il salvataggio al web server permetterebbe al flusso di proseguire immediatamente dopo aver inviato la richiesta, sfruttando al meglio il tempo di attesa dell'AI. Per il momento si è scelto dunque di mantenere la logica di salvataggio direttamente nel flusso, privilegiando una soluzione più semplice e immediata, ma senza rinunciare agli altri benefici architetturali forniti dal web server.

Una volta completato lo sviluppo, il flusso è stato sottoposto a un'attenta fase di test sfruttando l'intuitiva interfaccia del portale. Quest'ultima consente di monitorare in tempo reale l'esecuzione dei flussi, analizzarne le tempistiche e identificare eventuali colli di bottiglia o anomalie nel processo. Grazie a queste funzionalità, è stato possibile verificare il corretto funzionamento dell'integrazione e ottimizzare le prestazioni dell'automazione.

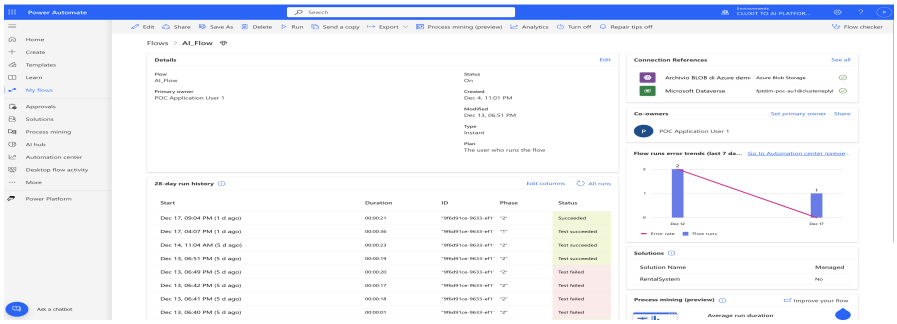


**Figura 4.24:** Portale Make.PowerAutomate.com

# Capitolo 5

# Risultati e Valutazioni

Questo capitolo illustra, innanzitutto, il processo di test a cui è stato sottoposto il sistema, per poi approfondire i risultati ottenuti e i benefici apportati all'azienda e al contesto di business di riferimento. Sulla base di questi dati, verrà analizzato il grado di soddisfacimento dei requisiti iniziali, evidenziando quelli pienamente raggiunti, quelli che presentano margini di miglioramento e quelli che, eventualmente, non sono stati completamente soddisfatti.

## 5.1 Pre-processing e Test effettuati

La fase di test non ha rappresentato soltanto il momento conclusivo di questo lavoro di tesi, ma si è rivelata un elemento essenziale anche durante la progettazione del sistema. Fin dalle prime fasi, infatti, i test hanno guidato le scelte architetturali con l'obiettivo di ottimizzare le prestazioni complessive e garantire la robustezza dell'intero ecosistema software. Per definire criteri di valutazione affidabili e oggettivi, sono stati condotti test manuali mirati, focalizzati su specifiche caratteristiche del sistema. Questi test hanno considerato sia il comportamento di singoli componenti, sia il funzionamento risultante dall'integrazione di più moduli, valutando la loro interoperabilità e risposta a condizioni operative variabili.

Una delle prime sfide incontrate è stata la mancanza di un dataset preesistente di immagini di automobili danneggiate, necessario per testare il sistema. Per ovviare a questa criticità, si è adottato un approccio pratico: la raccolta autonoma di immagini mediante fotografie scattate direttamente sul campo. Questo processo ha simulato il comportamento tipico dell'utente finale, consentendo di generare un dataset realistico e rappresentativo delle condizioni operative previste per l'applicazione.

Una volta ottenuto un numero sufficiente di immagini, è stata avviata una prima fase di pre-elaborazione per garantire la riservatezza e l'usabilità dei dati. In

particolare l'obiettivo è stato quello di rimuovere i **metadati EXIF** di ciascuna immagine. EXIF (Exchangeable Image File Format) è un formato di metadati incorporato nelle immagini digitali scattate con fotocamere, smartphone e altri dispositivi. Questi metadati forniscono informazioni aggiuntive come posizione GPS, modello della fotocamera, data e ora dello scatto. La loro presenza può pertanto rappresentare un rischio per la privacy e la sicurezza dei dati, motivo per cui è stato necessario intervenire rimuovendoli. Per questa operazione, sono state utilizzate applicazioni di messaggistica che, in modo nativo, eliminano i metadati EXIF durante l'invio delle immagini.

Terminata questa prima fase, le immagini hanno subito un ulteriore step di pre-processing con l'obiettivo di garantire l'anonimizzazione dei dati sensibili. A tal fine è stato utilizzato **Watermarkly**, un servizio online che ha permesso di offuscare automaticamente informazioni identificative, come le targhe dei veicoli e i volti eventualmente presenti nelle immagini. Questo processo non solo ha assicurato la protezione dei dati personali, ma ha anche reso il dataset idoneo ai test, preservando al contempo l'integrità visiva necessaria per le analisi successive.

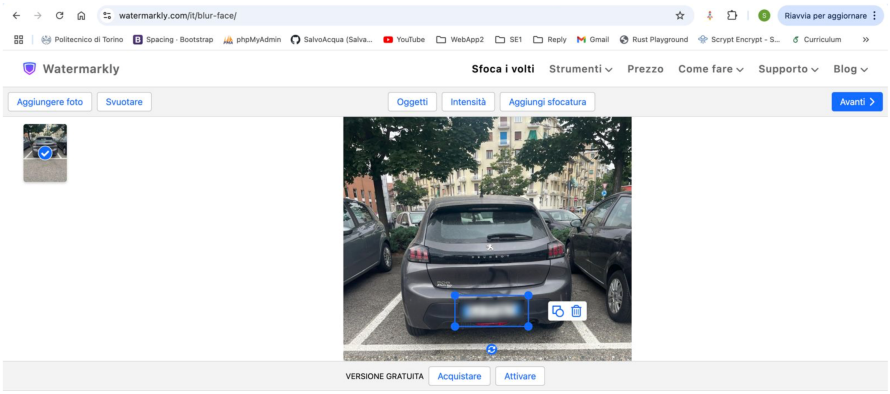


**Figura 5.1:** Sfocatura della targa tramite Watermarkly

Una volta rimossi i metadati e offuscate le informazioni sensibili, si è passati all'effettiva fase di test. Per valutare le performance del sistema sviluppato, sono state realizzate tre tipologie di test manuali:

- *Test sulla latenza del web server:* invio di richieste HTTP tramite Postman per misurare i tempi di risposta del server.
- *Test sull'affidabilità del modello AI:* analisi delle risposte restituite dal modello per verificare la correttezza delle previsioni.
- *Test sulle tempistiche del flusso Power Automate:* misurazione del tempo richiesto per completare l'intero processo di elaborazione e archiviazione dei dati.

Per testare la velocità del web server sono state eseguite, tramite Postman, 45 richieste HTTP verso questo componente. Ogni richiesta includeva, come parametro, l'URL di un'immagine differente archiviata su Azure Blob Storage. Durante il test, è stato registrato il tempo trascorso fino alla ricezione della risposta, permettendo di valutare la velocità di elaborazione del server.

I risultati sono stati memorizzati all'interno di un foglio Excel, includendo, oltre ai tempi di latenza, anche informazioni relative alla precisione delle risposte ottenute. Sono state considerate, per queste analisi, due sole tipologie di danni comuni: graffio (S - Scratch) e ammaccatura (D - Dent). Per ciascuna risposta del web server è stata compilata una tabella con i seguenti parametri:

- *True Positive (TP):* danni identificati correttamente dal modello.
- *False Negative (FN):* danni presenti nell'immagine ma non rilevati dal modello.
- *False Positive (FP):* danni erroneamente segnalati dal modello.

In Figura 5.2 sono riportati i risultati dei test ottenuti con due diversi approcci di valutazione: flessibile e severo. La distinzione tra i due metodi riguarda la gestione dei casi in cui il modello identifica correttamente un danno, ma la posizione segnalata risulta errata. Nell'approccio flessibile, tali casi vengono classificati come True Positive, considerando la corretta individuazione del danno indipendentemente dalla precisione della localizzazione, a differenza dell'approccio severo che li conteggia, invece, sia come False Positive (FP) che come False Negative (FN).

| Web Server Latency | |
|---|---|
| Image | Time (s) |
| 1 | 6,91 |
| 2 | 8,82 |
| 3 | 7,14 |
| 4 | 9,13 |
| 5 | 9,03 |
| 6 | 6,13 |
| 7 | 8,96 |
| 8 | 8,81 |
| 9 | 7,41 |
| 10 | 8,91 |
| 11 | 8,85 |
| 12 | 8,8 |
| 13 | 7,64 |
| 14 | 7,87 |
| 15 | 8,93 |
| 16 | 7,31 |
| 17 | 8,44 |
| 18 | 6,58 |
| 19 | 6,55 |
| 20 | 9,2 |
| 21 | 8,99 |
| 22 | 6,21 |
| 23 | 7,38 |
| 24 | 8,13 |
| 25 | 6,38 |
| 26 | 9,1 |
| 27 | 8,43 |
| 28 | 8,52 |
| 29 | 6,71 |
| 30 | 5,51 |
| 31 | 5,75 |
| 32 | 7,68 |
| 33 | 5,58 |
| 34 | 6,99 |
| 35 | 6,48 |
| 36 | 5,58 |
| 37 | 9,08 |
| 38 | 8,91 |
| 39 | 8,97 |
| 40 | 6,88 |
| 41 | 8,75 |
| 42 | 6,37 |
| 43 | 8,3 |
| 44 | 7,74 |
| 45 | 6,15 |

| | Flexible Approach | | | Strict Approach | | |
|---|---|---|---|---|---|---|
| Image | TP | FN | FP | TP | FN | FP |
| 1 | D + S | | | D + S | | |
| 2 | | | | | | |
| 3 | D + S | | | S | D | D |
| 4 | D + S | | | S | D | D |
| 5 | D + S | D | | D + S | D | |
| 6 | D + S + S | | | S + S | D | D |
| 7 | S | | | S | | |
| 8 | S | | | S | | |
| 9 | S | D | | S | D | |
| 10 | | D | | | D | |
| 11 | D + S | | | D + S | | |
| 12 | S | | D | S | | D |
| 13 | S | | | S | | |
| 14 | D | | S | D | | S |
| 15 | D + S | | | D + S | | |
| 16 | S | | | S | | |
| 17 | S | S | | S | S | |
| 18 | D | | S | D | | S |
| 19 | D | | | D | | |
| 20 | S | | D | S | | D |
| 21 | D + S | | | | D + S | D + S |
| 22 | | | | | | |
| 23 | D | | S | D | | S |
| 24 | S | | | S | | |
| 25 | | | | | | |
| 26 | | S | | | S | |
| 27 | S | | | S | | |
| 28 | S | | | S | | |
| 29 | D | | S | D | | S |
| 30 | D + S | | | D + S | | |
| 31 | S | | | S | | |
| 32 | S | | D | S | | D |
| 33 | | S | | | S | |
| 34 | D + S | | | D + S | | |
| 35 | S | | D | S | | D |
| 36 | | | | | | |
| 37 | | | | | | |
| 38 | D + S | | | D + S | | |
| 39 | D + S | | | D + S | | |
| 40 | | D + S | | | D + S | |
| 41 | D | S | | D | S | |
| 42 | D + S | | | D + S | | |
| 43 | D + S | | | D + S | | |
| 44 | D + S | | | D + S | | |
| 45 | S | | | S | | |

**Figura 5.2:** Metriche per la valutazione di precisione e richiamo del modello AI e tempi di risposta del web server

L'ultimo test ha esaminato, infine, il tempo richiesto affinché il flusso Power Automate completi l'intero processo, dalla fase di recupero dei metadati delle immagini sul cloud fino al salvataggio finale dei risultati forniti dal modello AI su Dataverse. A differenza dei primi due test, che esaminavano il comportamento di un singolo componente (il web server), questo test ha valutato le prestazioni di un intero sottosistema composto da parti differenti. Oltre al web server, durante l'esecuzione del flusso sono, infatti, coinvolte piattaforme come Dataverse e Azure Blob Storage. Per questo test, sono stati creati 12 noleggi fittizi, ai quali sono state associate diverse immagini di veicoli per le rispettive fasi di check-in. Per garantire un'analisi più completa e realistica, è stato deciso di associare un numero di immagini che rispecchiasse una situazione verosimile. In particolare, per 4 noleggi sono state caricate 2 immagini ciascuno, per altri 4 noleggi 3 immagini e per gli ultimi 4 noleggi, 4 immagini ciascuno. L'esecuzione dei flussi è stata avviata e monitorata tramite il portale Make.PowerAutomate.com, i cui risultati sono riportati in Figura 5.3.

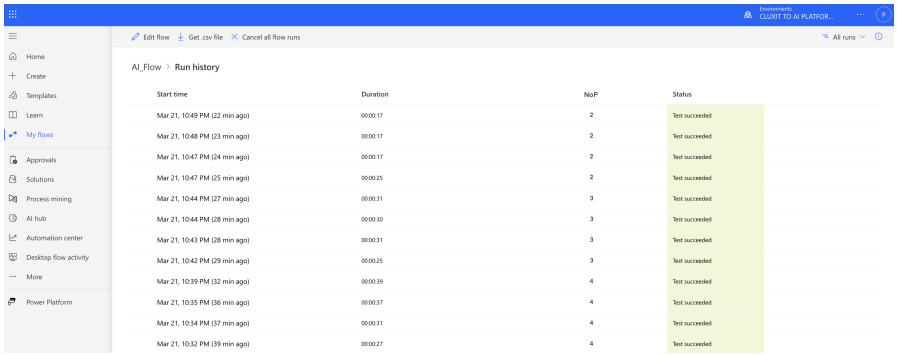


**Figura 5.3:** Tempi di esecuzione del flusso Power Automate

I risultati ottenuti da tutti questi test sono estremamente utili per valutare le performance generali del sistema ottenuto. A partire da questi, infatti, vengono estrapolate le informazioni essenziali e presentate all'interno delle sezioni successive.

## 5.2 Risultati raggiunti

Dopo aver esaminato la fase di pre-processing delle immagini e le successive attività di test, vengono presentate una serie di caratteristiche tipiche di un sistema software. I risultati ottenuti in ciascuno di questi ambiti saranno quindi analizzati, evidenziando i punti di forza della soluzione sviluppata e, al tempo stesso, segnalando eventuali criticità riscontrate.

### 5.2.1 Funzionalità

Le funzionalità esposte dal sistema soddisfano perfettamente le esigenze del contesto di riferimento. Confrontando, infatti, i risultati ottenuti con i requisiti funzionali individuati durante la fase di analisi, si può osservare come tutti i comportamenti desiderati per questo servizio siano stati realizzati con successo.

Il sistema consente, di fatto, la gestione completa delle informazioni relative ai clienti, veicoli e noleggi, lasciando la possibilità all'azienda di creare, modificare o eliminare in qualsiasi momento ciascuna istanza di queste entità. Il CRM sviluppato integra queste capacità attraverso form e tabelle interattive, corredate da pulsanti che ne facilitano la manipolazione. Dal lato utente, l'app mobile fornisce ai clienti uno strumento intuitivo per scattare e caricare foto del veicolo noleggiato direttamente sulla piattaforma. Grazie a un'interfaccia semplice e alla fluida integrazione tra Canvas App e Azure Blob Storage, l'operazione può essere completata in pochi clic. La funzionalità chiave del sistema consiste nell'analisi visiva dei veicoli. Questa abilità sfrutta la potenza di uno dei modelli di intelligenza artificiale forniti da Azure OpenAI per restituire all'azienda un elenco di possibili danni, con annesse informazioni specifiche per ciascuno di essi. Sebbene la tipologia di immagini analizzabili dal modello non sia circoscritta a contesti specifici, la qualità dei risultati è sicuramente dipendente dalla risoluzione della foto scattata, dalla luminosità e da altri fattori tecnici e ambientali. L'ispezione digitale realizzata non si limita, infine, alla sola identificazione dei danni, includendo anche l'archiviazione di questi all'interno del database. Tramite questa funzionalità è possibile, dunque, tracciare la cronologia dello stato di un veicolo, analizzando i danni rilevati in ogni noleggio e ottenendo, di fatto, uno storico sulla base dei dati registrati.

Nonostante le numerose funzionalità, il sistema presenta alcune limitazioni funzionali che, sebbene non critiche per il suo funzionamento né essenziali per il contesto di utilizzo, sono state segnalate per futuri miglioramenti. Una delle funzionalità mancanti riguarda l'impossibilità di caricare più foto contemporaneamente all'interno della piattaforma. Vincolare l'utente all'inserimento di un'immagine per volta può sicuramente impattare negativamente l'esperienza utente. Il cliente, inoltre, non avendo accesso all'area in cui sono memorizzate le immagini, non può in nessun modo visualizzare quali di queste siano state effettivamente caricate o eventualmente rimuoverle prima del processamento da parte dell'AI. Un punto debole dell'app mobile è, infine, rappresentato dall'assenza di una sezione per visualizzare tutti i noleggi effettuati dall'utente. Sebbene sia comprensibile che il cliente non possa modificare i noleggi già conclusi, per esempio, dovrebbe comunque avere la possibilità di consultarli per recuperare informazioni importanti come il veicolo e le date relative al noleggio.

### 5.2.2 Performance

Dai test effettuati è possibile estrapolare informazioni significative relative alle prestazioni del sistema, sia in termini di velocità che di affidabilità. I risultati ottenuti dimostrano l'efficienza del prodotto software realizzato, conclusione supportata dal confronto con i requisiti non funzionali identificati precedentemente.

#### Velocità

I test sulle tempistiche relative al web server e al flusso Power Automate hanno evidenziato le seguenti **latenze medie**:

- *Tempo medio di risposta del web server:* 7.69 secondi
- *Tempo medio di esecuzione del flusso:*
  - *Gestione di 2 immagini:* 19 secondi
  - *Gestione di 3 immagini:* 29.2 secondi
  - *Gestione di 4 immagini:* 33.5 secondi

Per comprendere appieno il significato di questi risultati, è necessario considerare le due tipologie di ispezioni precedentemente discusse nel capitolo 3: quelle che prevedono la presenza di un operatore al momento dell'ispezione e quelle self-service.

Nel caso delle ispezioni con operatore, l'applicazione rappresenta un valido strumento di ottimizzazione del processo di check-in e check-out per entrambe le parti coinvolte, ovvero il cliente e l'addetto alle ispezioni. L'operatore risparmia tempo poiché, non solo non deve più scattare le foto essendo un'operazione ora affidata al cliente, ma riceve anche indicazioni dettagliate sulla tipologia e sulla posizione dei danni. Questo permette all'addetto di concentrarsi sulla verifica delle anomalie segnalate anziché effettuare un controllo meticoloso dell'intero veicolo. Il processo di ispezione diventa, inoltre, più rapido ed efficiente grazie al salvataggio dei risultati in modo automatico e centralizzato. Anche il cliente beneficia di una riduzione del tempo di attesa, poiché i tempi di esecuzione del flusso uniti al rapido check dell'operatore risultano significativamente inferiori rispetto a quelli di un'ispezione manuale tradizionale. Se consideriamo quindi un tempo totale di circa 13 minuti per effettuare un controllo tradizionale, comprensivo della compilazione del modulo di ispezione, con il supporto della soluzione proposta i tempi si ridurrebbero a circa 3 minuti per l'ispezione (o anche meno) e al massimo 1 minuto per l'esecuzione del flusso. Questo si traduce in una riduzione dei tempi di check-in e check-out di oltre il 69%, con la possibilità di ottenere risparmi ancora maggiori nei casi più favorevoli.

Per quanto riguarda le ispezioni self-service, il miglioramento in termini di efficienza permane per l'operatore, il quale continua a svolgere una mera attività

di verifica. Tuttavia, per il cliente si registra un lieve incremento del tempo richiesto, in quanto deve ora occuparsi personalmente della cattura delle immagini, un'operazione precedentemente eseguita dall'addetto. Nonostante ciò, il beneficio complessivo del sistema rimane evidente, poiché l'automazione permette comunque un risparmio di tempo notevole in ogni caso per l'azienda e, nel caso di ispezione dell'operatore dal vivo, anche per il cliente.

### Affidabilità

Dal punto di vista dell'affidabilità, invece, le prestazioni ottenute sono illustrate nella seguente figura:

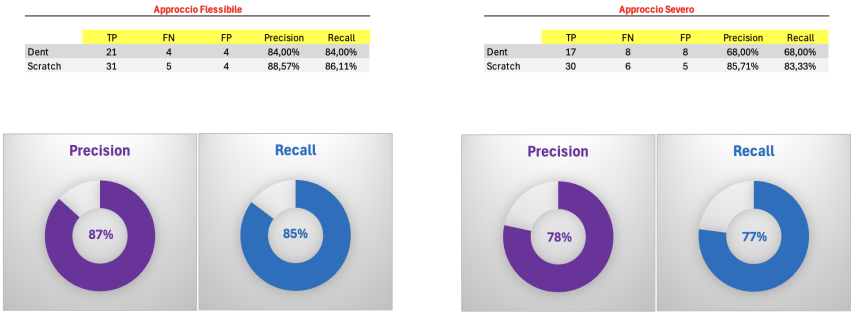


**Figura 5.4:** Performance di Azure OpenAI su due principali tipologie di danni

Teoricamente, l'affidabilità del processo migliora quando il software viene utilizzato come strumento di supporto. Questa affermazione è giustificata dalla seguente formula:

$$\mathbf{Error_{PROB}} = \text{HumanError}_{\text{PROB}} \times \text{SoftwareError}_{\text{PROB}}$$

Affinché si verifichi un errore, entrambi gli attori coinvolti (operatore e software) devono commetterlo simultaneamente. Poiché l'errore umano e quello del software sono eventi indipendenti, la probabilità di errore complessiva è data dal prodotto delle singole probabilità. Tuttavia, è importante precisare che, mentre l'errore del sistema è indipendente da quello dell'operatore, ammettere il viceversa potrebbe non essere del tutto corretto. L'operatore potrebbe, infatti, essere influenzato dai suggerimenti dell'AI, portandolo, inconsapevolmente, a confermare o ignorare un'anomalia che altrimenti avrebbe valutato diversamente. Questo fenomeno potrebbe introdurre una lieve variazione nel calcolo della probabilità di errore totale. Di conseguenza, la formula rappresenta un'approssimazione del caso reale, ma rimane comunque una stima attendibile.

Un aspetto interessante che emerge da questa relazione è che l'utilizzo del software come supporto non può peggiorare il tasso di errore complessivo del sistema. Nel peggiore dei casi, l'affidabilità rimane invariata, poiché qualsiasi valore di $\text{SoftwareError}_{\text{PROB}}$ sarà sempre compreso tra 0 e 1. Di conseguenza, l'integrazione del software rappresenta un valore aggiunto, garantendo almeno il mantenimento delle performance iniziali o, più probabilmente, il loro miglioramento.

Per concludere la trattazione relativa alle performance, si intende ora esaminare simultaneamente entrambe le caratteristiche. È stato precedentemente analizzato come tempo e affidabilità siano strettamente correlati tra di loro. Alla luce dei risultati ottenuti in termini di velocità e affidabilità, le aziende di autonoleggio che vorranno adottare questa soluzione avranno diverse possibilità di ottimizzazione:

- *Ottimizzazione dei tempi di gestione:* Se si vuole aumentare la velocità delle fasi di check-in e check-out mantenendo un valore di affidabilità costante, è possibile ridurre il tempo dedicato all'ispezione manuale. Tuttavia, accelerare il processo può comportare un maggiore rischio di errori da parte dell'operatore. Per mantenere invariata l'affidabilità complessiva del servizio, il software deve quindi supportare l'operatore nell'identificare i danni con la stessa precisione, compensando quindi l'aumento di errore legato alla velocità. Questo principio segue la logica della formula: se uno dei fattori aumenta, l'altro deve ridursi in modo proporzionale per mantenere costante l'affidabilità del sistema.

- *Ottimizzazione dell'affidabilità del servzio:* Se l'azienda di autonoleggio decide di focalizzarsi sul miglioramento dell'affidabilità, basterà mantenere invariato il tempo dedicato all'ispezione manuale. In questo caso, sarà il componente software a garantire il boost di affidabilità desiderato.

- *Approccio bilanciato (Trade-off):* Giocando sui parametri della formula, è possibile calibrare il rapporto tra velocità e affidabilità in base alle esigenze specifiche del servizio, trovando il giusto compromesso tra rapidità operativa e accuratezza del controllo.

Come ultima osservazione si vuole sottolineare come i risultati di precisione e richiamo ottenuti, così come quelli relativi alla velocità del sistema, debbano essere considerati con cautela, poiché frutto dell'analisi di poche elaborazioni. Sebbene un numero così limitato di campioni possa non rappresentare perfettamente la realtà, è ragionevole supporre che, aumentando il numero di elaborazioni, i risultati possano variare leggermente, senza però subire stravolgimenti significativi.

### 5.2.3 Scalabilità e Configurabilità

Il sistema sviluppato è stato progettato tenendo in forte considerazione il vincolo di scalabilità analizzato durante la fase di raccolta dei requisiti. L'utilizzo di Dataverse per la gestione di dati strutturati e di Azure Blob Storage per quelli non strutturati offre la possibilità di manipolare una grande quantità di informazioni in modo efficiente. Entrambi i servizi forniscono, infatti, accesso ai dati su scala globale con tempi di risposta ottimizzati, garantendo così elevata affidabilità e disponibilità. A livello computazionale, le decisioni prese in fase di progettazione hanno avuto la scalabilità come principio di riferimento. L'introduzione del web server è sicuramente una delle prime conseguenze di questa scelta, essendo quel componente in grado di scorporare le operazioni di recupero delle immagini dal cloud, analisi dei loro contenuti e salvataggio dei risultati. Questo approccio riduce il carico sui singoli componenti, ottimizzando le prestazioni del sistema e facilitando l'integrazione con eventuali moduli aggiuntivi. A sostegno di questa architettura è stato adottato Azure App Service come strumento di hosting, il quale si contraddistingue tra i servizi disponibili di questo genere per la sua capacità di scalare automaticamente, aumentando o diminuendo il numero di risorse coinvolte, al variare del carico di richieste. Attualmente il server distribuito, essendo parte di un sistema implementato come Proof of Concept (POC), utilizza un determinato piano tariffario che non permette lo scaling automatico. Qualora il progetto dovesse evolvere in un'applicazione reale su larga scala, sarà necessario effettuare l'upgrade a un piano più avanzato, in grado di garantire un livello di scalabilità adeguato alle esigenze operative.

Oltre alla scalabilità, un altro obiettivo fondamentale è stato quello di rendere il sistema altamente configurabile. Al fine di aumentarne il valore di business e garantirne l'adattabilità a diversi sotto-contesti applicativi, si è scelto di implementare alcune funzionalità in una forma essenziale, soddisfacendo il requisito primario ma limitandosi alle operazioni di base. In altre parole, le funzionalità presenti forniscono una soluzione operativa immediata, ma sono progettate per essere estese con logiche avanzate che potranno essere sviluppate successivamente per arricchire e specializzare ulteriormente il comportamento del sistema. Questo approccio consente di costruire un'architettura modulare e scalabile, facilitando l'integrazione di nuovi livelli di complessità senza dover modificare radicalmente la struttura esistente.

Grazie a questa impostazione, il progetto apre numerose opportunità di crescita e miglioramento, molte delle quali verranno introdotte nel Capitolo 6, dedicato a una panoramica riassuntiva del sistema sviluppato e all'analisi delle prospettive di sviluppo futuro.

### 5.2.4 Considerazioni ulteriori

Per completare la trattazione sui risultati ottenuti, è importante affrontare alcune tematiche trasversali che, pur non essendo state individuate come obiettivi principali durante la fase di progettazione, rappresentano comunque vantaggi significativi nel contesto applicativo del software sviluppato. Per questo motivo, si è scelto di raccogliere queste osservazioni in un unico paragrafo, così da offrire una visione più completa del sistema e del suo impatto operativo.

Uno degli aspetti più rilevanti riguarda il cambiamento del ruolo del cliente all'interno del processo di ispezione. Tradizionalmente, il cliente aveva un ruolo passivo, in attesa che un operatore effettuasse le verifiche sul veicolo noleggiato. Con l'adozione di questo sistema, invece, il cliente diventa il primo attore attivo del processo, essendo responsabile dell'acquisizione delle immagini necessarie per l'ispezione. In altre parole, scattando le foto al veicolo, l'utente si fa carico di una parte del lavoro che, nel modello tradizionale, sarebbe spettato all'addetto alle ispezioni. Da un lato, questa trasformazione potrebbe essere percepita come un onere aggiuntivo per l'utente, influenzando la sua esperienza d'uso. Dall'altro, però, introduce un importante vantaggio: il processo garantisce una maggiore tutela per il cliente stesso. Poiché il sistema richiede che le immagini vengano scattate prima di procedere con le fasi successive, queste vengono automaticamente salvate nella galleria del dispositivo dell'utente. Ciò rappresenta una forma di garanzia in caso di eventuali contestazioni o errori, ad esempio danni erroneamente attribuiti o problemi nell'associazione tra veicolo e contratto di noleggio.

Un altro aspetto fondamentale da sottolineare è che il sistema non è stato progettato per sostituire completamente il lavoro manuale dell'addetto alle ispezioni, bensì per affiancarlo e ottimizzarne le attività. Questo principio è stato un caposaldo dell'intero processo di sviluppo, in cui l'obiettivo primario è sempre stato quello di fornire uno strumento di supporto per l'operatore umano, piuttosto che rimpiazzarlo del tutto. La qualità dei risultati ottenuti e le funzionalità offerte dal sistema, tuttavia, non escludono la possibilità che alcune aziende scelgano di adottare una soluzione completamente automatizzata. In tal senso, le imprese di autonoleggio potrebbero valutare l'adozione di un modello full-tech, in cui l'intero processo di ispezione viene affidato al software, eliminando del tutto l'intervento umano. Il sistema utilizzato come strumento alternativo all'ispezione manuale non ha pertanto la pretesa di fornire un servizio altamente preciso e affidabile, ma sicuramente rappresenta un'opportunità per le aziende che possono accettare un tasso di errore più elevato, risparmiando però in costi sul personale e ottenendo ispezioni più veloci. D'altro canto, infine, il software sviluppato si presta anche a scenari ibridi, in cui l'ispezione manuale non viene completamente eliminata, ma è integrata in modo più mirato. Ad esempio, l'azienda potrebbe scegliere di affidarsi al modello AI per un primo screening dei danni e intervenire con verifiche manuali

solo nei casi in cui il sistema segnali anomalie rispetto alla fase di check-in. In questa configurazione, il numero di ispezioni condotte dagli operatori si ridurrebbe drasticamente, mantenendo comunque un livello di controllo accettabile. Tuttavia, è importante sottolineare che, anche in questo caso, il tasso di errore potrebbe essere superiore rispetto a un processo interamente manuale.

Un'ulteriore aspetto da considerare riguarda i costi legati al mantenimento della soluzione proposta. Sebbene tra i requisiti individuati, il mantenimento basso dei costi non sia stato identificato come priorità assoluta da raggiungere, lo sviluppo di questa piattaforma ha tenuto ugualmente conto dell'ottimizzazione delle risorse. Ad esempio, per contenere le spese di archiviazione, è stata evitata la memorizzazione delle immagini su Dataverse, optando per soluzioni più efficienti per la gestione di dati non strutturati. Al netto di questo, bisogna comunque evidenziare come il principale fattore di costo rimanga, tuttavia, la gestione delle licenze Microsoft per l'applicazione mobile. Attualmente l'utilizzo di Power App impone all'azienda di autonoleggio l'acquisto di una licenza individuale per ogni utente che intende accedere alla piattaforma. Questa restrizione rappresenta una limitazione importante che determina costi significativi e proporzionali al numero di utenti.

# Capitolo 6

# Conclusione

In un contesto in cui l'ottimizzazione dei processi aziendali diventa sempre più rilevante, il sistema realizzato in questa tesi si pone come un valido supporto per migliorare la gestione delle ispezioni. L'impiego di tecnologie avanzate è stato essenziale per il raggiungimento dei risultati desiderati. L'integrazione tra AI generativa e sistemi CRM, in particolare, risulta essere oramai un pattern consolidato, ampiamente apprezzato dal mercato moderno e sempre più richiesto. Le aziende, specialmente nel settore automotive, trovano in sistemi simili benefici tecnici e funzionali difficilmente replicabili con soluzioni differenti.

Il lavoro di tesi affrontato si è mosso proprio in questa direzione, portando alla progettazione e implementazione di un sistema informatico capace di automatizzare le ispezioni durante le fasi di check-in e check-out di un noleggio auto. Le valutazioni funzionali effettuate durante la fase di analisi hanno trovato una concreta controparte nel sistema realizzato, che possiede tutte le funzionalità di base necessarie a soddisfare il fabbisogno specifico di queste operazioni. Anche a livello prestazionale, i risultati sperimentali mostrano la qualità della piattaforma realizzata, pur evidenziando margini di miglioramento.

Avvicinandoci alla conclusione di questa tesi, vengono pertanto presentati possibili sviluppi futuri che rappresentano un'opportunità per il raggiungimento di un sistema sempre più completo e performante. Tra questi vi sono delle proposte funzionali che, in virtù dell'alto livello di configurabilità che si è voluto raggiungere, sono state volutamente lasciate aperte a personalizzazioni e adattamenti in base alle esigenze specifiche degli utenti finali.

Un'evoluzione interessante potrebbe prevedere la realizzazione di logiche personalizzate sul caricamento delle immagini o sul pre-processing di queste prima di inviarle al modello di AI. Il sistema potrebbe, per esempio, effettuare dei controlli specifici sul numero e sulla tipologia di immagini da memorizzare sul cloud, obbligando l'utente a scattare delle foto su tutte le principali parti dell'automobile. Ciò consentirebbe di garantire una copertura completa e uniforme delle immagini da

analizzare. La presenza del web server apre, inoltre, numerose possibilità di sviluppo. Le immagini potrebbero essere, per esempio, sottoposte a fasi di pre-processing preliminari, al fine di correggere eventuali difetti come bassa luminosità o sfocatura. In questo modo i dati forniti al modello di AI presenterebbo una qualità visiva ottimizzata, portando dunque ad analisi ancora più precise.

Un altro aspetto interessante da esplorare riguarda la gestione dei risultati generati dal modello di AI. Dopo l'analisi delle immagini del check-in, potrebbe essere implementata una logica di confronto tra la lista dei danni rilevati inizialmente e quella restituita dal modello al momento della restituzione del veicolo. La comparsa di danni in una nuova area o una maggiore stima di accuratezza per uno specifico danno rispetto a quanto segnalato in fase iniziale potrebbe indicare nuovi problemi o danni non precedentemente registrati.

Sviluppi ulteriori potrebbero riguardare, ancora, l'integrazione del sistema con CRM o ERP esterni utilizzati per gestire prenotazioni, contratti e altre informazioni relative ai veicoli. Si potrebbero, infine, integrare funzionalità di addebito automatico, in base alla differenza tra il periodo di noleggio effettivamente utilizzato e quello originariamente acquistato. Questo potrebbe semplificare la gestione dei costi, consentendo di applicare in modo preciso le tariffe in caso di ritardi o modifiche ai periodi di noleggio.

Un tema di grande rilevanza è, infine, quello delle licenze. I costi che l'azienda di autonoleggio deve sostenere per concedere ai clienti l'accesso all'app mobile, come illustrato nel capitolo 5, sono significativi e proporzionali al numero di clienti che accedono alla piattaforma. Un possibile miglioramento potrebbe essere quello di sostituire l'attuale Canvas App con una Power Page. Le Power Pages utilizzano, infatti, un modello di licenza differente, progettato per supportare scenari in cui gli utenti esterni accedono all'applicazione. Questa soluzione permetterebbe così all'azienda di acquistare pacchetti di autenticazioni mensili, lasciando solo gli utenti del CRM come systemusers. Di conseguenza, il cliente non dovrebbe più disporre di una licenza individuale, come avviene attualmente, con un notevole risparmio sui costi per l'azienda. Questo aspetto risulta particolarmente vantaggioso nel settore dell'autonoleggio, dove l'accesso alla piattaforma da parte dei clienti è sporadico e, nella maggior parte dei casi, concentrato in un singolo mese senza necessità di utilizzi ricorrenti.

In definitiva, il sistema proposto non è solo un miglioramento tecnologico, ma un punto di partenza per un nuovo modo di concepire l'efficienza aziendale nel settore dell'autonoleggio. L'AI diventa, pertanto, un alleato strategico capace di aprire nuove prospettive per il futuro del settore.

# Bibliografia


[1] Andrea Ballocchi. *Software defined vehicles: perché il futuro dell'auto passa da qui*. 2023. URL: https://tech4future.info/software-defined-vehicles (cit. a p. 1).

[2] Web Ratio. *Cos'è il Low-Code e come accelera lo sviluppo di applicazioni aziendali*. Topic on its website. URL: https://www.webratio.com/site/content/it/low-code (cit. a p. 3).

[3] IBM. *Che cos'è il low-code*. Topic on its website. URL: https://www.ibm.com/it-it/topics/low-code (cit. a p. 3).

[4] Cluster Reply. *Cluster Reply - Chi Siamo*. Blog post on Linkedin. 2023. URL: https://www.linkedin.com/company/cluster-reply/about/ (cit. a p. 4).

[5] Osservatorio CRM. *CRM e Data Analytics: insight dall'8° Osservatorio*. Rapp. tecn. Osservatorio CRM, 2023. URL: https://osservatoriocrm.it/crm-e-data-analytics-insight-8-osservatorio/ (cit. alle pp. 4, 5).

[6] Microsoft. *Power Platform Developer Documentation*. 2024. URL: https://learn.microsoft.com/it-it/power-platform/developer/get-started (cit. a p. 9).

[7] Hardit Bhatia. *Power Apps Primer: Canvas vs. Model-Driven Apps*. 2024. URL: https://global.hitachi-solutions.com/blog/canvas-vs-model-driven-apps/ (cit. alle pp. 10, 13).

[8] Microsoft. *Microsoft/Power Fx GitHub Repository*. 2024. URL: https://github.com/microsoft/Power-Fx (cit. a p. 11).

[9] ScienceDirect. *Model Driven Engineering*. URL: https://www.sciencedirect.com/topics/computer-science/model-driven-engineering (cit. a p. 11).

[10] Dev4side. *Microsoft Power Automate: 5 funzionalità per creare un workflow*. Blog on its website. URL: https://www.dev4side.com/blog/microsoft-power-automate (cit. a p. 12).

[11] Giada Pezzini. *Microsoft Dynamics 365: what it is, who it's for, and why it matters*. 2019. URL: https://www.lsretail.com/resources/dynamics-365-new-way-productive (cit. a p. 15).

[12] Cegeka. *Microsoft Dynamics 365 - Sviluppa il vero potenziale della tua azienda*. Blog on its website. URL: https://www.cegeka.com/it/solutions/dynamics365/soluzioni/microsoft-dynamics-365 (cit. a p. 15).

[13] Dev4side. *Azure Cognitive Services: What they are, pricing, and examples*. Blog on its website. URL: https://www.dev4side.com/en/blog/azure-cognitive-services (cit. a p. 21).

[14] Valerio Mariani. *Cognitive services: cosa sono i servizi cognitivi basati su AI*. 2022. URL: https://www.sergentelorusso.it/cognitive-services (cit. a p. 21).

[15] Daniele Grandini. *Azure OpenAI: cos'è e perché l'intelligenza artificiale generativa non è un rischio ma un alleato per l'evoluzione del lavoro*. URL: https://www.4ward.it/blog/azure-openai-cos%C3%A8-e-perch%C3%A9-lintelligenza-artificiale-generativa-non-%C3%A8-un-rischio-ma-un-alleato-per-levoluzione-del-lavoro (cit. a p. 23).