Politecnico di Torino

Politecnico di Torino

Laurea Magistrale in Ingegneria Informatica

2024-2025

Sessione di laurea Luglio 2025

Integrazione tra gestionale aziendale e marketplace digitali: automazione e ottimizzazione del ciclo di vendita online

Relatori:

Alberto Monge Roffarello

Candidato:

Edoardo Morello

Indice

Capitolo 1

Introduzione

1.1 Che cos'è un e-commerce

Il termine e-commerce si riferisce all'insieme delle transazioni commerciali condotte attraverso piattaforme digitali, principalmente via Internet. Questa modalità di scambio rappresenta una delle trasformazioni più significative nel panorama economico degli ultimi decenni, in quanto ha completamente rivoluzionato il modo in cui beni e servizi vengono offerti, venduti e distribuiti. Secondo l'OECD (Organization for Economic Co-operation and Development), il commercio elettronico comprende la "vendita o l'acquisto di beni o servizi condotti via Internet o altre reti informatiche, in risposta a comandi digitali specifici" [1]. L'attività di un e-commerce comprende tutti i vari processi di vendita: presentazione dei prodotti, vendita, gestione del pagamento, spedizione e supporto al cliente.

Le piattaforme di e-commerce possono essere classificate in diverse categorie, le cui principali sono B2C (business-to-consumer), B2B (business-to-business), C2C (consumer-to-consumer). Tra i vari modelli, il più diffuso è il B2C: le aziende vendono direttamente i loro prodotti o servizi ai consumatori finali attraverso diversi portali digitali. La struttura del B2C si basa su dinamiche di consumo simili a quelle del commercio tradizionale, con la mediazione di tecnologie che ne amplificano la portata, la velocità e la scalabilità. Le applicazioni B2C si presentano sotto forma di siti web o applicazioni che integrano cataloghi digitali, sistemi di pagamento elettronico, tracciamento degli ordini spediti, recensioni dei clienti e strumenti di assistenza virtuale.

L'espansione del commercio elettronico B2C è stata favorita dalla diffusione globale di Internet, dei dispositivi mobili e dal miglioramento delle infrastrutture logistiche. Secondo Statista, le vendite globali al dettaglio tramite e-commerce hanno raggiunto circa 5,8 trilioni di dollari nel 2023, con proiezioni di crescita fino a oltre 8 trilioni di dollari entro il 2027 [2]. Inoltre, il comportamento d'acquisto

dei consumatori è stato significativamente influenzato da fattori come la personalizzazione dell'offerta, l'immediatezza dell'accesso a una vasta gamma di prodotti e l'aumento della fiducia verso i pagamenti digitali.

Il modello B2C presenta comunque alcune sfide, tra cui la gestione della concorrenza globale, l'ottimizzazione dell'esperienza utente, la fidelizzazione del cliente e la protezione dei dati personali. Questi aspetti rendono il commercio elettronico B2C un ambito dinamico, che richiede continui adattamenti e miglioramenti tecnologici, organizzativi e strategici da parte delle imprese coinvolte.

1.2 Obiettivi della tesi

L'obiettivo principale della tesi consiste nell'integrazione di un sistema gestionale esistente, il software Zucchetti MAGO, con diversi e-commerce e marketplace, quali per esempio IBS, BricoBravo, E-Price, Shopify e, in particolare, Amazon. Tale integrazione ha come finalità la creazione di un'applicazione omnichannel in grado di reperire le vendite dai molteplici canali e incanalare la gestione nell'erp aziendale. Il lavoro da svolgere, nel conseguimento dell'obiettivo del progetto, consiste nell'automazione della maggior parte delle attività di un e-commerce: caricamento e aggiornamento del catalogo, preceduto dalla ricerca e dal recupero delle informazioni principali degli articoli, quali per esempio descrizione, categoria merceologica, prezzo di vendita verso il cliente, codice iva, immagini; evasione degli ordini, processo che spazia dal recupero delle informazioni inerenti alla vendita, la creazione della documentazione necessaria alla spedizione e la creazione di quest'ultima tramite vettori selezionati; individuazione di KPI rilevanti per il monitoraggio e il controllo dell'attività, finalizzati a orientare i manager verso le giuste scelte in ottica di business.

1.3 Struttura della tesi

La tesi è strutturata in modo da fornire una visione completa e logica degli aspetti inerenti l'integrazione del gestionale con i vari marketplace, analizzando, per ogni fase, i problemi e le soluzioni proposte e adottate:

- **Capitolo 1**: introduzione agli e-commerce. Obiettivi e struttura della tesi
- **Capitolo 2**: situazione iniziale e motivazione del progetto
- **Capitolo 3**: integrazione tra sistema gestionale e diversi marketplace
- **capitolo 4**: informazioni sugli articoli venduti online
- **Capitolo 5**: interfacciamento con i diversi fornitori

- **Capitolo 6**: gestione e evasione degli ordini cliente
- **Capitolo 7**: monitoraggio dell'attività
- **Capitolo 8**: discussioni
- **Capitolo 9**: conclusioni

Capitolo 2

Situazione iniziale e motivazione del progetto

2.1 Problema: molti articoli su più marketplace

L'azienda oggetto della tesi si occupa principalmente della gestione e distribuzione di un ampio catalogo di articoli eterogenei, principalmente prodotti videoludici e merchandise collegato, musica, libri e oggettistica varia, con un volume che supera le 900.000 referenze e che sarà destinato a crescere con l'uscita di novità e grazie alla continua ricerca da parte dell'azienda di nuovi fornitori. Ogni prodotto può variare per fornitore, condizioni commerciali, disponibilità e specifiche tecniche, e dev'essere reso disponibile su una pluralità di canali di vendita digitali (come Amazon, IBS, Shopify, ecc.). L'esigenza di mantenere sincronizzati, aggiornati e correttamente categorizzati tali prodotti su ciascun canale ha comportato una serie di criticità operative e sistemiche.

I marketplace in questione impongono standard differenti per la pubblicazione e l'aggiornamento dei prodotti, sia in termini di struttura dei dati (Amazon richiede il formato JSON [3], eBay quello XML [4] mentre Shopify utilizza query GraphQL [5]) sia per quanto riguarda i requisiti obbligatori e le regole di validazione dei contenuti. Ad esempio, la presenza di attributi specifici, immagini ad alta risoluzione, gerarchie categoriali distinte e codifiche proprietarie rende impossibile adottare una logica unificata di caricamento senza un'opportuna fase di adattamento.

A questo si aggiunge la disomogeneità delle interfacce API messe a disposizione dai diversi marketplace, ognuna con proprie logiche di autenticazione, modalità di invio dati, risposta ed eventuali limiti di chiamata (rate limits). Questi limiti, se non gestiti correttamente, possono portare a interruzioni nei flussi di pubblicazione o aggiornamento, con conseguenti impatti negativi sulla visibilità e disponibilità dei prodotti online.

Dal punto di vista gestionale, l'ERP adottato (MAGO) non è strutturalmente predisposto per supportare nativamente tali processi multicanale, né per contenere tutti i metadati richiesti dai marketplace. La struttura relazionale del database, pensata per un uso amministrativo e contabile, ha richiesto l'estensione tramite nuove colonne, tabelle e logiche per ospitare informazioni aggiuntive non previste dallo schema originale. Senza un sistema centralizzato capace di armonizzare i dati e distribuire in modo efficiente i flussi informativi verso l'esterno, la gestione manuale delle pubblicazioni sarebbe risultata insostenibile.

Infine, l'enorme quantità di articoli da processare, unita alla necessità di aggiornamenti frequenti (prezzi, disponibilità, descrizioni), ha evidenziato i limiti di scalabilità dei processi tradizionali e la necessità di un sistema automatizzato, in grado di gestire carichi elevati e sincronizzazioni distribuite nel tempo.

2.2 Motivazioni: grandi moli di dati eterogenei, errori umani

Le criticità operative descritte nella sezione precedente trovano origine in una combinazione di fattori strutturali e tecnici, che nel loro insieme concorrono a rendere particolarmente complessa la gestione del catalogo prodotti in un contesto multi-marketplace.

Il primo elemento di complessità è rappresentato dalla dimensione del catalogo. La gestione quotidiana di oltre 900.000 articoli impone un'attenzione continua alla qualità e coerenza dei dati, sia nella fase di inserimento iniziale sia durante le attività di aggiornamento. Tali volumi rendono impraticabile qualsiasi approccio manuale, non solo per motivi di tempo, ma anche per il rischio intrinseco di introduzione di errori sistematici.

Il secondo fattore rilevante è l'elevata eterogeneità dei prodotti e delle rispettive informazioni. Ogni articolo può presentare combinazioni uniche di attributi, immagini e schede tecniche. I diversi marketplace non solo richiedono formati e strutture distinte, ma attribuiscono anche significati differenti agli stessi concetti (per esempio, la nozione di 'categoria' o di 'SKU' può variare da una piattaforma all'altra). Questa frammentazione semantica impone una mappatura accurata e una normalizzazione preventiva dei dati prima della loro diffusione.

A questi aspetti si aggiunge la presenza costante di interventi umani nei flussi operativi, soprattutto nella fase iniziale del progetto, quando molte delle attività di codifica, caricamento e revisione venivano eseguite manualmente. L'intervento umano, sebbene inizialmente necessario, si è rivelato una delle principali fonti di errore: dalla duplicazione di informazioni all'errata associazione tra codici articolo e fornitori, fino alla pubblicazione incompleta o scorretta di dati critici per il posizionamento dei prodotti online.

Infine, la mancanza di un'infrastruttura informatica già predisposta per la sincronizzazione multi-canale ha costretto a operare all'inizio in modo reattivo, adattando le singole procedure caso per caso. Questo approccio incrementale ha evidenziato rapidamente la necessità di una strategia automatizzata per garantire scalabilità, affidabilità e qualità complessiva.

2.3 Soluzione: integrazione tra gestionale e piattaforme, automatizzazione procedure

In risposta alle problematiche evidenziate, ho progettato e implementato un sistema di integrazione centralizzato tra il gestionale aziendale (MAGO) e i principali marketplace digitali su cui opera l'azienda. La soluzione si è articolata in 3 vie principali: estensione strutturale del database gestionale, sviluppo di procedure di integrazione automatizzate e definizione di un sistema di monitoraggio e controllo continuo.

Dal punto di vista infrastrutturale, ho esteso il database di MAGO mediante l'aggiunta di colonne custom e la creazione di tabelle ad-hoc in grado di ospitare le informazioni per ciascun canale di vendita, mantenendo la compatibilità con la struttura originaria. Questa scelta ha consentito di centralizzare i dati relativi ai prodotti, arricchendoli di metadati specifici per marketplace (categorie, tag, immagini, identificativi), senza alterare le funzionalità contabili e amministrative del gestionale.

Le procedure di integrazione sono state sviluppate come moduli indipendenti, ciascuno specializzato nell'interfacciarsi con le API del rispettivo marketplace. Tali moduli si occupano di estrarre i dati relativi agli articoli dal database interno, trasformarli nel formato richiesto (JSON, XML, GraphQL o semplicemente inserirli in un file csv) e inviarli attraverso chiamate API asincrone. Ho inoltre implementato dei meccanismi di gestione dei rate limits per ogni piattaforma, distribuendo le chiamate nell'arco della giornata e riducendo il carico simultaneo sul gestionale. Ho posto particolare attenzione anche all'ottimizzazione degli algoritmi relativi alle chiamate API per sfruttare al massimo le opportunità fornite da ogni marketplace, adottando soluzioni specifiche nei casi in cui i rate limits erano stringenti.

L'intero sistema è supportato da una rete di procedure automatiche schedulate, che eseguono le operazioni ricorrenti (aggiornamento prezzi e disponibilità, pubblicazione nuovi articoli, sincronizzazione ordini) in modo distribuito e controllato. Ogni operazione è accompagnata da un processo di logging dettagliato, con registrazione su file distinti per ogni procedura e marketplace, allo scopo di facilitare il monitoraggio e l'analisi delle performance.

In aggiunta, ho implementato un sistema di notifica automatica via email per segnalare errori critici o malfunzionamenti anomali, in modo da consentire al team

tecnico di intervenire tempestivamente e limitare eventuali impatti negativi sulla presenza online dei prodotti.

Questa architettura modulare e scalabile ha permesso di raggiungere un'elevata affidabilità operativa, ridurre drasticamente gli errori umani e assicurare una gestione uniforme dei dati su tutti i canali digitali coinvolti. L'automazione dei processi quotidiani ha rappresentato un fattore determinante per garantire la continuità operativa e la crescita sostenibile dell'attività e-commerce dell'azienda.

Capitolo 3

Integrazione tra gestionale e varie piattaforme

3.1 Gestionale utilizzato

Il sistema gestionale adottato come nucleo informativo e operativo del progetto è MAGO, una soluzione ERP modulare e scalabile, largamente impiegata nelle piccole/medie imprese italiane. MAGO fornisce strumenti per la gestione amministrativa, contabile e logistica dell'azienda, integrando al proprio interno processi quali ordini, fatturazione, magazzino e anagrafica articoli e clienti.

Tuttavia, pur offrendo un'architettura robusta per la gestione interna dell'azienda, MAGO non è progettato nativamente per affrontare la complessità dell'integrazione multi-marketplace. In particolare, le sue tabelle standard non prevedono attributi specifici richiesti dai canali e-commerce moderni (es: ASIN, immagini multiple, mappature di attributi secondo tassonomie esterne, identificativi vari). Per rispondere a tali esigenze, è stata necessaria un'estensione strutturale del database, con l'aggiunta di nuove colonne a tabelle già esistenti (es: nella tabella 'Items', la tabella principale contenente le informazioni di un articolo, sono state aggiunte fino a 6 colonne 'UrlImage' per il salvataggio di più immagini e una decina di colonne 'Tags' per il salvataggio di tag o attributi vari) e con la creazione di nuove tabelle relazionate a quelle già esistenti, con campi personalizzati secondo lo standard di interoperabilità definito internamente.

Un elemento centrale nella fase di integrazione tra gestionale e marketplace è stata la progettazione e l'implementazione della tabella 'OrdiniCliente', creata adhoc per la gestione degli ordini dei clienti provenienti da tutte le varie piattaforme di e-commerce. Tale tabella costituisce un punto di convergenza unificato per tutti i flussi di vendita, permettendo una visione e una gestione centralizzata e coerente di tutti i dati. Ho strutturato la tabella in modo da contenere tutte le informazioni

rilevanti associate a un ordine: identificativo dell'ordine, identificazione dell'articolo ordinato, quantità ordinata, prezzo di vendita e di spedizione, IVA, riferimento all'eventuale fornitore da contattare per l'approvvigionamento del prodotto, numero del DDT (Documento Di Trasporto), numero di tracciamento della spedizione, riferimento ai dati anagrafici e logistici del cliente finale, stato dell'ordine (da spedire, spedito, rimborsato, annullato). Questa soluzione ha reso possibile non solo la tracciabilità completa del ciclo di vita di un ordine, ma anche l'automatizzazione di molte operazioni successive: generazione documenti, aggiornamenti di stato, modifiche di dati come quantità e indirizzo finale, gestione dei resi. Inoltre, la presenza di questo livello intermedio di astrazione ha facilitato la realizzazione di strumenti analitici e reportistici comuni a tutte le piattaforme, garantendo uniformità nei processi decisionali aziendali.

L'integrazione con le piattaforme esterne è stata realizzata senza modificare il core del gestionale, mantenendo un disaccoppiamento logico tra i dati aziendali e i dati relativi ai vari e-commerce. Questo approccio ha permesso di salvaguardare l'integrità del sistema, semplificare gli aggiornamenti del gestionale e facilitare la manutenzione evolutiva delle procedure automatiche.

Infine, l'utilizzo dei microservizi di MAGO è stato un supporto significativo per alcune funzioni transazionali, come la generazione documentale (documenti di trasporto, fatture) e la consultazione real-time delle giacenze. Tali servizi sono stati integrati nei flussi di automazione, migliorando la coerenza tra la gestione interna e le operazioni di vendita nei vari e-commerce.

3.2 Piattaforme di e-commerce scelte e varie metodologie di interfaccia

Il progetto ha richiesto l'integrazione con tre principali piattaforme e-commerce: Amazon, eBay e Shopify (oltre a queste, sono state prese in considerazione anche IBS, E-Price e BricoBravo, le quali utilizzano API simili a quelle di Amazon). Queste piattaforme, pur condividendo l'obiettivo comune della vendita online di beni, adottano metodologie di interfaccia profondamente diverse tra loro: rispettivamente API RESTful, interfaccia XML via feed e Web Services, e API GraphQL. Tale eterogeneità ha imposto un'analisi specifica delle logiche di ciascun sistema e l'adozione di tecniche di integrazione dedicate.

3.2.1 Amazon: RESTful APIs

Amazon espone le proprie API tramite la Selling Partner API (SP-API), un insieme modulare di interfacce REST protette da autenticazione OAuth 2.0. Le SP-API consentono la gestione dell'intero ciclo di vita dei dati: dalla creazione del catalogo

prodotti all'elaborazione ordini, passando per la logistica e la reportistica [3]. La comunicazione con il sistema avviene in formato JSON, attraverso chiamate asincrone per la gestione dei rate limits, diversi per ogni endpoint. La complessità principale risiede nella gestione della sicurezza, in quanto le richieste devono essere firmate con AWS Signature Version 4 [6], e nella necessità di orchestrare una pipeline di invio e polling per verificare l'elaborazione asincrona dei dati caricati.

3.2.2 eBay: Web Services basati su XML

eBay espone un'architettura API di tipo SOAP/XML [7], disponibile tramite la Trading API, la Finding API e la Inventory API, che coprono diverse aree funzionali del marketplace. Il formato dei messaggi è strutturato secondo uno schema XML conforme agli standard WSDL/XSD, e le operazioni richiedono un'attenta validazione sintattica e semantica dei messaggi di input [4]. Il caricamento dei prodotti, per esempio, avviene attraverso l'invocazione di chiamate come la 'AddFixedPriceItem', che prevede l'invio di dati incapsulati in complesse gerarchie XML. La struttura dei messaggi impone una rigida coerenza tra codifiche e categorie. Anche la gestione delle risposte, solitamente contenenti codici di errore annidati, ha richiesto un attento parsing e logging dettagliato.

3.2.3 Shopify: API GraphQL

Shopify rappresenta un caso tecnologicamente avanzato, grazie all'adozione delle API GraphQL, che permettono richieste flessibili e performanti. Le API sono organizzate in due principali ambienti: l'Admin API per la gestione dei contenuti (prodotti, ordini, clienti, spedizioni) e le Storefront API per la gestione della navigazione e dell'esperienza utente (parte non compresa nel progetto di tesi) [5]. L'approccio GraphQL consente di definire in modo preciso la struttura delle informazioni desiderate, evitando richieste ridondanti o poco utili e permettendo di recuperare solo ciò che è veramente utile. Esistono 2 tipi di chiamate: le query permettono di effettuare interrogazioni e sono paragonabili ai metodi GET delle API REST, mentre le mutation permettono di creare, modificare o eliminare i dati, similmente a come accade con i metodi POST, PUT e DELETE delle chiamate REST. La sintassi di query e mutation è fortemente tipizzata e deve aderire agli schemi forniti da Shopify. Questo consente di ridurre la latenza complessiva e migliorare l'efficienza dell'integrazione, specialmente in presenza di relazioni complesse tra prodotti, varianti, immagini e collezioni.

3.2.4 Strategia generale di interfacciamento

Per l'implementazione delle procedure di integrazione ho seguito un approccio ETL (Extract, Transform, Load), strutturato come segue:

- **Extract**: lettura dei dati dal gestionale MAGO, tramite query SQL su tabelle native ed estese ad-hoc;
- **Transform**: normalizzazione, mappatura e perfezionamento dei dati per aderire ai requisiti della piattaforma destinazione;
- **Load**: invio dei dati tramite API, con gestione dello stato delle operazioni, degli errori e dei rate limits.

Ogni procedura che ho sviluppato include un sistema di logging separato, che registra le varie operazioni effettuate, i payload inviati e le risposte ricevute, oltre a un sistema di notifiche automatiche via email per errori critici o anomali. Infine, per evitare sovraccarichi al gestionale o alle API esterne, le esecuzioni delle varie procedure sono state distribuite temporalmente durante l'arco della giornata mediante uno schedulatore di attività.

Capitolo 4

Informazioni sugli articoli venduti

4.1 Analisi sulle informazioni necessarie degli articoli

L'integrazione efficace tra gestionale e marketplace richiede la disponibilità e l'adeguata strutturazione di un insieme minimo di informazioni per ogni articolo destinato a essere venduto online. La definizione di questi dati non è uniforme, ma varia in base ai requisiti imposti dalle diverse piattaforme di e-commerce, dalle policy locali e dalle best practices del settore. Gli attributi indispensabili possono essere classificati nel modo seguente:

- **Identificativi univoci**: codice interno dell'articolo, SKU (Stock Keeping Unit), codice a barre (EAN). Questi elementi sono fondamentali per la corretta pubblicazione e tracciabilità del prodotto.
- **Informazioni descrittive**: titolo del prodotto, descrizione estesa, caratteristiche tecniche, colori, dimensioni. La qualità di queste informazioni influenza significativamente la visibilità delle inserzioni.
- **Immagini**: fotografie ad alta risoluzione, aderenti alle linee guida di formato e proporzione dei diversi marketplace. Per esempio, Amazon richiede immagini con sfondo bianco, copertura dell'85% dell'area disponibile e altre qualità [8].
- **Informazioni commerciali**: prezzo di vendita, eventuale prezzo promozionale, disponibilità di magazzino, tempi di spedizione stimati.
- **Categorie e attributi**: associazione del prodotto a una o più categorie di vendita e compilazione di attributi richiesti da ogni categoria, ad esempio il

formato di un cd, la compatibilità con gli accessori; inoltre sono importanti anche i tag per identificare le categorie di appartenenza di un articolo.

- **Informazioni logistiche**: il peso è l'attributo più importante ai fini della futura creazione di una spedizione, ma sono di interesse anche altre informazioni come dimensioni e eventuali restrizioni sulla spedizione, come la presenza di batterie.

- **Informazioni fiscali**: aliquota IVA, necessaria anche per i report di vendita.

Un aspetto centrale nell'organizzazione del catalogo articoli è stata la selezione dell'identificativo univoco da utilizzare come riferimento principale. In questo progetto è stato adottato l'EAN (European Article Number) come identificativo standard degli articoli, dal quale sono partito per la creazione dello SKU nei vari marketplace (Stock Keeping Unit). L'EAN è un codice numerico univoco, standardizzato a livello internazionale (ISO/IEC 15420), destinato a identificare specificamente un prodotto commerciale lungo tutta la catena distributiva [9]. Grazie alla sua diffusione globale e alla sua compatibilità con i sistemi di tracciamento automatizzati (scansione barcode), l'utilizzo dell'EAN ha permesso di garantire una mappatura precisa e non ambigua dei prodotti all'interno del database gestionale e nelle comunicazioni verso i marketplace.

Molti marketplace, tra i quali Amazon, IBS e altri, basano i propri sistemi di catalogazione su identificativi di prodotto già esistenti. In particolare, piattaforme come Amazon dispongono di cataloghi interni estremamente dettagliati: fornendo un identificativo come l'ASIN (Amazon Standard Identification Number), oppure un EAN valido, è possibile associare automaticamente un articolo alle informazioni già presenti nel database di Amazon. Queste informazioni comprendono il titolo del prodotto, la descrizione, il peso, le immagini ufficiali, le caratteristiche tecniche, senza la necessità di riproporle manualmente.

Questo approccio, noto come listing su catalogo esistente, consente di ridurre notevolmente il tempo necessario alla pubblicazione dei prodotti, abbassare notevolmente il rischio di errori di descrizione e garantire una coerenza informativa tra venditori diversi che offrono lo stesso prodotto.

In fase progettuale ho quindi previsto, per ciascun marketplace, una logica condizionale per il caricamento degli articoli:

- dove esiste un catalogo interno e il prodotto è già presente (Amazon, IBS, EPrice) ho utilizzato il solo identificativo;

- diversamente (Shopify, eBay) ho fornito tuttle le informazioni necessarie alla creazione della scheda prodotto.

Nel secondo caso, l'assenza o incompletezza di uno o più di questi dati (come il mancato recupero di immagini) può comportare rifiuti di pubblicazione, penalizzazioni nella visibilità dell'inserzione o, nei casi più gravi, la sospensione dell'account venditore. Per tale ragione, durante la fase progettuale, è stato previsto un sistema di verifica preliminare della completezza dei dati prima dell'invio verso le varie piattaforme, al fine di minimizzare errori. Gli articoli che presentano lacune su alcune informazioni importanti vengono identificati e segnalati tramite un report ai manager.

4.1.1 Focalizzazione sui prezzi

La determinazione di un prezzo di vendita corretto rappresenta uno degli aspetti più critici nella gestione di un catalogo multicanale. Le diverse piattaforme di e-commerce applicano regole specifiche sulla coerenza e la competitività dei prezzi, rendendo necessario un approccio flessibile e dinamico nella loro definizione.

La scelta del prezzo di vendita migliore è stata ed è ancora un processo in continua evoluzione e miglioramento. In una fase iniziale del progetto, il recupero dei prezzi di vendita è stato effettuato sfruttando le API di Amazon Selling Partner (SP-API) [3]. Tramite l'endpoint GET /products/pricing/v0/price, è stato possibile interrogare i prezzi attuali degli articoli a catalogo utilizzando l'ASIN come chiave di ricerca. Tuttavia, questa metodologia ha mostrato limiti significativi: il prezzo recuperato risulta essere quello proposto dal venditore attualmente più competitivo e non sempre rappresenta un valore affidabile per articoli nuovi. Inoltre, fluttuazioni rapide dovute a offerte temporanee o errori di prezzamento da parte di altri venditori rischiavano di generare valori non realistici che andavano a influenzare negativamente la vendita.

Per ovviare a tali problematiche, è stato introdotto l'utilizzo delle API di Keepa [10]. Keepa offre un monitoraggio storico dei prezzi su Amazon, permettendo di ottenere informazioni più stabili come il prezzo medio di vendita su archi temporali personalizzati (30, 90, 180 giorni). Questo ha consentito di disporre di una base di dati più solida per il calcolo di un prezzo di vendita ragionevole. Nonostante il miglioramento rispetto alla soluzione precedente, anche i dati di Keepa presentavano una variabilità intrinseca legata al tipo di articolo (novità, rarità, scarsità di storico; per esempio, per alcuni articoli, come i puzzle, venivano recuperati prezzi completamente fuori mercato che andavano a inficiare l'account venditore in generale).

Al fine di stabilire un metodo di pricing più controllabile e riproducibile, si è deciso di basare il prezzo finale su una logica interna, a partire dai prezzi di acquisto forniti dai diversi fornitori: non tutti i fornitori forniscono gli stessi dati relativi agli articoli, alcuni forniscono il solo prezzo di acquisto e in questo caso il prezzo di vendita viene calcolato moltiplicando tale valore per un parametro diverso in base

alla fascia di prezzo di acquisto; altri fornitori invece mostrano anche un prezzo di vendita consigliato e questo viene preso come buono. Il prezzo di vendita finale è calcolato effettuando una media tra i vari prezzi di vendita dedotti da ogni fornitore e controllando che questo prezzo sia comunque maggiore del prezzo di acquisto, in modo da evitare di avere un margine finale negativo.

Infine, il processo di determinazione del prezzo di vendita è stabilito basandosi sul marketplace target, in modo da garantire coerenza e competitività nelle varie casistiche:

- **Prezzo di base**: il prezzo iniziale di vendita viene calcolato partendo dal prezzo di acquisto fornito dai diversi fornitori, su cui viene applicato un ricarico predefinito. Tale ricarico varia in base ai fornitori e agli sconti applicati sui vari articoli da essi. Questo approccio consente di garantire la sostenibilità economica delle vendite.

- **Vendite su Shopify**: nel caso specifico dell'e-commerce Shopify, al prezzo di base possono essere applicate ulteriori politiche di sconto, in modo da incentivare il cliente finale ad effettuare l'acquisto dove l'azienda non paga ulteriori commissioni. Questi sconti vengono calcolati dinamicamente in base a diversi parametri, quali la quantità di articoli disponibili in magazzino e la disponibilità presso i fornitori.

- **Vendite tramite Amazon FBA**: per gli articoli destinati al programma Fulfilled By Amazon (FBA), di cui parleremo più avanti, il prezzo finale tiene conto di ulteriori considerazioni: le commissioni di vendita applicate stimate da Amazon, recuperate tramite API, i costi di logistica e stoccaggio. Questo consente di preservare la marginalità anche in presenza dei maggiori costi operativi tipici del servizio FBA.

In aggiunta a questi meccanismi automatici, ho previsto anche la possibilità, per ogni singolo articolo, di forzare manualmente il prezzo di vendita. Tale funzionalità è utile in casi specifici, ad esempio durante campagne promozionali, per allineamenti con prezzi di mercato particolarmente competitivi o in presenza di esigenze strategiche definite dal team commerciale dell'azienda. L'override manuale del prezzo viene registrato a livello di database, con tracciabilità della modifica e priorità rispetto ai calcoli automatici standard.

Lo studio e l'adozione di questa architettura ibrida, che combina automazione e flessibilità manuale, hanno permesso di ottimizzare il processo di pricing, riducendo gli errori, adattandosi rapidamente alle variazioni di mercato e garantendo un costante allineamento tra obiettivi di margine e piazzamento competitivo.

4.1.2 Analisi e confronto prezzi recuperati da fornitori e da altre piattaforme

Nel processo di definizione di un prezzo di vendita competitivo, ha giocato un ruolo fondamentale l'analisi comparativa dei prezzi recuperati da diverse fonti. In particolare, sono stati confrontati i listini forniti dai distributori ufficiali con le informazioni pubblicamente disponibili su piattaforme di e-commerce come Amazon, integrando anche dati storici e statistici provenienti da servizi terzi, come Keepa.

I prezzi di acquisto forniti dai distributori rappresentano il primo riferimento per la determinazione del margine lordo. Tuttavia, tali prezzi non sempre rispecchiano la reale dinamicità del mercato online, in cui il valore percepito di un prodotto può variare in modo sensibile in funzione della concorrenza, della disponibilità o della stagionalità. Per questo motivo, ho sviluppato un'infrastruttura in grado di recuperare in modo automatizzato e periodico il prezzo attuale degli articoli venduti su Amazon, sfruttando le API ufficiali messe a disposizione.

Tuttavia, ho riscontrato che le informazioni restituite dalle API di Amazon, sebbene aggiornate in tempo reale, non sempre risultavano affidabili per la definizione di una strategia di pricing: in molti casi, infatti, veniva restituito un prezzo non competitivo o con scarsa rilevanza di mercato. Per fronteggiare questo limite, ho integrato il servizio offerto da Keepa, una piattaforma esterna che fornisce una cronologia dettagliata dei prezzi di ogni articolo Amazon, compresi valori minimi, medi e massimi su intervalli temporali differenti. L'integrazione delle API di Keepa mi ha permesso di ottenere una visione più completa del comportamento storico dei prezzi, facilitando l'individuazione di anomalie e la definizione di un valore di riferimento più realistico. Un limite importante che ho riscontrato nell'utilizzo di Keepa e nell'analisi a campione di alcuni articoli è la difficoltà di adeguare un prezzo alla rarità di un articolo: nel caso di articoli con scorte in scadenza o molto ricercati, il prezzo dedotto analizzando i dati di Keepa non garantiva la marginalità attesa.

Il confronto tra i prezzi forniti dai distributori e quelli rilevati sui marketplace ha evidenziato, in diversi casi, incongruenze significative, sia verso l'alto che verso il basso. In presenza di prodotti in forte concorrenza, ad esempio, è risultato che il prezzo medio di mercato risultava inferiore al prezzo di acquisto, rendendo necessario l'esclusione temporanea del prodotto dai canali di vendita online, per evitare margini finali negativi.

Questa analisi ha rappresentato la base su cui ho costruito algoritmi di pricing adeguati, in grado di valutare per ogni articolo il prezzo di vendita ideale.

4.1.3 Implementazione caricamento inventario sui vari marketplace

Nel contesto della gestione centralizzata del catalogo dei prodotti, l'interfacciamento con i diversi marketplace ha richiesto un'attenta progettazione della procedura di sincronizzazione dell'inventario. Ogni piattaforma presenta infatti specificità tecniche e operative, sia in termini di API esposte, sia in termini di requisiti sui dati richiesti. Per affrontare in modo modulare questa complessità, l'infrastruttura che ho implementato distingue due macro-procedure operative per ogni marketplace integrato:

- **Procedura di creazione nuovi articoli**
- **Procedura di aggiornamento periodico di prezzo e giacenza**

La creazione dei nuovi articoli consiste nella pubblicazione iniziale del prodotto sul marketplace, operazione che richiede un set esteso di informazioni, variabili a seconda del marketplace: identificativo unico (EAN nel caso in oggetto), titolo, descrizione, categorie, immagini, peso e in alcuni casi della documentazione aggiuntiva. Questo processo può variare significativamente tra le varie piattaforme: ad esempio, su Amazon è spesso sufficiente fornire l'ASIN di riferimento per richiamare automaticamente molte delle informazioni già presenti nel catalogo globale, mentre su Shopify è necessario fornire esplicitamente ogni singolo campo. Anche per eBay, l'inserimento richiede un tracciato XML ben definito che includa dettagli sul prodotto e metodi di spedizione. Siccome il catalogo dei vari fornitori è recuperato giornalmente e può includere ogni giorno nuovi prodotti, la creazione di nuovi articoli avviene anch'essa giornalmente, per offrire le novità nel momento in cui escono e rimanere competitivi.

L'aggiornamento periodico di prezzo e disponibilità, invece, è una procedura a cadenza regolare, automatizzata attraverso job schedulati, che si limita ad aggiornare 2 parametri essenziali per il corretto funzionamento delle vendite online: il prezzo di vendita e la quantità disponibile (stock). Questa operazione si fonda sulle informazioni ottenute dal sistema gestionale MAGO e viene propagata a ciascun marketplace utilizzando le API corrispondenti. Nella stessa procedura viene gestita anche l'attivazione/disattivazione di determinati articoli, in quanto, per diverse ragioni, come un errore di mappatura dell'articolo da parte del fornitore, può essere necessario eliminare un articolo da determinati marketplace per un certo periodo per poi reinserirli successivamente a problema risolto.

Questa suddivisione è fondamentale per garantire affidabilità e controllo nei processi: la procedura di creazione articoli, infatti, è soggetta a validazioni molto più rigorose e può causare errori bloccanti in caso di dati non conformi, come per esempio la mancanza di immagini o altri attributi, mentre l'aggiornamento

giacenze e prezzi è solitamente un processo più leggero, essenziale per mantenere allineato l'inventario effettivo con quello pubblicato, evitando overselling o disallineamenti di prezzo. La progettazione e l'implementazione di queste due categorie di procedure hanno richiesto una logica modulare e scalabile, in grado di gestire diverse condizioni a seconda della piattaforma, come limiti di chiamate (rate limits), formati richiesti, requisiti di autenticazione e modalità di risposta asincrona da parte dei sistemi terzi. In particolare, sono stati previsti sistemi di logging avanzato e gestione degli errori per ciascun job in modo da tenere traccia di ogni chiamata. Questa architettura ha permesso di ridurre significativamente l'intervento manuale dell'operatore, garantendo una sincronizzazione continua ed efficiente del catalogo, anche in presenza di un elevato numero di articoli e di aggiornamenti frequenti.

Il punto di partenza architetturale delle procedure di caricamento e aggiornamento del catalogo è rappresentato dalla tabella *ItemMarketplace*, un'entità di appoggio creata ad hoc nel database, con lo scopo di mappare esplicitamente il legame tra un articolo presente nell'anagrafica interna (identificato univocamente dal suo codice EAN) e le diverse piattaforme esterne di vendita.

La struttura della tabella *ItemMarketplace* include i seguenti campi fondamentali:

- **EAN**: codice identificativo univoco dell'articolo a livello globale.
- **IDMarketplace**: identificativo interno della piattaforma (es.Amazon, eBay, Shopify)
- **CodiceArticoloMarketplace**: codice con cui l'articolo viene identificato dalla piattaforma (es. SKU, Asin, ItemID)
- **PrezzoManuale**: prezzo forzato da utilizzare sul marketplace
- **CategoriaSpedizione**: categoria di spedizione da applicare all'articolo nel marketplace
- **Attivo**: campo intero che rappresenta lo stato di attivazione dell'articolo nel determinato marketplace. Il valore assunto da questo campo permette di distinguere le diverse condizioni operative per la fase di aggiornamento:
  - **0**: articolo non attivo (rimosso o non visibile temporaneamente nel marketplace)
  - **1**: articolo attivo (presente e visibile nel marketplace)
  - **2**: articolo da attivare (articolo non attivo che deve essere riattivato e reso nuovamente visibile nel marketplace)
  - **3**: articolo da disattivare (articolo attivo che deve essere rimosso dal marketplace)

L'adozione di un campo numerico multi-valore mi ha permesso di gestire in modo transazionale e asincrono la pubblicazione, disaccoppiando il momento della decisione (per esempio, segnare un articolo come 'da attivare') dal momento dell'effettiva operazione di caricamento o rimozione. Questo approccio si è rivelato molto utile in presenza di procedure schedulate che operano in orari definiti per ottimizzare l'uso delle risorse e rispettare i limiti di chiamate API.

Procedura di creazione nuovi articoli

Ho progettato la procedura di creazione di nuovi articoli su ciascun marketplace per garantire tracciabilità, controllo e coerenza dei dati pubblicati, specialmente nel contesto multi-piattaforma in esame, in cui gli articoli possono essere sincronizzati su più marketplace.

Durante l'esecuzione della procedura di pubblicazione, il sistema consulta la tabella ItemMarketplace per verificare la presenza dell'articolo sul marketplace target. Se l'articolo non esiste o se il valore del campo *Attivo* indica che l'articolo è da attivare, il record viene inserito nella lista degli articoli da aggiungere al marketplace. La procedura di creazione dei nuovi articoli comprende le seguenti fasi:

- **Preparazione dei dati** nel formato richiesto dalla piattaforma (per esempio JSON per le chiamate REST, XML per eBay, query GraphQL per Shopify)
- **Validazione dei campi obbligatori** (EAN, Titolo, Prezzo); in caso di assenza di uno di questi campi l'articolo non viene caricato
- **invio dei dati via API**, gestendo attese imposte dai rate limit
- **Analisi della risposta API** per il controllo di eventuali errori da riportare
- **Inserimento o aggiornamento dell'articolo nella tabella *ItemMarketplace***, con salvataggio del codice assegnato dal marketplace e aggiornamento dello stato a Attivo (1).

Ho definito la classe *InfoArticolo* per racchiudere le informazioni fondamentali necessarie al caricamento di un articolo; in base al marketplace, la classe è estesa per inglobare le ulteriori informazioni che sono richieste per un completo caricamento dell'offerta.

```
1     using System;
2
3     public class InfoArticolo
4     {
5         public string EAN { get; set; }
6         public string Titolo { get; set; }
```

```
7           public decimal Prezzo { get; set; }
8           public int Qta { get; set; }
9           public int CategoriaSpedizione { get; set; }
10      }
```

I campi sono i seguenti:

- **EAN**: identificativo univoco dell'articolo dal quale si parte per costruire lo SKU (Stock Keeping Unit)
- **Titolo**: nome commerciale del prodotto
- **Prezzo**: prezzo di vendita, che può essere quello calcolato automaticamente partendo dai prezzi di acquisto dei vari fornitori oppure il prezzo manuale impostato dall'utente; al prezzo di vendita base si somma un valore *DeltaPrezzo* tabellato per impostare un incremento/decremento sul prezzo dinamicamente
- **Qta**: quantità disponibile alla vendita; comprende la somma della quantità disponibile in casa e dai vari fornitori
- **CategoriaSpedizione**: categoria di spedizione tabellata per il calcolo delle spese e delle tempistiche di spedizione

Esempio di creazione catalogo Amazon

Nel caso del marketplace Amazon, l'integrazione è stata realizzata utilizzando le Selling Partner API (SP-API), le quali permettono di operare in maniera specifica sui singoli articoli del catalogo. In particolare, per creare un nuovo articolo viene utilizzato il seguente endpoint:

```
1 PUT /listings/2021-08-01/items/{sellerId}/{sku}
```

Questa chiamata consente di creare o aggiornare le informazioni di listing per un determinato SKU. Tuttavia, prima di effettuare la chiamata effettiva, è fondamentale avere 2 informazioni obbligatorie per la validazione del payload:

- **ASIN**: codice identificativo univoco dell'articolo nel catalogo Amazon
- **ProductType**: tipo di prodotto secondo la tassonomia ufficiale Amazon, necessario per sfruttare correttamente il JSON secondo lo schema previsto

Se queste informazioni non sono già presenti nel sistema interno, perché salvate in precedenza, vengono recuperate dinamicamente attraverso due chiamate distinte:

- **searchCatalogItems**: permette di recuperare i dettagli completi di un prodotto Amazon dato l'EAN, tra cui il *ProductType* associato:

```
1          GET /catalog/2022-04-01/items/
2
```

- **getProductTypeDefinitions**: restituisce lo schema JSON necessario per strutturare in modo corretto e completo il payload della chiamata *putListingItem*, oltre che il valore di categoria del prodotto da utilizzare

Una volta ottenute tutte queste informazioni, costruisco dinamicamente il corpo della richiesta *putListingItem* secondo lo standard JSON previsto da Amazon, includendo i campi essenziali presenti nella classe *InfoArticolo* prima definita. L'operazione viene eseguita articolo per articolo, in modo da rispettare i rate limit e gestire singolarmente eventuali errori restituiti in fase di validazione.

In caso di esito positivo, registro l'ASIN nella tabella *ItemMarketplace* insieme allo stato Attivo impostato a 1, consentendo così il tracciamento e la futura gestione dell'articolo.

Esempio di creazione catalogo IBS

Per quanto riguarda il marketplace IBS, invece, la procedura di caricamento dei nuovi articoli è differente e prevede l'utilizzo della piattaforma Mirakl Seller API, che non opera tramite invii singoli per articolo ma tramite l'upload massivo di un file contenente più righe prodotto in un formato predefinito.

La struttura del file varia a seconda della configurazione del marketplace (anche il marketplace EPrice sfrutta questa piattaforma ma ha delle differenze), ma include generalmente i seguenti campi:

- Codice EAN
- Nome del prodotto
- Descrizione
- Prezzo
- Quantità disponibile
- Categoria
- Immagini (sotto forma di URL pubblico)
- Altri campi

Una volta costruito il file localmente sulla base dei dati presenti internamente sul gestionale (tramite la classe *InfoArticolo*), la procedura automatizzata effettua l'upload attraverso la chiamata:

```
1    POST /api/offers/imports
```

Questa chiamata API permette di inviare file di tipo multipart/form-data. Dopo l'invio, viene restituito un identificativo del job di importazione del file, che utilizzo per monitorare lo stato dell'operazione attraverso la chiamata di una seconda API:

```
1    GET /api/offers/imports/{importId}
```

Alla conclusione del processo di import, Mirakl mette a disposizione un report degli errori di caricamento, scaricabile attraverso l'endpoint:

```
1    GET /api/offers/imports/{importId}/error_report
```

Questo report è fondamentale per l'analisi, in quanto elenca per ogni riga del file i potenziali errori di validazione, come campi mancanti, formati non conformi o categorie non riconosciute. Il sistema sviluppato automatizza il download e l'archiviazione locale del report, registrandolo nei log della procedura, così da consentire una successiva revisione ed eventuali azioni correttive.

Questa modalità di caricamento asincrona e basata su file formattati è ben adatta a contesti in cui si gestiscono grandi volumi di articoli e consente di ridurre la frequenza delle chiamate API.

Procedura di aggiornamento di giacenze e prezzi

L'aggiornamento periodico delle disponibilità e dei prezzi degli articoli già presenti sui vari marketplace costituisce una delle fasi più delicate ed essenziali del processo di integrazione, in quanto valori errati di prezzo o disponibilità possono provocare grandi problemi per l'attività. Data la natura dinamica delle giacenze e dei listini, soggetti a variazioni dovute a vendite, riordini, promozioni o modifiche dei prezzi da parte dei vari fornitori, è fondamentale garantire un allineamento costante tra il gestionale e le piattaforme di vendita online.

Prima di procedere con l'aggiornamento dei dati, ho scelto di eseguire una verifica dello stato degli articoli nella tabella *ItemMarketplace*, dove il campo Attivo identifica se un prodotto deve essere attivato o disattivato. Gli articoli segnati come 'da disattivare' vengono processati all'inizio.

Nel caso del marketplace di Amazon, per esempio, la disattivazione avviene attraverso una chiamata all'API:

```
DELETE /listings/2021-08-01/items/{sellerId}/{sku}
```

La procedura attende la risposta della chiamata e, in caso di esito positivo, viene aggiornato il campo Attivo dell'articolo corrispondente nel marketplace nella tabella *ItemMarketplace*, impostando lo stato su 'disattivato'. Questo garantisce la coerenza tra il database locale e lo stato del prodotto sul marketplace.

Per quanto riguarda l'aggiornamento di prezzo e disponibilità degli articoli ancora attivi, l'approccio che ho adottato è volutamente ottimizzato, a causa del grande volume di dati da gestire (oltre 800.000 offerte, destinate ad aumentare nel tempo). Risulterebbe inefficiente e rischioso tentare di aggiornare indiscriminatamente tutti i prodotti a ogni esecuzione della procedura, in quanto non tutti gli articoli hanno subito effettive modifiche di prezzo o giacenza.

Prendendo come esempio il caso del marketplace di Amazon, si effettua prima il download dell'inventario attuale pubblicato tramite le Report API di Amazon. Il report ottenuto contiene, per ogni SKU, il prezzo attuale e la quantità disponibile. Si recuperano poi per ogni articolo caricato sul marketplace i rispettivi valori attuali di prezzo e giacenza, che vengono poi confrontati con ogni riga presente nel report. Solo in presenza di una differenza su uno dei due valori (prezzo, quantità o entrambi), l'articolo viene inserito nella lista di aggiornamento. L'aggiornamento avviene mediante l'invio di un listing feed in formato JSON, utilizzando l'API:

```
POST /feeds/2021-06-30/documents
```

Il payload della chiamata è conforme allo schema utilizzato per la creazione dell'articolo, precedentemente recuperato, e permette di aggiornare in un'unica operazione fino a 10.000 offerte. Questo approccio consente di:

- Minimizzare le chiamate API, rispettando i limiti di rate imposti
- Evitare errori dovuti a tentativi di aggiornamento di offerte non cambiate

Dopo l'invio del feed, è fondamentale verificare che le modifiche siano state effettivamente applicate. Amazon fornisce un report di elaborazione del feed, recuperabile tramite l'API:

```
GET /feeds/2021-06-30/feeds/{feedId}/document
```

Questo report dettagliato indica, per ogni SKU, se l'aggiornamento è stato applicato con successo o se si sono verificati degli errori, come ad esempio:

- SKU non riconosciuto

- Dati mancanti o non validi
- Errori di Amazon

Questa logica di aggiornamento è applicata anche agli altri marketplace compatibili con aggiornamenti in bulk. In ogni caso, tutti gli esiti delle operazioni di aggiornamento o di disattivazione vengono registrati nei log specifici della procedura, garantendo una completa tracciabilità e facilitando interventi di correzione.

Capitolo 5

Interfaccia con fornitori

5.1 Analisi di diverse modalità di recupero cataloghi prodotti

All'interno del progetto aziendale affrontato, la gestione dei cataloghi dei fornitori ha rappresentato una delle sfide più rilevanti e complesse, a causa dell'eterogeneità delle modalità di fornitura dei dati e della struttura stessa dei cataloghi ricevuti. L'obiettivo era garantire che tutte le informazioni rilevanti potessero essere acquisite, integrate, e centralizzate nel sistema gestionale di MAGO, pur partendo da fonti e formati profondamente diversi tra loro. In primo luogo, è stato necessario progettare e sviluppare un'infrastruttura in grado di gestire modalità di acquisizione differenti a seconda del fornitore. Le principali modalità operative identificate sono:

- **Accesso automatico via FTP/SFTP**: in molti casi, i fornitori mettono a disposizione i loro listini aggiornati tramite server FTP o SFTP. Il sistema che ho implementato prevede una connessione schedulata a tali server, il download dei file (in formato CSV, Excel o txt) e la successiva fase di parsing e validazione. Una volta importati i dati, questi vengono salvati in un database intermedio creato ad hoc per ospitare i cataloghi di ogni differente fornitore.

- **Caricamento manuale da parte dell'operatore**: alcuni fornitori non mettono a disposizione interfacce automatizzate. In questi casi, i file devono essere scaricati manualmente da portali o ricevuti via email. Ho pertanto sviluppato un'interfaccia utente all'interno del portale gestionale, dedicata all'upload e alla validazione manuale dei file ricevuti. L'interfaccia permette all'operatore di verificare subito eventuali errori formali o incogruenze nei dati.

- **Gestione particolare per fornitori specifici (es. EM)**: alcuni fornitori, come nel caso del fornitore EM, necessitano una gestione particolare, in quanto

hanno un catalogo completo caricato tramite upload, e anche un catalogo delle sole disponibilità. Ho quindi predisposto una pagina ad hoc che effettua query puntuali in tempo reale per recuperare e modificare le informazioni degli articoli, limitando così il carico dati e ottimizzando le risorse.

Uno degli aspetti più critici nella gestione di questi cataloghi è la loro struttura interna, che può variare significativamente da un fornitore all'altro. Alcuni cataloghi possono contenere informazioni dettagliate come dimensioni, peso, descrizione estesa, immagini, brand, data di disponibilità futura o riferimenti a codici alternativi. Altri possono limitarsi a fornire solo i campi essenziali, ovvero prezzo di acquisto e disponibilità. Per affrontare questa eterogeneità di informazioni, ogni fornitore è stato gestito attraverso l'uso di tabelle specifiche in un database separato dal gestionale MAGO, progettate ad hoc per ospitare tutti i campi disponibili in quel particolare catalogo. Questo approccio consente di mantenere la completezza e l'integrità delle informazioni originali, senza perderne alcuna a causa di una normalizzazione prematura o forzata. Tuttavia, al fine di poter integrare le informazioni nel sistema gestionale centrale, ho identificato un sottoinsieme minimo e imprescindibile di dati fondamentali che devono essere necessariamente presenti in ogni catalogo per consentirne l'utilizzo:

- **EAN**: codice identificativo univoco utilizzato come chiave primaria per la mappatura degli articoli. L'EAN è essenziale non solo per l'integrazione nel gestionale, ma anche per il corretto incrocio con cataloghi di altri fornitori e per il caricamento sui marketplace.
- **Prezzo di acquisto**: indispensabile per la successiva determinazione del prezzo di vendita e per il calcolo dei margini.
- **Disponibilità**: indica il numero di unità attualmente presenti nel magazzino del fornitore o la possibilità di ordinare l'articolo. Questa informazioni è importante per l'aggiornamento delle giacenze e per l'automazione degli ordini di approvvigionamento. Alcuni fornitori, invece di indicare la quantità disponibile all'acquisto, indicano soltanto se l'articolo è disponibile o meno: in questi casi viene fissata una disponibilità fittizia di 3 a indicare che l'approvvigionamento è possibile.

Durante la fase di centralizzazione dei dati, il sistema effettua quindi un processo di estrazione, trasformazione e normalizzazione (ETL), in cui i dati contenuti nelle tabelle ad hoc vengono uniformati per popolare il database di MAGO. Viene mantenuto un mapping preciso tra i campi dei singoli cataloghi e i campi del gestionale, con particolare attenzione alla coerenza e all'aggiornamento dei dati sensibili. Questo approccio modulare e scalabile consente di integrare nuovi fornitori nel

sistema con tempi contenuti, semplicemente progettando nuove tabelle intermedie e definendo le regole di trasformazione per i campi principali, mentre l'interfaccia utente e le procedure automatizzate restano in gran parte utilizzabili.

5.2 Implementazione gestione ordini fornitori

5.2.1 Recupero cataloghi

Il primo passo per una gestione efficiente degli ordini ai fornitori è la disponibilità costante e aggiornata dei cataloghi prodotti. Per questo motivo, ho sviluppato un sistema automatizzato per il recupero periodico dei cataloghi da ciascun fornitore, tenendo conto delle modalità eterogenee con cui queste informazioni vengono rese disponibili.

I fornitori gestiti si dividono principalmente in due categorie: quelli che mettono a disposizione un file scaricabile tramite protocollo FTP/SFTP e quelli che richiedono l'importazione manuale di un file CSV o Excel. Per ciascun fornitore è stato sviluppato un modulo dedicato, in grado di interpretare correttamente il formato del catalogo, analizzare i dati ed effettuare un primo salvataggio in una base dati intermedia creata ad hoc.

Tale base dati è costituita da un database separato da quello del gestionale MAGO, in cui ogni fornitore ha una propria tabella ad hoc progettata per contenere tutti i campi presenti nel rispettivo file. Questa strategia permette di preservare l'integrità e la specificità delle informazioni originarie, anche nel caso non siano direttamente utilizzabili nel processo di integrazione con il database centrale. Ad esempio, alcuni fornitori mettono a disposizione campi accessori come categorie intermedie, codici di identificazione interni, dimensioni di imballo, descrizioni multilingua o compatibilità con altri articoli, mentre altri forniscono solo i campi essenziali.

I tre attributi ritenuti fondamentali e sempre presenti – ovvero EAN, prezzo di acquisto e disponibilità – vengono sempre estratti e successivamente centralizzati nel database principale del gestionale. L'EAN funge da identificatore univoco del prodotto, il prezzo di acquisto consente il calcolo del margine e della strategia di prezzamento delle offerte, mentre la disponibilità è essenziale per la pianificazione degli ordini e la visibilità online.

Per i fornitori che non espongono cataloghi accessibili in autonomia e per i quali ho valutato che il tentativo di recupero automatizzato non fosse possibile, ho previsto un'interfaccia utente web dedicata, che consente il caricamento manuale del file fornito. Il sistema effettua una validazione preliminare del formato e dei dati fondamentali, nonché fornisce un file di import che mostra i campi da inserire nel file, garantendo una corretta importazione nel database secondario.

Una menzione particolare va fatta per il fornitore EM, il quale richiede una gestione più strutturata: in questo caso, ho reso possibile l'upload di 2 file diversi, uno per l'aggiornamento dell'intero catalogo disponibile e uno per l'aggiornamento delle sole giacenze; inoltre, per gestire in modo più preciso e tempestivo l'aggiornamento di singoli articoli soggetti a modifiche frequenti di prezzo o disponibilità, ho scelto di implementare una tabella per l'aggiornamento di questi campi.

In sintesi, il recupero dei cataloghi è stato concepito per garantire modularità, scalabilità e affidabilità, con l'obiettivo di alimentare il gestionale con dati coerenti, aggiornati e validati, riducendo al minimo l'intervento umano e il rischio di errore.

5.2.2 Creazione ordini ai fornitori

Seguendo l'obiettivo primario di automatizzazione dei flussi di approvvigionamento, ho condotto un'analisi volta a progettare un sistema che permettesse la generazione e la trasmissione automatica degli ordini ai fornitori, partendo dalla domanda proveniente dai marketplace. L'obiettivo della ricerca è stato garantire la tracciabilità degli ordini, ridurre l'intervento manuale e adattare il sistema alle differenti modalità operative previste da ciascun fornitore.

Dall'analisi dei flussi di evasione degli ordini cliente è emersa la necessità di disporre di una tabella specifica per ogni fornitore, contenente gli ordini da generare. Ciascuna di queste tabelle è strutturata con i seguenti attributi fondamentali:

- **Marketplace**
- **NumeroOrdine**
- **EAN**
- **Qta**
- **DataCreazioneOrdineFornitore**
- **Ordinato**

Dove *Marketplace* e *NumeroOrdine* sono i riferimenti dell'ordine cliente che ha richiesto l'articolo, *EAN* è l'identificativo univoco dell'articolo da ordinare, *Qta* è la quantità da ordinare, *DataCreazioneOrdineFornitore* è il timestamp in cui viene effettuato l'ordine e *Ordinato* è un booleano che indica se l'ordine è stato effettuato o meno. Una struttura di questo tipo ha permesso una mappatura diretta tra la domanda (proveniente dal marketplace) e l'azione (ordine verso il fornitore), agevolando la fase di monitoraggio.

La fase di sperimentazione ha evidenziato che l'inserimento dei dati nelle tabelle ordini fornitore può essere automatizzato al momento dell'evasione degli ordini

cliente, ovvero quando il sistema rileva la mancanza di disponibilità di un articolo in magazzino e individua il fornitore idoneo per la fornitura, basandosi sulla disponibilità e sul prezzo di acquisto (in caso di articolo presente su più fornitori, la logica vuole che venga scelto quello dove l'articolo costa di meno).

Durante la fase di studio, ho classificato le modalità di trasmissione degli ordini in tre principali categorie:

- **Upload automatico tramite FTP/SFTP**: adottato dai fornitori che mettono a disposizione un endpoint per il caricamento diretto dei file d'ordine in formato CSV o Excel. In questo caso, il sistema recupera gli ordini da effettuare, genera il file nel formato specifico richiesto dal fornitore e lo carica automaticamente nella directory di destinazione, aggiornando lo stato dell'articolo da ordinare nel database in modo che non venga processato ulteriori volte.

- **Invio via e-mail**: utilizzato dai fornitori che non dispongono di un sistema automatico di ricezione ordini, ma richiedono un file allegato da inviare a un indirizzo e-mail dedicato. Anche in questo caso, il processo di generazione ed invio del file è stato automatizzato.

- **Procedure custom**: alcuni fornitori, come nel del fornitore *DL*, richiedono una procedura di caricamento più articolata, composta da 2 fasi distinte: nella prima viene creato e caricato un file contenente gli articoli da ordinare, mentre nella seconda fase è necessario trasmettere un file aggiuntivo (con estensione ".done") che ne confermi la chiusura. Per progettare questa interfaccia ho dovuto implementare uno script con controllo sequenziale e temporizzato dei due file, per evitare errori di sincronizzazione.

Uno dei risultati chiave emersi da questa analisi è stata la necessità di sincronizzare la creazione e l'invio degli ordini fornitore con le tempistiche operative delle varie aziende partner. Dopo diversi test si è stabilito che l'invio degli ordini risulta più efficace se programmato nelle prime ore della giornata lavorativa (tra le 6:30 e le 7:00). La scelta di questo orario consente ai vari fornitori di ricevere e processare gli ordini durante la mattinata, permettendo la spedizione nel pomeriggio, in modo da ridurre al minimo i tempi di approvvigionamento. Inoltre, la scelta di eseguire la procedura di creazione degli ordini fornitore la mattina presto permette, in caso di errori, di agire tempestivamente per la loro risoluzione.

La soluzione trovata si è dimostrata efficace nel ridurre in modo significativo i ritardi di evasione e nell'incrementare l'affidabilità dell'intero processo di approvvigionamento. I risultati ottenuti rappresentano una base solida per eventuali estensioni future, come l'integrazione con sistemi EDI [11] o l'introduzione di meccanismi predittivi per l'approvvigionamento.

5.2.3 Recupero e caricamento delle fatture

Il processo di acquisizione delle fatture dai fornitori, fondamentale per la tracciabilità degli ordini e per la corretta gestione amministrativa, è stato oggetto di una progettazione orientata sia alla flessibilità operativa, sia all'automazione dei processi. Analizzando le varie tipologie di invio delle fatture da parte dei vari fornitori, ho identificato e implementato due modalità distinte di gestione del caricamento delle fatture:

- **Caricamente manuale da interfaccia web**: per i fornitori che non offrono modalità automatizzate di trasmissione delle fatture, ho progettato e realizzato una pagina dedicata all'interno del portale gestionale. Questa interfaccia consente agli utenti di caricare manualmente i file delle fatture ricevuti (in formato pdf o excel), scegliendo anche il metadati fondamentali (quali fornitore e numero di documento). Ogni file caricato in questo modo viene salvato in una cartella sul server e i dati essenziali riguardo articolo, quantità e prezzo vengono registrati in apposite tabelle per la successiva elaborazione.

- **Recupero automatico tramite accesso ai portali fornitori**: per i fornitori dotati di un portale web, che pubblicano regolarmente le fatture in aree riservate accessibili tramite credenziali private, ho progettato e sviluppato un sistema automatizzato di recupero delle fatture. In particolare, ho scelto di utilizzare *Selenium* [12], uno strumento open-source ampiamente utilizzato per l'automazione dei browser web. Selenium consente di simulare il comportamento di un utente umano nella navigazione di un sito web, permettendo di automatizzare la fase di login, la navigazione verso la pagina contenente i documenti e il download del file in modo controllato. L'utilizzo di Selenium si è rilevato estremamente efficace nei casi in cui i portali non offrono API di accesso ai dati. L'approccio basato su WebDriver permette infatti di eseguire il recupero dei file anche da portali protetti da autenticazione e con interfacce complesse. Il processo è stato schedulato per essere eseguito in modo periodico e include meccanismi di logging automatico, controllo degli errori e notifica automatica via e-mail in caso di problemi di accesso, errori di connessione o mancato download.

Implementazione recupero file da fornitore con Selenium

```
1       ChromeOptions options = new ChromeOptions();
2
3       options.AddArgument("--headless");
4       options.AddArgument("--start-maximized");
5
```

```
6       options.AddUserProfilePreference("profile.
        default_content_settings_popups", 0);
7       options.AddUserProfilePreference("download.default_directory",
        ConfigurationManager.AppSettings["directoryDownloadFatture"]);
8       options.AddUserProfilePreference("download.prompt_for_download
        ", false);
9       options.AddUserProfilePreference("safebrowsing.enabled", true)
        ;
10      options.AddUserProfilePreference("safebrowsing.
        disable_download_protection", true);
```

Per prima cosa creo l'oggetto *ChromeOptions*, che contiene le configurazioni per inizializzare correttamente l'istanza di Chrome che verrà avviata:

- –**headless**: avvia Chrome senza l'interfaccia utente visibile, utile per l'esecuzione in background sul server.
- –**start-maximized**: avvia l'istanza di Chrome a schermo intero, in modo da evitare problemi di rendering dei componenti o comportamenti anomali dovuti a elementi visibili solo a certe risoluzione dello schermo.
- **profile.default_content_settings_popups**: impostato a 0, disabilita i popup che potrebbero inficiare il corretto funzionamento della navigazione automatizzata.
- **download.default_directory**: imposta la il percorso predefinito di download dei file; il valore viene preso dal file di configurazione in modo che possa essere modificato senza dover ricompilare il progetto.
- **download.prompt_for_download**: impostato a *false*, evita che venga mostrato un prompt per ogni download per permettere il download di file senza un intervento manuale, che andrebbe contro l'obiettivo di automazione.
- **safebrowsing.enabled**: impostato a *true*, attiva il Safe Browsing di Chrome per proteggere l'utente da siti pericolosi.
- **safebrowsing.disable_download_protection**: impostato a *true*, disattiva la protezione che blocca il download di file potenzialmente pericolosi, necessario perchè Chrome potrebbe bloccare il download di certi file.

Successivamente viene creato il driver per accedere alla pagina web desiderata:

```
1       IWebDriver driver = new ChromeDriver(options);
2
```

```
3      driver.Navigate().goToUrl(ConfigurationManager.AppSettings["
       urlPortaleFornitore"]);
```

Viene poi istanziato l'oggetto *ChromeDriver*, utilizzando le opzioni prima definite; questo crea l'istanza del browser Google Chrome pronta per essere controllata in modo automatico. Ci si collega poi all'url desiderato, anch'esso definito nel file di configurazione dell'applicazione.

```
1      WebDriverWait wait = new WebDriverWait(driver, TimeSpan.
       FromSeconds(5));
2      wait.Until(ExpectedConditions.ElementIsVisible(By.Id("
       txtLoginUsername")));
3
4      Thread.Sleep(10000);
```

L'oggetto *WebDriverWait* imposta un'attesa esplicita di 5 secondi e viene utilizzato per attendere che l'elemento identificato da *id="txtLoginUsername"* sia visibile: questo serve per attendere che venga raggiunta la pagina e che essa sia completamente caricata.

Ho poi definito una pausa forzata di 10 secondi in quanto, in seguito a vari test, l'ambiente del server di produzione si è rivelato più veloce dell'ambiente di test e, per evitare errori di timeout successi, quest'attesa statica permette un'esecuzione senza intoppi.

```
1      IWebElement usernameTextbox = driver.FindElement(By.Id("
       txtUsername");
2      IWebElement passwordTextbox = driver.FindElement(By.Id("
       txtPassword");
3
4      usernameTextbox.SendKeys(ConfigurationManager.AppSettings["
       username"]);
5      passwordTextbox.SendKeys(ConfigurationManager.AppSettings["
       password"]);
6
7      IWebElement bottoneLogin = driver.FindElement(By.LinkText("
       Login"));
8
9      bottoneLogin.Click();
10
11     wait.Until(ExpectedConditions.ElementIsVisible(By.Id("
       divCarrello")));
12     bool accesso = true;
```

La funzione *FindElement* serve per identificare gli elementi nella pagina responsabili del login, ovvero le textbox relative a username e password e il bottone per inviare richiesta: le due textbox vengono identificate dal loro id, mentre il bottone di accesso, non avendo un identificativo, dopo dei tentativi, ho deciso di individuarlo tramite il suo testo, dopo essermi assicurato che fosse l'unico nella pagina in modo da non generare ambiguità che potessero condurre a successivi errori.

Successivamente, la funzione *Click* applicata sull'elemento *bottoneLogin* simula la pressione del bottone da parte dell'utente e invia la richiesta di login; ho poi inserito un'attesa per permettere di ricaricare la pagina in seguito alla risposta dal server del fornitore e, nel momento in cui l'elemento identificato dall'id *divCarrello* risulta visibile, significa che la procedura di login ha avuto successo e la variabile booleana *accesso* è impostata a true.

```
1     driver.Navigate().GoToUrl(ConfigurationManager.AppSettings["
      urlPaginaFatture"]);
2     wait.Until(ExpectedConditions.ElementVisible(By.TagName("tbody
      "));
3
4     int rigaUltimaFattura = 1;
5
6     IWebElement bodyTabellaDocumenti = driver.FindElement(By.
      TagName("tbody"));
7     IWebElement colonnaTipoDocumento = bodyTabellaDocumenti.
      FindElement(By.XPath(".//tr[" + rigaUltimaFattura + "]/td[1]"))
      ;
8
9     while(colonnaTipoDocumento.Text != "Fattura"){
10        rigaUltimaFattura++;
11        colonnaTipoDocumento = bodyTabellaDocumenti.FindElement(By
      .XPath(".//tr[" + rigaUltimaFattura + "]/td[1]"));
12    }
13
14    IWebElement colonnaNumeroDocumento = bodyTabellaDocumenti.
      FindElement(By.XPath(".//tr[" + rigaUltimaFattura + "]/td[2]"))
      ;
15    IWebElement colonnaDataFattura = bodyTabellaDocumenti.
      FindElement(By.XPath(".//tr[" + rigaUltimaFattura + "]/td[3]"))
      ;
16    IWebElement colonnaDownload = bodyTabellaDocumenti.FindElement
      (By.XPath(".//tr[" + rigaUltimaFattura + "]/td[10]"));
17
18    string numeroFattura = colonnaNumeroDocumento.Text;
19    DateTime dataFattura = DateTime.Parse(colonnaDataFattura.Text)
      ;
20
```

```
21      IWebElement linkDownloadFattura = colonnaDownload.FindElement(
        By.TagName("a"));
22      linkDownloadFattura.Click();
```

Dopo aver eseguito il login con successo, si naviga nella pagina dell'area riservata responsabile della visualizzazione dei vari documenti (fatture, note d'accredito, ecc.) e si attende il caricamento dell'elemento individuato dal tag "tbody", che corrisponde alla tabella dei vari documenti. La tabella contiene i documenti ordinati per data di pubblicazione sulla pagina, bisogna quindi andare a ricercare la prima riga che contiene una nuova fattura non ancora scaricata: ogni colonna della tabella viene identificata dal suo XPath [13], in questo modo viene recuperato ogni campo di ogni singola riga fino a che non si trova la riga dell'ultima fattura, ovvero quella che corrisponde al campo *colonnaTipoDocumento* uguale a "Fattura".

Vengono poi individutati, nella riga trovata, i campi relativi al numero della fattura (*numeroFattura*)e la data di emissione (*dataFattura*), che verranno successivamente salvati nel database interno; il campo relativo al link di download viene infine individuato e ne viene simulata la pressione per effettuare il recupero del file, che verrà salvato nella directory decisa inizialmente nei parametri di configurazione del driver.

Una volta recuperate, le fatture vengono archiviate sul server e vengono caricate tramite microservizio nel gestionale interno. Le fatture sono importanti per andare a caricare nel magazzino del gestionale interno la giacenza degli articoli che sono in arrivo, in modo che vengano generati i documenti necessari (ddt, etichette di spedizione) per la corretta evasione degli ordini cliente.

La dualità scelta tra modalità manuale e automatica permette di adattarsi alla varietà di fornitori coinvolti, garantendo nello stesso tempo efficienza, controllo e tracciabilità. In caso di aggiunta futura di fornitori, la strategia manuale può continuare ad essere utilizzata, previa implementazione del parsing del nuovo tipo di documento; se invece un nuovo fornitore dispone di un sito web raggiungibile per scaricare le fatture, una nuova procedura ad-hoc per automatizzare l'accesso al sito verrà implementata. Il caricamento del documento di acquisto sul gestionale interno resterà comunque invariato, garantendo modularità e efficienza.

Capitolo 6

Gestione degli ordini clienti

6.1 Gestione centralizzata dei vari marketplace di vendita

Nel contesto di studio in oggetto di una vendita multicanale su diversi marketplace digitali, la gestione centralizzata degli ordini rappresenta un requisito di fondamentale importanza per garantire efficienza, tracciabilità e coerenza nei flussi operativi. A fronte della varietà e dell'eterogeneità delle informazioni fornite da ciascun marketplace (sia in termini di struttura dei dati, sia in termini di contenuti) ho dovuto progettare una soluzione unificata in grado di normalizzare i dati relativi agli ordini cliente in un unico punto di accesso all'interno del sistema gestionale.

A tal fine, ho ideato e implementato la tabella *OrdiniClienti*, un contenitore centralizzato creato ad hoc nel database aziendale, pensato per contenere tutte le informazioni rilevanti associate a ciascun ordine cliente proveniente dai diversi marketplace (Amazon, Shopify, eBay, ecc.). La struttura della tabella che ho progettato risulta flessibile ma completa e la sua definizione si è evoluta nel tempo per raccogliere sempre più informazioni e nel modo migliore possibile. I suoi campi fondamentali sono:

- **Marketplace**: nome del marketplace di origine dell'ordine
- **NumeroOrdine**: identificativo univoco dell'ordine nel marketplace associato
- **DataOrdine**: data e ora in cui il cliente ha effettuato l'ordine
- **SKU**: sku relativo alla riga d'ordine
- **EAN**: EAN associato allo sku
- **Qta**: quantità dello sku ordinata

- **PrezzoUnitarioArticolo**: prezzo di vendita associato al singolo articolo
- **IVAPrezzoUnitarioArticolo**: IVA associata al prezzo del singolo articolo
- **PrezzoSpedizione**: prezzo di spedizione relativo all'intero ordine
- **IVAPrezzoSpedizione**: IVA associata al prezzo di spedizione dell'ordine
- **Stato**: da spedire, spedito, reso, rimborsato, annullato
- **Cliente**: riferimento alla riga della tabella contenente le informazioni di spedizione
- **Vettore**: vettore scelto per la spedizione
- **Tracking**: tracking per monitorare l'andamento della spedizione
- **DataSpedizione**: data e ora di conferma della spedizione con il vettore scelto
- **NumeroDDT**: numero identificato del documento di trasporto associato alla spedizione
- **MessaggioErrore**: messaggio di errore possibile associato all'ordine ("giacenza insufficiente", "errore creazione ddt", "errore creazione spedizione", ecc.)
- **CommissioniRigaOrdine**: importo totale delle commissioni dovute al marketplace per la vendita dell'articolo

Oltre alle informazioni essenziali, al fine di garantire una corretta e ottimale gestione delle varie casistiche associate a ogni ordine, ho aggiunto dei campi di supporto utili per seguire tutti gli stati possibili di un ordine:

- **OrdinatoFornitore**: flag che indica se l'articolo nella riga d'ordine è stato ordinato da fornitore
- **Fornitore**: codice fornitore da cui è stato ordinato l'articolo
- **DataOrdineFornitore**: data e ora in cui è stato inoltrato l'ordine di approvvigionamento al fornitore
- **Reso**: flag che indica se è da gestire il reso dell'articolo
- **DataReso**: data di effettuazione del reso
- **ImportoRimborso**: importo del rimborso concesso al cliente

- **DataRimborso**: data e ora di effettuazione del rimborso
- **DaRispedire**: flag che indica se la riga d'ordine è da rispedire
- **DaSpedireParziale**: flag che indica se la riga d'ordine è da inserire in una spedizione parziale

Queste informazioni non sono necessariamente valorizzate in ogni riga d'ordine, ma averle è risultato molto utile nella creazione di report e KPI, nonché nell'automazione di varie procedure per la gestione del processo di evasione degli ordini.

La chiave primaria che identifica ogni riga è una chiave multipla formata dai campi *Marketplace, NumeroOrdine, SKU*: la scelta di avere una tabella unica per la testa e il dettaglio dell'ordine è dovuta alla volontà di semplificare e centralizzare il più possibile il recupero e il salvataggio delle informazioni di ogni ordine, evitando di generare situazioni di riferimenti errati o problemi di sincronizzazione.

Questa struttura consente non solo una visualizzazione uniforme degli ordini ma anche una gestione coerente e intuitiva delle azioni operative collegate agli ordini, come l'approvvigionamento dei prodotti, la preparazione delle spedizioni, la fatturazione e la gestione dei resi.

Durante la fase di progettazione ho posto particolare attenzione alla definizione delle azioni utili e automatizzabili nel ciclo di vita dell'ordine, oltre che alle possibili operazioni che gli operatori umani possono effettuare per gestire casistiche difficilmente automatizzabili (modifica dell'indirizzo di consegna dell'ordine per errore di battitura da parte del cliente, volontà di annullamento dell'ordine, richiesta di modifica di un articolo, ecc.). Tra le principali attività che ho identificato ci sono le seguenti:

- **Verifica della disponibilità degli articoli** in magazzino o presso i fornitori al momento della ricezione dell'ordine: nel caso di mancanza di giacenza, viene valorizzato il campo *MessaggioErrore* con un testo appropriato; in caso di ordine da fornitore, il flag e i campi associati vengono valorizzati a dovere per contenere le informazioni relative all'approvvigionamento.
- **Tracciamento e aggiornamento dello stato dell'ordine**, integrando i dati di spedizione dei corrieri e gli aggiornamenti dei marketplace: il campo *Tracking* e *DataSpedizione* indicano che la spedizione è stata creata e consegnata al corriere scelto per l'instradamento verso la destinazione finale; i vari flag di reso, rispedizione e spedizione parziale indicano alle procedure di processamento degli ordini quali azioni effettuare.

Questa strategia di centralizzazione ha permesso di superare la frammentazione tipica della gestione separata dei vari ordini per ogni marketplace, offrendo una

visione integrata e controllata dell'intero ciclo operativo. Inoltre, la creazione di questa tabella unificata ha facilitato lo sviluppo di sistemi di monitoraggio, reporting e analisi dei KPI legati all'evasione degli ordini, migliorando la capacità dell'azienda di individuare colli di bottiglia, anomalie o inefficienze e di intervenire in modo tempestivo con azioni correttive.

Infine, ho scelto questa struttura modulare in quanto è particolarmente adatta a supportare la futura scalabilità del sistema, rendendo possibile l'aggiunta di nuovi marketplace senza dover riprogettare l'intera logica gestionale, ma semplicemente mappando i nuovi canali di vendita e adattando le necessarie logiche di importazione e validazione dei dati.

Al fine di fornire una panoramica più dettagliata sulla gestione degli ordini, ho scelto di soffermarmi sul caso del marketplace di Amazon, in quanto, oltre alla normale vendita di articoli gestita dal venditore (FBM) [14], presenta anche l'opportunità di vendere secondo il programma FBA [15], affidando gran parte della gestione delle vendite direttamente ad Amazon.

6.2 Panoramica delle principali modalità di vendita su Amazon

Amazon rappresenta oggi uno dei principali marketplace globali e offre ai venditori due modalità principali per la gestione della logistica e della spedizione degli ordini: FBM (Fulfilled By Merchant) e FBA (Fulfilled By Amazon). Queste due modalità presentano caratteristiche, vantaggi e svantaggi distinti ed è stato fondamentale comprenderne le differenze al fine di poter integrare entrambe nella strategia commerciale dell'azienda.

Fulfilled By Merchant (FBM)

Con il modello FBM, è il venditore che gestisce in modo autonomo tutto il processo logistico della vendita: dallo stoccaggio degli articoli, alla preparazione, fino alla spedizione e alla gestione dell'eventuale reso richiesto dal cliente. Amazon in questo caso funge da vetrina, mettendo a disposizione la propria piattaforma per la pubblicazione e vendita degli articoli, ma non interviene nella logistica.

Tra i vantaggi presenti in questa modalità di vendita, si possono riscontrare:

- maggiore controllo sulla gestione del magazzino e della logistica
- costi inferiori per articoli di grandi dimensioni o basso margine, dove le commissioni FBA risulterebbero penalizzanti [16]
- possibilità di personalizzare la spedizione, inserendo nel pacco materiale promozionale o omaggi per aumentare la fidelizzazione del cliente

- non è necessario spedire quantità di articoli ad Amazon, eliminando il rischio di avere giacenza inveduta nei loro magazzini che porterebbe un costo sempre maggiore nel tempo

Sono però presenti anche degli svantaggi per questa modalità, tra i quali:

- maggiore lavoro operativo, soprattuto nei periodi di picco delle vendite, come possono essere le feste natalizie o pasquali
- minore visibilità delle offerte rispetto agli articoli FBA: spesso, infatti, gli articoli FBA sono accompagnati dalla dicitura "Prime" e di una posizione più favorevole nei risultati della ricerca
- la gestione dei resi è totalmente a carico del venditore, che si deve occupare di effettuare il rimborso e recuperare l'articolo che il cliente vuole rendere

Fulfilled By Amazon (FBA)

Nel modello FBA, il venditore spedisce, in modo anticipato rispetto alla vendita finale, gli articoli a uno dei centri logistici di Amazon. Una volta arrivati nel centro di destinazione, Amazon si prende in carico la gestione dell'intero processo logistico degli articoli, inclusa la spedizione, l'assistenza clienti e i resi.

Tra i vantaggi che si ottengono utilizzando questo modello, si possono trovare i seguenti:

- aggiunta della dicitura "Prime" nelle offerte degli articoli, con conseguente aumento di visibilità
- riduzione del lavoro logistico interno all'azienda, in quanto questo è interamente a carico di Amazon
- maggiore affidabilità percepita dal cliente finale
- migliore gestione degli ordini internazionali, grazie alla rete logistica globale di Amazon

D'altro canto, anche il modello FBA presenta degli svantaggi, quali per esempio:

- costi più elevati, in quanto Amazon applica tariffe di stoccaggio e gestione degli articoli che vanno a ridurre, in certi casi anche sensibilmente, i margini di profitto del venditore [17]
- rischio di accumulo di articoli nei centri logistici di Amazon, con conseguente aumento dei costi di stoccaggio

- bassa flessibilità nella personalizzazione della spedizione, in quanto non è più a carico del venditore
- necessità di adeguarsi a precise linee guida per la scelta e la preparazione degli articoli da inviare, in quanto non tutti gli articoli possono essere spediti con questo programma e la preparazione comprende diverse fasi di etichettatura e impacchettamento

Nella soluzione intrapresa, motivata anche e soprattutto dalle scelte aziendali, ho studiato e implementato una logica ibrida, in cui alcuni articoli sono gestiti in FBA (soprattutto quelli più richiesti, le novità e gli articoli che portano un margine di profitto maggiore), mentre altri vengono gestiti solo in FBM per ottimizzare i costi e garantire una maggiore flessibilità operativa. In sostanza, la scelta di FBA è volta soprattutto ad aumentare il volume delle vendite, in modo da favorire un ricambio degli articoli in magazzino generando e generare un profitto.

Questa distinzione ha comportato anche la progettazione di procedure separate sia per il caricamento degli articoli, sia per l'aggiornamento di prezzi e disponibilità, con considerazioni aggiuntive per il calcolo dei margini (in FBA, infatti, sono da tenere in considerazione anche i costi di gestione e stoccaggio applicati da Amazon sugli articoli).

6.3 Vendita gestita direttamente

Come detto in precedenza, nella modalità di vendita FBM (Fulfilled By Merchant), è l'azienda stessa a gestire in modo diretto tutti gli aspetti del processo di evasione: dalla ricezione dell'ordine alla creazione del Documento di Trasporto (DDT), fino alla spedizione del pacco attraverso il vettore di trasporto più conveniente. In questo sottocapitolo mi concentrerò sulle scelte fatte per integrare in modo efficace le vendite Amazon con il gestionale aziendale e per ottimizzare e rendere flessibile la decisione su quale sia il vettore di spedizione migliore da utilizzare.

6.3.1 Integrazione tra vendite Amazon e gestionale

L'integrazione degli ordini provenienti da Amazon con il gestionale aziendale è stata progettata in due fasi distinte:

- **Recupero ordini Amazon FBM da file**: Amazon mette a disposizione dei venditori un report, raggiungibile tramite API all'endpoint

```
1     POST https://sellingpartnerapi-na.amazon.com/reports
      /2021-06-30/reports
2
```

Questo report consiste in un file csv contenente tutte le informazioni relative agli ordini clienti ricevuti in un determinato lasso temporale: il file viene elaborato da una procedura automatica che ne estrae i campi più rilevanti (EAN, quantità, informazioni di spedizione, prezzo vendita, commissioni) e li inserisce nella tabella relativa a tutti gli ordini (*OrdiniClienti*).

- **Processamento e creazione DDT**: nella seconda fase, è stata implementata una procedura automatizzata che verifica gli ordini presenti nella tabella *OrdiniClienti*, controlla la disponibilità dei prodotti nel magazzino e, in caso di giacenza positiva per ogni articolo presente nell'ordine, genera il Documento Di Trasporto (DDT) direttamente nel gestionale MAGO.

Il *Documento Di Trasporto* (DDT) è un documento fiscale non obbligatorio ai fini IVA, ma ampiamente utilizzato in ambito logistico e commerciale per accompagnare le merci durante il trasporto e per giustificare lo spostamento di beni tra 2 persone. È previsto dal D.P.R. 472/1996 [18] e sostituisce, quando utilizzato, la fattura immediata, consentendo la fatturazione differita. Il DDT contiene informazioni essenziali come:

- Dati del cedente e del ricevente
- Data e numero progressivo del documento
- Descrizione delle merci trasportate
- Quantità
- Causa del trasporto (vendita, reso)
- Indirizzo di partenza e di arrivo

L'utilizzo del Documento Di Trasporto permette una maggiore flessibilità nella gestione della logistica, facilitando la tracciabilità dei movimenti di magazzino e aumentando l'efficienza della procedura di fatturazione mensile. Inoltre, è un documento indispensabile per garantire la conformità durante i controlli stradali o in fase di ricezione merci a destinazione.

Nella soluzione proposta, ho progettato, tramite la piattaforma *Telerik Reporting* [19], il DDT che viene generato in modo automatico per ogni ordine elaborato con successo e pronto per essere spedito, integrando tutte le informazioni necessarie salvate nel sistema; il file viene poi stampato e accompagnato al pacco pronto per essere consegnato al vettore di trasporto.

6.3.2 Analisi vettori di trasporto

Un altro aspetto importante nella gestione diretta delle spedizioni è la scelta del vettore di trasporto, in quanto incide in maniera significativa sui costi complessivi e sulla soddisfazione del cliente finale. A tale scopo, ho condotto un'analisi basata su dati storici di spedizione e costi effettivamente sostenuti, con l'obiettivo di ottimizzare le decisioni in modo automatizzato.

Scelte logistiche per spedizioni internazionali

Per le spedizioni internazionali si è scelto di utilizzare **SPRING GDS** [20], una società di logistica internazionale nata dall'associazione tra PostNL, Royal Mail e Singapore Post. SPRING funge da intermediario per le spedizioni internazionali, offrendo servizi a basso costo per pacchi di piccole dimensioni, grazie all'integrazione con i servizi postali nazionali dei vari paesi. In particolare, SPRING consente il consolidamento delle spedizioni e la tracciabilità, con tariffe molto competitive rispetto ad altri corrieri espressi internazionali come DHL o UPS.

Scelte logistiche per spedizioni nazionali

Per quanto riguarda le spedizioni in Italia, l'analisi dei costi ha mostrato che, per gli ordini composti da un singolo articolo o di piccole dimensioni, *Poste Italiane* risulta essere l'opzione più economica, garantendo comunque una buona affidabilità e tempi di consegna accettabili. Tuttavia, per ordini comprendenti più articoli o comunque con merce voluminosa, la spedizione tramite Poste Italiane diventa sensibilmente più costosa o, addirittura, non fattibile. In questi casi si è deciso di optare per uno tra i corrieri espressi *BRT* o *GLS*, in base alla regione di destinazione del pacco.

Dall'analisi dei dati storici di consegna e costi per regione, ho implementato una tabella di preferenza corriere (*VettorePerRegione*), che associa a ogni regione il vettore scelto per la spedizione. Questa tabella, che può essere modificata rapidamente in caso di ottimizzazioni o temporanei blocchi di consegna per un corriere in una determinata regione, viene utilizzata al momento della generazione dell'etichetta di spedizione, rendendo il processo di decisione del tipo di spedizione completamente automatizzato ma aggiornabile.

6.4 Vendita gestita da Amazon

6.4.1 Perché è utile e cosa la differenzia dalla vendita gestita direttamente

Il programma *Fulfilled By Amazon* (FBA) rappresenta, come detto, una modalità alternativa alla vendita diretta (FBM), nella quale l'azienda invia i propri articoli ai centri logistici di Amazon, che si occupa poi direttamente della gestione dell'inventario, dell'evasione e spedizione degli ordini verso il cliente finale, del servizio clienti e dei resi.

Questa modalità comporta numerosi vantaggi:

- Accesso al badge "Prime" e maggiore visibilità degli articoli sulla piattaforma. I clienti Prime, pagando un canone annuo, hanno diritto a accedere a vantaggi come la spedizione gratuita su questi articoli e questo incrementa affidabilità e vendite.

- Tempi di consegna più rapidi, in quanto sono gestiti direttamente da Amazon nelle migliori condizioni possibili.

- Diminuzione del carico di lavoro per la logistica interna dell'azienda, in quanto molte merci vengono spedite a blocchi in anticipo verso i centri di logistica di Amazon.

Come anche degli svantaggi:

- Tariffa delle commissioni maggiore rispetto alle vendite FBM, in quanto sono da considerare i costi di gestione logistica, deposito, resi.

- Perdita di controllo diretto sulla spedizione, in quanto viene completamente presa in carico e gestita da Amazon

- Necessità di aggiornare le scorte nei magazzini di Amazon, ovvero rispedire gli articoli nel momento in cui si scende sotto una certa soglia, variabile per articolo in base alle vendite.

La differenza principale tra FBM e FBA risiede quindi nella responsabilità logistica e nella distribuzione dei costi e dei benefici: nel primo caso il venditore ha maggiore controllo operativo e un conseguente maggiore onere organizzativo e lavorativo; nel secondo caso, Amazon si prende carico di molte fasi del processo di vendita, evasione e post-vendita a fronte di maggiori commissioni.

6.4.2 Analisi articoli venduti con gestione Amazon

Nel contesto dell'azienda in analisi, il catalogo di vendita si compone principalmente di articoli videoludici (cd, dvd, blu-ray), libri e merchandising di vario tipo (action figures, portachiavi, oggetti vari). Questo tipo di merci ha influenzato in modo determinante i criteri di selezione degli articoli più adatti per la vendita tramite FBA.

Uno dei primi punti emersi dallo studio degli articoli è che la convenienza economica della vendita seguendo il modello FBA dipende in modo preponderante dal prezzo dell'articolo. Siccome Amazon applica commissioni fisse e variabili di gestione logistica, stoccaggio e vendita, il margine unitario sugli articoli a basso prezzo risulta spesso irrisorio. Per questa ragione, in accordo con l'azienda, abbiamo scelto di escludere dalla logica FBA tutti gli articoli con prezzo di vendita finale (PVC) minore di 30€, soglia identificata come limite sotto il quale il margine risulta troppo basso per coprire le effettive spese.

La selezione degli articoli è quindi basata su una serie di fattori:

- Prezzo di vendita maggiore di 30€, per garantire, come detto, un margine minimo.
- Dimensioni e peso contenuti, al fine di rendere minimi i costi di logistica e stoccaggio da parte di Amazon.
- Alta richiesta o novità, in modo da spedire nei centri articoli che hanno un'elevata probabilità di essere venduti, in modo da evitare costi di giacenza a lungo termine.
- Basso tasso di reso, il quale comporterebbe costi aggiuntivi

Dallo studio, è emerso che la vendita FBA viene utilizzata non tanto per aumentare la marginalità di profitto, quanto piuttosto come metodo per aumentare il volume delle vendite complessive. Gli articoli FBA, infatti, godono di maggiore visibilità sul marketplace e sono solitamente preferiti dagli utenti Prime.

Questa strategia, quindi, è mirata non tanto alla massimizzazione del profitto per unità venduta, ma bensì all'aumento della rotazione dell'inventario e alla scalabilità delle vendite, mantenendo comunque una soglia minima di margine per evitare di andare in perdita.

6.4.3 Motivazioni dell'integrazione diretta con il gestionale

L'integrazione diretta tra la vendita FBA e il gestionale aziendale MAGO rappresenta un passaggio strategico molto importante per mantenere il controllo e la tracciabilità delle attività di logistica, vendita e analisi operativa. Sebbene la

logistica e la gestione degli ordini cliente siano affidate direttamente a Amazon in questo modello, è fondamentale mantenere una replica sincronizzata dei movimenti e delle informazioni principali all'interno del gestionale aziendale.

Monitoraggio dell'inventario

La prima motivazione riguarda il monitoraggio dell'inventario FBA. Senza un'integrazione diretta, si avrebbero poche informazioni su ciò che è effettivamente disponibile presso i centri logistici di Amazon. Per fronteggiare questo problema, ho progettato e implementato un sistema di movimentazione automatica del magazzino, che registra su MAGO tutti gli invii di articoli verso Amazon. Inoltre, viene recuperato giornalmente lo stato delle giacenze di ogni articolo presente nei centri FBA, in questo modo si ha una visuale completa di ciò che è effettivamente disponibile, di ciò che non è disponibile (difettoso, danneggiato da cliente, danneggiato da Amazon, smarrito) e di ciò che è riservato (in transito verso un centro logistico, nel carrello di un cliente, riservato da Amazon per altre motivazioni).

Queste informazioni permettono di gestire un sistema che segnala quando, per un certo articolo, si scende sotto una determinata soglia (personalizzabile) di giacenza, in modo da attivare automaticamente la procedura per l'invio di una nuova spedizione. Questa logica consente di mantenere costante il volume delle vendite e ridurre al minimo il rischio di non disponibilità di certi articoli.

Recupero ordini e elaborazione dati

In parallelo alla sincronizzazione dell'inventario, il sistema progettato si occupa anche del recupero degli ordini evasi tramite FBA, che vengono registrati all'interno del gestionale, sia per fini contabili che per la creazione di report utili ai manager aziendali. L'integrazione, in questo modo, consente non solo di tenere traccia delle vendite, ma anche di identificare gli articoli che vengono venduti con maggiore frequenza, in modo da poter definire azioni strategiche specifiche, come:

- aumento delle scorte per gli articoli che vengono venduti con maggiore frequenza
- aggiornamenti del prezzo basati sulla domanda reale e sulle offerte proposte da altri venditori

6.4.4 Studio informazioni utili da mostre per la creazione di un Fulfillment

La definizione di un piano fulfillment efficace per la vendita in modalità FBA ha richiesto la disponibilità e l'elaborazione di un insieme di informazioni che

permettono di prendere decisioni strategiche sui rifornimenti, ottimizzare i margini di profitto e mantenere costante la disponibilità degli articoli nei centri logistici di Amazon. L'integrazione diretta tra il gestionale aziendale e il programma FBA ha permesso di centralizzare e visualizzare tutte queste informazioni in modo ordinato e strutturato, facilitando e orientando il processo decisionale degli utenti.

1. **Identificativo articolo**: per ogni articolo che deve essere spedito, sono essenziali le sue caratteristiche principali, come l'EAN, identificativo interno dal quale si parte per creare lo SKU utilizzato su Amazon, l'ASIN, identificativo utilizzato da Amazon, il titolo e una breve descrizione, informazioni essenziali per capire cosa si vuole spedire. Queste informazioni sono la base per ogni successiva analisi effettuata.

2. **Prezzo di vendita**: viene riportato il prezzo di vendita desiderato, salvato nel gestionale, e, tramite le API offerte da Amazon, viene anche recuperato e mostrato il prezzo competitivo nella piattaforma per lo stesso articolo attualmente presente sul marketplace. Questo confronto consente di valutare la fattibilità economica della vendita in FBA, in quanto avere un prezzo molto più alto del prezzo competitivo porterebbe a poche vendite.

3. **Costo di acquisto**: per ogni articolo, viene mostrato il prezzo di acquisto più aggiornato, recuperato come l'inferiore tra quelli proposti dai vari fornitori. Questo consente, oltre a calcolare i margini di vendita potenziali, anche di decidere se un articolo è economicamente sostenibile per le vendite col programma FBA, soprattutto in relazione alle commissioni imposte da Amazon.

4. **Stima sulle commissioni**: attraverso l'appropriato endpoint, viene recuperata una stima accurata per la commissione FBA applicabile a ogni articolo (in base a categoria, peso e dimensioni), fornendo una chiara visione della fattualità della vendita dell'articolo.

5. **Giacenze in tempo reale**: grazie all'integrazione prima descritta, le giacenze giornaliere degli articoli presenti nei centri logistici vengono recuperate e salvate. Questo consente di identificare in modo rapido i prodotti che sono in esaurimento, di confrontare le disponibilità attuali con le soglie di giacenza minima impostate (con la possibilità di modificarle). Inoltre, viene anche mostrata la giacenza interna nei magazzini dell'azienda in modo da avere una visione di come le giacenze vengono spartite tra i programmi FBA e FBM.

6. **Storico delle vendite**: per ogni prodotto, vengono analizzati più storici di vendita (calcolati grazie al recupero delle vendite FBA), su diversi intervalli temporali (3, 30 giorni e 6, 12 mesi), al fine di stimare la velocità di rotazione del singolo articolo e prevedere i futuri andamenti della domanda.

7. **Data di uscita del prodotto**: questo campo è particolarmente utile per identificare le novità e pianificare le spedizioni prima del lancio ufficiale del prodotto e avere, in questo modo, un vantaggio competitivo sugli altri venditori.

Queste informazioni vengono mostrate in una dashboard dedicata all'interno del portale web dell'azienda, con funzionalità aggiuntive di:

- **filtro** per ogni campo, in modo da trovare tempestivamente gli articoli di interesse
- **modifica** di alcuni attributi, come soglia minima e prezzo di vendita

Capitolo 7

Monitoraggio

7.1 Perché è importante misurare l'andamento del sistema

Nel contesto di questa tesi, riguardante un sistema per la gestione integrata dei processi di e-commerce multicanale, la misurazione costante delle performance risulta fondamentale per garantire un buon controllo operativo, l'identificazione repentina di problemi e anomalie e il miglioramento continuo.

L'adozione di un approccio basato su dati oggettivi e misurabili permette di:

- rilevare la presenza di possibili colli di bottiglia nei flussi di lavoro (ad esempio rallentamenti nel recupero degli ordini, aggiornamenti non eseguiti correttamente o problemi di gestione della memoria dovuti a un accesso in contemporanea da parte di diverse procedure alla medesima struttura dati);
- ottimizzare le risorse, sia tecniche che umane, intervenendo dove evidenziano delle difficoltà (ad esempio un'attenzione maggiore per il processo di preparazione fisica degli ordini da parte dei magazzinieri);
- migliorare la qualità del servizio, monitorando tempi di evasione, errori, ordini con giacenze insufficienti, approvvigionamenti non arrivati da parte dei fornitori, articoli non caricati correttamente nei vari marketplace, ecc.
- pianificare decisioni strategiche sulla base delle informazioni raccolte (ad esempio aumentare i prezzi di certi articoli per aumentare il profitto o impostare la spedizione gratuita su articoli molto richiesti per incentivarne la vendita).

Inoltre, date le numerose attività che devono essere periodicamente svolte, sono attive molteplici procedure schedulate durante l'arco di ogni giornata e la raccolta continua dei dati in esecuzione, esito di procedure ed eventuali errori è un modo

ottimale per permettere una manutenzione preventiva e per evitare interruzioni del servizio.

Un dettagliato e completo sistema di logging delle varie procedure permette di avere traccia di ogni operazione svolta, garantendo una facile e tempestiva individuazione di possibili errori, con una breve spiegazione per avere una comprensione a primo impatto della problematica.

In caso di errori gravi, è stato attivato un sistema di invio di mail automatico verso le persone responsabili, in modo da ricevere un avviso immediato dell'avvenimento di un'anomalia.

7.2 Esempi di KPI rilevanti

I **KPI** (Key Performance Indicator) rappresentano le metriche fondamentali per valutare l'efficienza e l'efficacia delle diverse procedure schedulate. Per monitorare l'andamento del lavoro svolto, ho diviso i KPI in diverse categorie:

- **KPI operativi**

  1. **Tempo medio di evasione ordine cliente** (dal momento di recupero dell'ordine al momento dell'avvenuta spedizione): si nota che gli ordini comprendenti articoli da fornitore richiedono più tempo per essere evasi, in quanto il tempo di consegna da parte dei fornitori varia molto in base alla stagione e alla quantità di articoli acquistati (la spedizione da parte del fornitore non avviene fino a che non si raggiunge un numero minimo di articoli ordinati).
  2. **Numero di ordini evasi entro 1 giornata**
  3. **Tempo di completamento delle procedure**: una misura sul tempo di esecuzione delle diverse procedure è importante per capire quali procedure si possono provare ad ottimizzare e per miglioare lo scheduling stesso.
  4. **Numero articoli aggiornati correttamente**: capire quanti articoli vengono correttamente aggiornati nei vari marketplace rispetto al totale degli articoli attivi permette di avere un'idea della probabilità di avere errori nei vari cataloghi online degli e-commerce.

- **KPI logistici**

  1. **Giacenza articoli nei magazzini di Amazon (FBA)**: grazie al recupero periodico dell'inventario disponibile si ha una statistica sullo stato di ogni articolo presente nei vari centri;

2. **Numero di ordini FBA/giorno**: metrica utile per comprendere ogni quanto viene effettuata una spedizione verso i centri logistici di Amazon per rifornire le giacenze;
3. **Articoli FBA sotto soglia**: essenziale per comprendere la velocità con cui si raggiunge la soglia di rifornimento, in modo da capire se alzarla o abbassarla.
4. **Articoli ordinati fornitore/arrivati**: rapporto essenziale per capire quanto sia affidabile l'approvvigionamento verso i vari fornitori.

- **KPI commerciali**

1. **PrezzoVendita/CostoTotale per articolo**: avere un'idea sul margine di proffito per le vendite permette di capire se la strategia di pricing è efficace o necessita di miglioramenti
2. **Totale vendite per marketplace**
3. **Totale vendite/mese**

L'identificazione di questi KPI tiene conto delle esigenze informative sia operative (monitoraggio del sistema) che manageriali (supporto decisionale).

7.3 Modalità di visualizzazione

Al fine di rendere accessibili e facilmente consultabili le informazioni aggregate attraverso i diversi KPI, ho scelto una modalità di visualizzazione semplice, ma flessibile e funzionale. Dopo aver valutato diverse opzioni, ho scelto di adottare un'approccio basato sulla creazione e esportazione in formato csv o Excel. Le motivazioni che mi hanno spinto a puntare su questo tipo di presentazione dei dati sono le seguenti:

- **Familiarità**: Excel è un software comunemente utilizzato in ambito manageriale e ciò rende le informazioni di facile comprensione, con la possibilità di eseguire ulteriori elaborazioni sui dati.
- **Flessibilità**: è possibile utilizzare filtri, formule, grafici dinamici e tabelle pivot per personalizzare l'analisi senza avere vincoli.
- **Portabilità e archiviazione**: i file generati possono essere facilmente modificati, salvati e condivisi, in modo da essere utilizzati anche offline.
- **Automazione**: la creazione dei file può essere facilmente automatizzata attraverso procedure che possono essere richiamate nel momento della necessità o schedulate.

Per recuperare questi KPI nel momento del bisogno, all'interno del portale web è presente una pagina dedicata. L'utente può selezionare l'intervallo temporale, il marketplace o altri filtri specifici per la generazione dei vari KPI, per ottenere il file richiesto generato dinamicamente basandosi sui dati reali.

Capitolo 8

Discussioni

8.1 Impatti dell'integrazione tra il gestionale e i vari marketplace

L'integrazione realizzata tra il gestionale aziendale MAGO e i diversi marketplace ha avuto un impatto significativo sull'organizzazione interna e sull'efficienza operativa. L'unificazione dei flussi informativi inerenti alle varie fasi della vendita online (catalogo, ordini, giacenze e resi) ha permesso di superare la precedente gestione manuale e frammentata, nella quale si procedeva manualmente con il caricamento del catalogo e il recupero degli ordini separatamente per ogni negozio.

La soluzione adottata ha introdotto numerosi benefici:

- **Riduzione degli errori**: grazie all'automazione dei processi di creazione e caricamento del catalogo, aggiornamento dell'inventario, approvvigionamento da fornitori, evasione ordini e gestione resi, il numero di errori umani nella trascrizione o trasmissione delle informazioni si è significativamente ridotto.

- **Maggiore tempestività**: il continuo aggiornamento dei dati permette una reazione più rapida ai cambiamenti di disponibilità, prezzo o altri attributi di vendita, evitando di avere offerte obsolete o non sincronizzate con il gestionale aziendale.

- **Efficienza nell'approvvigionamento dai fornitori**: la relazione diretta tra ordine cliente e ordine fornitore consente una pianificazione precisa e automatizzata nelle richieste di rifornimento, tenendo conto dei prezzi e delle tempistiche dei vari fornitori, in modo da ottimizzare l'acquisto.

- **Centralizzazione delle informazioni**: tutte le informazioni di rilievo vengono incanalate in un unico sistema, permettendo agli operatori e ai manager

aziendali di ottenere una visione completa dell'andamento generale, con la possibilità di analisi dettagliata dei vari processi.

- **Ottimizzazione delle strategie di vendita**: il monitoraggio continuo delle vendite e delle giacenze permette di definire regole sempre più precise e dettagliate (ad esempio, soglie di giacenza minima per gli ordini FBA, selezione dinamica del vettore di spedizione ottimale) che si traducono in un vantaggio competitivo.

8.2 Confronto tra soluzione proposta e utilizzo dashboard di Amazon

Uno degli obiettivi iniziali del progetto era superare le limitazioni del Seller Central di Amazon, il quale, pur mettendo a disposizione diversi strumenti di gestione, non risponde pienamente a tutte le esigenze di un'azienda che gestisce centinaia di migliaia di offerte in contemporanea, su più marketplace diversi.

Il confronto tra le 2 soluzioni si articola su diversi aspetti:

- **Scalabilità**: il Seller Central di Amazon si focalizza maggiormente sulle operazioni specifiche per i singoli prodotti; la necessità di gestire piu di 800.000 articoli rende la soluzione proposta molto più funzionale e snella.
- **Automazione**: invece delle azioni manuali (modifica attributi articoli, gestione ordini) possibili nella dashboard di Amazon, con l'implementazione personalizzata quasi tutte le attività sono completamente automatizzate e schedulate in job periodici che possono essere attivati anche su richiesta.
- **Tracciabilità**: nella soluzione progettata è presente un sistema di logging centralizzzato e dettagliato per ogni procedura in modo da tenere traccia di tutte le operazioni che vengono effettuate.
- **Integrazione**: la connessione diretta con il gestionale è fondamentale e il Seller Central è totalmente indipendente.
- **Reporting**: Amazon mette a disposizione un insieme di dashboard e report predefiniti, con limitate personallizzazioni; la soluzione proposta, invece, permette di ottenere KPI personalizzati e esportabili facilmente in formato Excel.

In particolare, l'utilizzo del Seller Central obbliga l'utente ad interagire direttamente sulla piattaforma messa a disposizione da Amazon, con interfacce non pensate e non ottimizzate per il caso aziendale in oggetto, in cui ci sono centinaia

di migliaia di articoli da gestire. L'implementazione progettata e implementata, invece, sfrutta direttamente le API di Amazon per gestire tutte le operazioni in modo automatizzato, tracciato e controllato. Inoltre, l'integrazione diretta con il gestionale MAGO permette di allineare automaticamente e tenere sincronizzato l'inventario, gli ordini e i resi, evitando disallineamenti o doppi inserimenti fatti per errore.

Capitolo 9

Conclusioni

9.1 Sintesi dei risultati raggiunti

Il progetto ha portato alla realizzazione di un sistema modulare, scalabile e automatizzato, in grado di integrare in modo efficace e funzionale il gestionale aziendale MAGO con i principali e-commerce (Amazon, IBS, Shopify, ecc.). Alcuni dei risultati più significativi raggiunti sono i seguenti:

- **Normalizzazione e centralizzazione dei dati**: mediante la creazione di tabelle ad-hoc (*OrdiniClienti*, *ItemMarketplace*, ecc.) i flussi eterogenei relativi a cataloghi, ordini e giacenze sono stati normalizzati e resi disponibili in modo uniforme e integrato.

- **Automazione**: tutte le fasi operative principali (dal recupero degli ordini cliente alla generazione di DDT e spedizioni, dalla comunicazione con i diversi fornitori alla sincronizzazione di inventario e giacenze) sono state rese automatiche tramite procedure schedulate per essere eseguite in modo periodico, minimizzando l'intervento di utenti.

- **Ottimizzazione logistica e pricing**: attraverso logiche ibride (modello FBM e FBA), la scelta dinamica del vettore di spedizione, l'applicazione di soglie di riordino e regole di pricing (prezzo di vendita finale calcolato basandosi su costo fornitore, sconti e commissioni), si è migliorata l'efficienza operativa e il margine medio.

- **Reporting e monitoraggio**: la definizione di specifici KPI e la generazione automatica dei relativi file Excel permettono monitoraggi continui sull'andamento del sistema e analisi strategiche periodiche, garantendo un insieme di informazioni affidabili per orientare nel modo ottimale le decisioni manageriali.

- **Tracciabilità completa**: grazie a un sistema di logging centralizzato e l'utilizzo di notifiche automatiche per la segnalazione di eventuali problematiche, è stata rafforzata la trasparenza dei processi e la risoluzione di anomalie che possono portare a errori.

9.2 Prospettive di sviluppo futuro

Nell'ottica di miglioramento continuo, la modularità con cui è stato creato il sistema rende possibili diversi ampliamenti per il futuro:

1. **Integrazione di altri marketplace**: è possibile aggiungere facilmente nuove piattaforme in futuro senza dover riprogettare le basi logiche ma semplicemente sviluppando le procedure necessarie e ampliando quelle già esistenti.

2. **Report avanzati**: sviluppo di dashboard interattive diverse dai file Excel per facilitare l'analisi in tempo reale dei vari dati recuperati e aumentare la visibilità aziendale nelle diverse piattaforme.

3. **App per dispositivi mobili**: in futuro si può pensare di sviluppare un'applicazione per smartphone, in modo da avere una piattaforma ad-hoc più facilmente utilizzabile in mobilità rispetto al portale web che, seppure completo e funzionale, risulta più adatto ad essere utilizzato da pc.

4. **Utilizzo di AI**: l'aggiunta dell'intelligenza artificiale può aiutare in modo significativo le scelte aziendali; nella gestione del catalogo, può essere utile nella creazione di descrizioni dettagliati dei prodotti; può anche essere utilizzata per un'assegnazione dinamica del prezzo basata su uno storico dati più ampio. Anche per quanto riguarda il supporto clienti, può essere funzionale sfruttare l'intelligenza artificiale per rispondere automaticamente a certe domande, permettendo di dedicare la forza lavoro verso altre mansioni di maggiore rilevanza.

La forza dell'architettura progettata risiede nella flessibilità evolutiva: le basi implementate permettono un adattamento continuativo ai cambiamenti tecnici e di mercato, mantenendo sempre un'elevata qualità e automazione delle diverse procedure.

Ringraziamenti

Grazie alla mia famiglia, che mi ha sempre sostenuto e spronato a dare il meglio in ogni momento.

Grazie *Andrea*, per avermi dato la possibilità di conciliare studio e lavoro nel migliore dei modi e per i preziosi consigli.

Grazie *Luca*, per avermi compreso come persona più di chiunque altro e per l'amico che sei. Potessi, dividerei questo traguardo con te. Ti voglio bene.

Grazie *Michelle* e *Elisa*, per essere state felici quanto me per i miei successi. La vostra amicizia è preziosa.

Grazie a tutti i miei amici che mi sono stati accanto e mi hanno aiutato ad alleggerire i periodi più stressanti.

Grazie a *me stesso*: per aver portato a termine questo percorso; per i momenti di felicità e spensieratezza, trascorsi con una birra in mano e i tuoi amici; per i momenti tristi, passati con la solita musica nelle cuffie; per aver imparato a stare da solo; per non aver mai cambiato le tue idee e essere sempre stato te stesso. Quando sarai felice, facci caso. Quando sarai giù di morale, ricordati sempre che è dal dolore che si può ricominciare. La vita prendila sempre com'è.

Bibliografia

[1] European Commission. *Organisation for Economic Co-operation and Development*. 2011. URL: https://www.oecd.org/digital/ieconomy/ (cit. a p. 1).

[2] Statista. *Global Retail E-commerce Sales 2014-2027*. 2024. URL: https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/ (cit. a p. 1).

[3] Amazon Services. *Amazon Selling Partner API*. 2024. URL: https://developer-docs.amazon.com/sp-api/ (cit. alle pp. 4, 10, 14).

[4] eBay. *eBay developer API*. 2024. URL: https://developer.ebay.com/docs (cit. alle pp. 4, 10).

[5] Shopify. *GraphQL Admin API reference*. 2025. URL: https://shopify.dev/docs/api/admin-graphql (cit. alle pp. 4, 10).

[6] Amazon Web Services. *AWS Signature Version 4*. 2025. URL: https://docs.aws.amazon.com/AmazonS3/latest/API/sig-v4-authenticating-requests.html (cit. a p. 10).

[7] Stoplight. *SOAP API*. 2024. URL: https://stoplight.io/api-types/soap-api (cit. a p. 10).

[8] Amazon Inc. *Requisiti relativi alle immagini del prodotto*. 2025. URL: https://sellercentral.amazon.it/help/hub/reference/external/1881 (cit. a p. 12).

[9] GS1. *EAN/UPC Family*. 2015. URL: https://www.gs1.org/docs/barcodes/GS1_Barcodes_Fact_Sheet-GS_EAN_UPC_family.pdf (cit. a p. 13).

[10] Keepa. *Keepa API*. 2024. URL: https://keepa.com/#!api (cit. a p. 14).

[11] IBM. *Cos'è l'EDI (Electronic Data Interchange)?* 2025. URL: https://www.ibm.com/it-it/think/topics/edi-electronic-data-interchange (cit. a p. 29).

[12] Software Freedom Conservancy. *About Selenium*. 2025. URL: https://www.selenium.dev/about/ (cit. a p. 30).

[13] Software Freedom Conservancy. *Locator strategies*. 2025. URL: https://www.selenium.dev/documentation/webdriver/elements/locators/ (cit. a p. 34).

[14] Amazon.com Services. *Amazon FBM (Fulfilled By Merchant)*. 2025. URL: https://sell.amazon.com/programs/fulfilled-by-merchant (cit. a p. 38).

[15] Amazon.com Services. *Amazon FBA (Fulfilled By Amazon)*. 2025. URL: https://sell.amazon.com/fulfillment-by-amazon (cit. a p. 38).

[16] Jungle Scout. *Amazon FBA vs FBM Comparison Guide: Which One is Better for Your Amazon Business?* 2024. URL: https://www.junglescout.com/resources/articles/amazon-fba-vs-fbm/ (cit. a p. 38).

[17] Amazon.com Services. *Standard selling fees*. 2025. URL: https://sell.amazon.com/pricing (cit. a p. 39).

[18] Gazzetta Ufficiale. *D.P.R. 14 agosto 1996, numero 472*. 1996. URL: https://www.normattiva.it/esporta/attoCompleto?atto.dataPubblicazioneGazzetta=1996-09-12&atto.codiceRedazionale=096G0487 (cit. a p. 41).

[19] Progress Software Corporation. *Telerik Reporting*. 2025. URL: https://www.telerik.com/products/reporting/creating-reports.aspx (cit. a p. 41).

[20] Spring. *Spedizione Ultimo Miglio per e-commerce*. 2025. URL: https://www.spring-gds.com/it/servizi-spedizioni-e-commerce/consegna-ultimo-miglio/ (cit. a p. 42).