Politecnico di Torino

Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria Informatica

A.a. 2025/2026

Sessione di laurea Marzo 2026

Integrazione real-time tra ERP e sistemi esterni: progettazione di un'architettura event-driven

Relatore: Giovanni Malnati

Candidata: Luciana Galliani

Tutor aziendale: Diego Franchino

Sommario

Negli ultimi anni, la crescente digitalizzazione dei processi aziendali ha reso sempre più centrale il tema dell'integrazione tra sistemi informativi eterogenei, chiamati a comunicare in modo affidabile e a garantire la consistenza dei dati tra applicazioni diverse. Le imprese moderne operano infatti attraverso ecosistemi applicativi complessi nei quali i sistemi ERP, fulcro dei dati amministrativi, commerciali e operativi, devono interagire con applicativi dipartimentali e piattaforme esterne con livelli di tempestività sempre maggiori.

In questo contesto, i meccanismi di integrazione tradizionali basati su polling periodico, pur essendo consolidati, evidenziano limiti strutturali quali elevata latenza nella propagazione delle modifiche, carico costante sul database e difficoltà di scalabilità in scenari che richiedono aggiornamenti in tempo reale.

Il presente lavoro di tesi, sviluppato durante il tirocinio presso Sistemi S.p.A., si propone di progettare un'architettura orientata agli eventi (event-driven) per introdurre un modello di integrazione più reattivo tra l'ERP eSOLVER e i sistemi esterni, con particolare attenzione all'integrazione con eSOLVER CRM.

La metodologia adottata ha previsto l'analisi dell'architettura esistente, l'individuazione delle criticità del modello basato su interrogazioni cicliche e la progettazione di una nuova soluzione fondata su una coda lavori transazionale. Quest'ultima può essere popolata tramite servizi REST, trigger SQL o direttamente dall'ERP. Tale soluzione introduce benefici operativi significativi, consentendo di rilevare rapidamente le modifiche ai dati e di attivare in modo puntuale i flussi di sincronizzazione. Il caso d'uso preso a modello riguarda l'esportazione delle anagrafiche dei potenziali clienti verso il CRM, scelto come scenario rappresentativo per valutare l'efficacia del paradigma basato sugli eventi.

Le verifiche svolte durante l'implementazione hanno dimostrato la fattibilità tecnica del modello ideato e la sua integrazione con l'infrastruttura esistente, evidenziando come l'adozione di tale architettura consenta di superare le principali limitazioni concettuali del polling.

In conclusione, il lavoro offre un approccio più moderno e reattivo per i contesti che richiedono maggiore tempestività, ponendo le basi per l'estensione del modello event-driven ad altre integrazioni aziendali.

Indice

Elenco delle tabelle

Elenco delle figure

Glossario

Sistemi dipartimentali

Software dedicati alla gestione di un singolo processo aziendale, specializzati nel supportare esigenze specifiche. Alcuni esempi di dipartimentali utilizzati in azienda sono: eSOLVER CRM, API2CART (per l'e-commerce) e gsped (per la comunicazione con i corrieri).

Enterprise Resource Planning (ERP)

Sistema software che integra e automatizza i principali processi aziendali come la finanza, le risorse umane, la produzione e la supply chain, in un'unica piattaforma. L'obiettivo è migliorare l'efficienza e il processo decisionale centralizzando le informazioni e i flussi di lavoro.

Extract, Transform, Load (ETL)

Processo di integrazione dei dati che combina, pulisce e organizza i dati provenienti da più fonti in un unico set di dati coerente per l'archiviazione in un data warehouse, data lake o altro sistema di destinazione [1].

Data Migrator System (DMS)

Suite di programmi che consente di configurare degli schemi di estrazione dei dati (da origini diverse), di disegnare delle regole per la loro trasformazione ed infine di creare un flusso per l'esecuzione dell'intero processo. Casi d'uso tipici: attività di migrazione oppure di generica trasformazione ETL.

Standard de facto

Standard che non ha una certificazione ufficiale, ma è diventato la norma in un settore a causa della sua ampia accettazione e diffusione sul mercato. Questa popolarità può derivare da vari fattori, come la convenienza, la superiorità tecnologica o la dominanza di un particolare prodotto o sistema.

Change Data Capture (CDC)

Tecnica di integrazione dei dati che acquisisce e trasmette le modifiche (inserimenti, aggiornamenti, eliminazioni) apportate a un database di origine in tempo quasi reale, aggiornando in modo efficiente i sistemi downstream.

Business Process Model and Notation (BPMN)

Standard grafico globale per rappresentare visivamente i processi aziendali, usando simboli definiti per descrivere fasi, attori, flussi di dati e decisioni. Serve a facilitare la comunicazione tra analisti, sviluppatori e stakeholder, supportare l'automazione, individuare colli di bottiglia e migliorare l'efficienza, facendo da ponte tra business e implementazione IT.

Sistemi In Rete (SIR)

Infrastruttura cloud di Sistemi S.p.A.

Human Resources (HR)

Sigla che indica il dipartimento aziendale che gestisce il personale, la selezione, la formazione e le relazioni interne.

Capitolo 1

Introduzione

1.1 Contesto

Negli ultimi anni, c'è stata una crescita siginificativa della digitalizzazione dei processi aziendali, mettendo sempre più in risalto il tema dell'integrazione tra sistemi eterogenei. Le aziende moderne operano infatti attraverso un ecosistema di applicativi che devono scambiarsi informazioni in modo affidabile, coerente e preferibilmente in tempo reale. In questo scenario, i sistemi gestionali svolgono un ruolo chiave: raccolgono e organizzano le informazioni fondamentali dell'azienda, come dati amministrativi, contabili, commerciali e operativi.

Assicurare una comunicazione efficiente tra il software gestionale aziendale e gli applicativi dipartimentali è tuttavia una sfida complessa, infatti richiede architetture di integrazione robuste, scalabili e capaci di adattarsi a volumi di dati e requisiti di tempestività sempre maggiori.

In questo contesto si inserisce la mia esperienza di tirocinio svolta presso **Sistemi S.p.A.**, azienda italiana specializzata nella progettazione, sviluppo e distribuzione di software gestionale per imprese, professionisti e associazioni. Le principali linee di prodotto dell'azienda includono software per la contabilità e gestione aziendale, la fatturazione elettronica e la gestione del personale.

Tra i diversi team che compongono l'area di sviluppo dell'azienda, ho collaborato con il gruppo **eSOLVER**, dedicato all'evoluzione dell'omonimo prodotto Enterprise Resource Planning (ERP) e alle sue integrazioni con sistemi dipartimentali.

eSOLVER è una soluzione gestionale completa e modulare, progettata per supportare le imprese di diversi settori industriali in ambito amministrativo, commerciale e produttivo, grazie a un'architettura flessibile e fortemente parametrizzabile. La piattaforma consente inoltre integrazioni con applicativi specialistici, tra cui eSOLVER CRM (piattaforma online che permette di gestire le relazioni con i clienti),

PROFIS (gestionale per servizi contabili e fiscali) e JOB (gestionale per l'elaborazione delle paghe e la gestione amministrativa del personale), permettendo alle aziende di centralizzare i flussi informativi.

Nel quadro aziendale esaminato, l'integrazione tra l'ERP e i sistemi dipartimentali è attualmente realizzata tramite Data Migrator Sistemi (DMS).

Tale motore opera secondo una modalità asincrona e schedulata: l'ERP viene interrogato ciclicamente mediante polling per identificare eventuali variazioni, che sono successivamente propagate ai sistemi esterni tramite servizi REST. Sebbene questa soluzione sia affermata e ampiamente utilizzata, presenta alcune limitazioni strutturali:

- elevata latenza tra la modifica dei dati e la loro sincronizzazione effettiva;
- carico aggiuntivo sul database, dovuto a interrogazioni periodiche anche in assenza di aggiornamenti;
- difficoltà di scalabilità e di adattamento a contesti che richiedono aggiornamenti in tempo reale.

Nonostante la solidità della soluzione, la crescita dei volumi informativi e la necessità di ridurre i tempi di propagazione hanno evidenziato limiti significativi, soprattutto negli scenari che richiedono maggiore tempestività o un elevato numero di aggiornamenti.

Alla luce di queste criticità, risulta evidente la necessità di una revisione architetturale: l'obiettivo è evolvere verso un modello più moderno e reattivo, capace di intercettare le modifiche e attivare immediatamente i flussi di sincronizzazione. Il modello proposto non sostituisce quello attualmente in uso, ma si pone come un'alternativa efficace per tutti quei casi in cui è richiesta una maggiore tempestività.

Il contesto applicativo su cui si è concentrato il lavoro riguarda l'integrazione tra **eSOLVER ed eSOLVER CRM**, con particolare attenzione ai flussi che gestiscono l'esportazione delle anagrafiche dei clienti potenziali: un caso d'uso rappresentativo, utile per valutare l'impatto dell'introduzione di un modello di sincronizzazione in tempo reale.

Sebbene alla conclusione del tirocinio il progetto non fosse ancora stato messo in produzione, la funzionalità sviluppata è stata inserita tra le novità previste per il rilascio di giugno 2026. Inoltre, non si esclude a priori la possibilità di estendere la soluzione proposta anche ad altri sistemi dipartimentali.

1.2 Obiettivo

L'obiettivo principale di questo elaborato è la progettazione e realizzazione di una nuova architettura per l'integrazione sincrona tra eSOLVER di Sistemi e i sistemi

esterni, mediante l'evoluzione del DMS verso un paradigma più ricettivo e orientato agli eventi.

In particolare, il progetto mira a:

- superare il modello basato su polling, introducendo un meccanismo di notifica diretta delle modifiche effettuate nell'ERP;
- introdurre una coda lavori transazionale, che può essere popolata in diversi modi: tramite servizio REST, trigger SQL o eSOLVER stesso;
- consentire al DMS di reagire in tempo quasi reale agli eventi generati, eseguendo sincronizzazioni puntuali e immediate;
- ridurre la latenza e il carico elaborativo, migliorando l'efficienza e la reattività complessiva del sistema;
- definire un'architettura scalabile ed estendibile per supportare nuove integrazioni in futuro.

1.3 Struttura della tesi

Il seguente elaborato è organizzato in cinque sezioni che analizzano ed esplorano le varie fasi di sviluppo del progetto di tesi. Dopo l'introduzione iniziale, i capitoli successivi sono così ordinati:

- **Capitolo 2. Background e stato dell'arte**: Questo capitolo presenta il quadro teorico e lo stato dell'arte relativi ai principali approcci di integrazione tra sistemi, con particolare attenzione alle architetture event-driven e alle soluzioni già esistenti. Viene inoltre analizzata la soluzione attualmente in uso, evidenziandone i limiti e le motivazioni che hanno portato alla definizione di un'architettura alternativa.
- **Capitolo 3. Architettura del sistema**: Il terzo capitolo illustra l'architettura progettata, descrivendone la struttura complessiva e analizzando nel dettaglio i principali componenti che la costituiscono, insieme al ruolo che ciascuno di essi svolge all'interno del flusso di integrazione.
- **Capitolo 4. Implementazione**: Questo capitolo descrive le soluzioni progettate per il popolamento della coda lavori. Dopo alcune modifiche preliminari alla struttura esistente, sono state analizzate diverse modalità di inserimento dei job con l'obiettivo di rendere il meccanismo flessibile e adattabile a differenti scenari applicativi.

- **Capitolo 5. Conclusioni e sviluppi futuri**: L'ultimo capitolo sintetizza i risultati raggiunti nel corso del lavoro e propone possibili evoluzioni future della soluzione sviluppata, con riferimento a miglioramenti architetturali e potenziali estensioni funzionali. futuri.

Capitolo 2

Background e stato dell'arte

In questo capitolo vengono presentati i principali approcci alla gestione dell'integrazione tra sistemi, con particolare attenzione ai modelli basati su polling e alle architetture event-driven, oggi ampiamente adottate per la propagazione immediata dei cambiamenti. Vengono inoltre analizzate alcune tecnologie moderne, come Apache Kafka e Debezium, che rappresentano soluzioni consolidate per la gestione di eventi e la cattura dei cambiamenti nei database. L'obiettivo è delineare il quadro tecnologico di riferimento, evidenziare vantaggi e limiti dei diversi approcci e comprendere le motivazioni che hanno portato l'azienda a scegliere una strada alternativa, basata sull'adattamento di strumenti già presenti nel proprio ecosistema.

2.1 Evoluzione dei modelli di integrazione

Negli ultimi anni, l'integrazione tra sistemi informativi eterogenei ha assunto un ruolo sempre più centrale per le aziende, rappresentando un fattore chiave per garantire coerenza dei dati, automazione dei processi e interoperabilità tra piattaforme. I sistemi moderni, spesso distribuiti, basati su microservizi o composti da applicazioni on-premise e cloud, richiedono modalità di comunicazione affidabili, tempestive e scalabili [2].

Tradizionalmente, tale integrazione è stata realizzata tramite elaborazioni pianificate (batch) o meccanismi di polling, in cui un sistema interroga periodicamente un altro per verificare la presenza di aggiornamenti. Questo modello, sebbene semplice, presenta limiti significativi:

- **Alta latenza**: un cambiamento nei dati può essere rilevato solo al successivo intervallo di polling;

- **Scarsa efficienza**: anche in assenza di novità, la risorsa continua a essere interrogata;

- **Scalabilità limitata**: aumentando il numero di oggetti o servizi da monitorare, la frequenza del polling cresce fino a saturare le risorse;

- **Carico concentrato in pochi momenti (es. batch notturni).**

Esempio pratico: sincronizzazione tra ERP e sistema esterno.

Supponiamo che l'ERP debba comunicare a un sistema esterno ogni nuova fattura. Con il polling, il sistema esterno controlla ogni 5 minuti se ci sono nuove fatture; se ne arriva una subito dopo il controllo, verrà vista con un ritardo di 5 minuti; se il numero di fatture cresce molto, il carico sul database aumenta esponenzialmente.

Per chiarire in modo concreto le limitazioni del modello basato su polling, si consideri il seguente scenario. Supponiamo che l'ERP debba comunicare a un sistema esterno ogni nuova fattura. Con un approccio a polling, il sistema esterno interroga l'ERP, ad esempio ogni 5 minuti, ma in alcuni contesti l'intervallo può essere ancora più ampio, per verificare la presenza di nuovi dati. Se una fattura viene registrata subito dopo un controllo, essa verrà rilevata solo al ciclo successivo, introducendo un ritardo minimo di 5 minuti. Inoltre, al crescere del numero di fatture, il volume delle interrogazioni aumenta in modo significativo, con un impatto proporzionale sul carico del database.

Per superare tali problematiche, si è progressivamente affermato il paradigma delle *architetture event-driven*, in cui ogni variazione di stato o modifica ai dati genera un evento in grado di attivare immediatamente le operazioni necessarie [3].

In questo modello:

- i sistemi non interrogano più periodicamente un altro sistema;

- è la sorgente dell'informazione che pubblica un evento quando avviene un cambiamento;

- i consumatori si registrano e vengono notificati in tempo reale.

Questo approccio riduce drasticamente la latenza, migliora l'efficienza nell'utilizzo delle risorse e permette una migliore distribuzione del carico computazionale.

In conclusione, il passaggio da un modello basato su polling a un'architettura event-driven rappresenta un'evoluzione naturale nell'integrazione tra sistemi aziendali, in particolare in contesti ad alta dinamicità dei dati.

2.2 Architetture event-driven

Le architetture event-driven si basano sul principio secondo cui i componenti del sistema reagiscono alla produzione e alla propagazione di eventi, che rappresentano modifiche rilevanti allo stato applicativo. Un evento può essere generato da un database, da un servizio applicativo o da un utente, e viene propagato attraverso meccanismi asincroni come code di messaggi o sistemi di streaming [4].

Tali architetture risultano particolarmente adatte ai moderni scenari di integrazione, dove i sistemi coinvolti sono numerosi, distribuiti e spesso appartenenti a domini tecnologici differenti. Tra i vantaggi più rilevanti si possono citare:

- **Bassa latenza**: le sincronizzazioni non dipendono da job periodici, ma avvengono alla generazione dell'evento;
- **Scalabilità**: i componenti sono disaccoppiati e gli eventi possono essere elaborati in parallelo;
- **Affidabilità**: l'utilizzo di broker di messaggistica garantisce durabilità e consegna degli eventi;
- **Flessibilità**: l'aggiunta di nuovi consumatori non richiede modifiche ai produttori.

Negli ultimi anni, l'adozione di piattaforme di messaggistica e stream processing come Apache Kafka, RabbitMQ e delle più recenti soluzioni cloud-native (ad esempio AWS EventBridge o Azure Event Grid) ha accelerato in modo significativo la diffusione di architetture event-driven all'interno dei sistemi enterprise.

Questi strumenti non solo consentono una gestione scalabile ed efficiente dei flussi di eventi, ma offrono anche funzionalità avanzate come il versionamento degli schemi, utile per mantenere la compatibilità tra produttori e consumatori; la persistenza dei messaggi, che garantisce che gli eventi non vadano persi; il replay degli eventi, che permette di rielaborare gli stream in caso di errori o di nuove logiche; e il monitoraggio degli stream, indispensabile per la supervisione e il debugging. Tali caratteristiche risultano essenziali per garantire consistenza, affidabilità e tracciabilità dei dati quando più applicazioni devono rimanere allineate in tempo reale.

Tra queste tecnologie, **Apache Kafka** si è affermato come standard de facto per la realizzazione di piattaforme di integrazione basate su event streaming, grazie alla sua capacità di gestire grandi volumi di dati in maniera distribuita, durabile e tollerante ai guasti [5]. In questo contesto si inserisce **Debezium**, una soluzione di Change Data Capture (CDC) che sfrutta Kafka per intercettare e propagare in modo affidabile ogni modifica ai dati presenti nei sistemi sorgente [6]. L'integrazione tra Kafka e Debezium rappresenta oggi uno dei modelli più efficaci per realizzare

sincronizzazioni real-time, superando i limiti dei meccanismi tradizionali basati su polling.

Sebbene nessuna di queste tecnologie venga adottata nella soluzione proposta in questo elaborato, esse costituiscono un riferimento importante nello stato dell'arte. Per questo motivo, nel paragrafo successivo verranno brevemente approfondite, così da fornire un quadro completo delle principali alternative rispetto all'approccio progettuale scelto.

2.2.1 Apache Kafka e Debezium

Apache Kafka è una piattaforma di messaggistica distribuita progettata per gestire flussi di dati in tempo reale, consentendo l'invio, la ricezione, l'archiviazione e l'elaborazione di messaggi in modo altamente scalabile e resiliente. Grazie alla sua architettura distribuita, è capace di sostenere elevati volumi di dati e un alto throughput, risultando particolarmente adatto a sistemi complessi e ad ambienti caratterizzati da una forte dinamicità [5].

I dati vengono organizzati in *topic*, canali logici che permettono ai *produttori* di inviare messaggi e ai *consumatori* di leggerli in modo indipendente, secondo il paradigma *publish–subscribe*. Ogni topic è suddiviso in *partizioni*, che consentono di distribuire il carico tra più broker e di garantire la scalabilità orizzontale del sistema. I *broker* del cluster, responsabili della persistenza e della distribuzione dei messaggi, garantiscono che le informazioni rimangano disponibili per i consumatori anche in caso di interruzioni o errori temporanei, grazie a meccanismi di replica e tolleranza ai guasti.

Altro aspetto rilevante è la persistenza dei messaggi su disco per un periodo configurabile di tempo, che permette ai consumatori di rileggere gli eventi e di gestire in modo affidabile eventuali ritardi o riavvii dei servizi. Questo approccio rende Kafka non solo un sistema di messaggistica, ma anche un vero e proprio *log distribuito* per eventi.

Kafka e Debezium sono spesso usati insieme per costruire pipeline di dati in tempo reale, in particolare in scenari di integrazione e replicazione dei dati tra diversi sistemi. In questi casi, Kafka funge da canale di comunicazione per i dati e Debezium si occupa di catturare e propagare i cambiamenti nel database.

Tale componente permette di mantenere allineati più sistemi senza ricorrere a meccanismi di sincronizzazione periodica o a interrogazioni invasive sui database di origine, migliorando l'efficienza e riducendo l'accoppiamento tra i servizi [6].

Il flusso di lavoro tipico di Debezium è il seguente:

1. **Monitoraggio delle modifiche al database**: si collega al database (ad esempio MySQL, PostgreSQL o MongoDB) e rileva i cambiamenti in tempo reale, come l'inserimento di nuove righe, la modifica di righe esistenti o la cancellazione di record.

2. **Cattura dei cambiamenti (CDC)**: cattura queste modifiche e le trasforma in eventi.

3. **Invio degli eventi a Kafka**: gli eventi vengono pubblicati su topic Kafka, dove possono essere consumati da altri servizi o applicazioni per elaborazioni o archiviazione in tempo reale.

Per queste ragioni, la combinazione di tali piattaforme risulta ideale per architetture a microservizi ed event-driven.

2.3 Contesto aziendale

Nel contesto aziendale analizzato, la soluzione attualmente in uso per l'integrazione con sistemi dipartimentali si basa su sincronizzazioni asincrone pianificate.

Di seguito è riportato lo schema architetturale corrente.

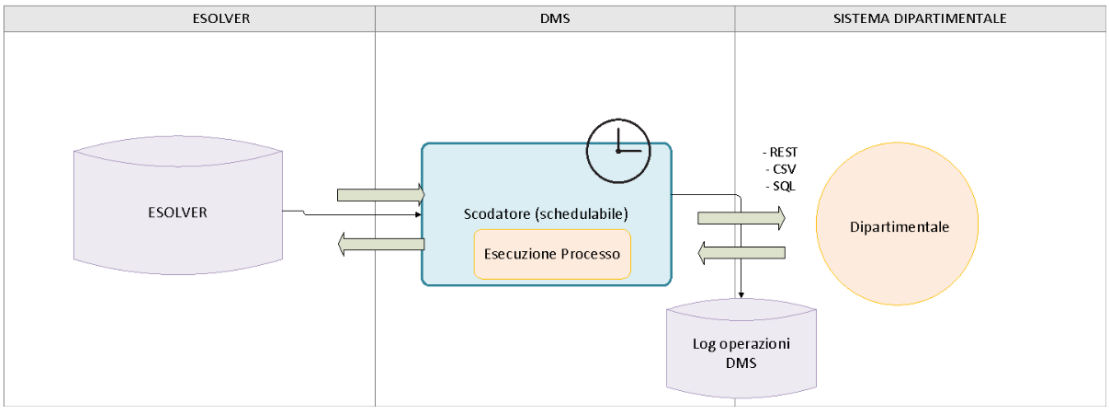


Figura 2.1: Architettura attualmente in uso

In questo scenario, il flusso di esportazione viene avviato esclusivamente su base schedulata. Nel caso di installazioni su cloud Sistemi (SIR), l'intervallo di esecuzione è configurabile dall'utente, ma non può essere impostato a valori inferiori ai 60 minuti. Di conseguenza, pur essendo parametrizzabile, la frequenza di sincronizzazione rimane comunque vincolata a una soglia minima temporale, che limita la tempestività dell'allineamento dei dati.

Il Data Migrator Sistemi (DMS) legge i dati da eSOLVER, li trasforma in file JSON e, per ogni file JSON creato, chiama il servizio REST specifico sul dipartimentale per l'acquisizione dei dati. L'esportazione avviene inoltre in modalità massiva: viene raccolto un delta di dati che viene inviato in blocco all'avvio della schedulazione.

Questo approccio comporta un ritardo fisiologico tra la modifica dei dati e la loro effettiva propagazione, riducendo la reattività dei sistemi dipartimentali. In

presenza di aggiornamenti frequenti, il delta può crescere rapidamente, generando picchi di carico o rallentamenti nel processo di sincronizzazione.

L'adozione di tecnologie come Apache Kafka e Debezium non è stata prevista per decisione dell'azienda, principalmente per ragioni operative e di semplificazione dell'infrastruttura. L'introduzione di tali strumenti avrebbe infatti comportato numerosi vincoli, in gran parte legati alla complessità del pacchetto software e alla necessità di competenze specifiche per l'installazione, la configurazione e la gestione dei servizi. Ciò sarebbe potuto risultare problematico per i Partner incaricati della distribuzione e dell'installazione dei prodotti, con un conseguente incremento della richiesta di assistenza.

A questo si somma un ulteriore elemento legato all'infrastruttura cloud proprietaria (SIR). In tale contesto, a seguito dell'acquisto del prodotto da parte del cliente, viene creata automaticamente un'installazione server completa (applicativo e database), senza l'inclusione di componenti di terze parti. L'introduzione di un sistema di messaggistica esterno avrebbe comportato una maggiore complessità anche lato cloud, richiedendo meccanismi aggiuntivi di installazione, configurazione e gestione automatica dei nuovi servizi. Questo avrebbe inciso sia sulla progettazione dell'infrastruttura sia sui processi di provisioning e manutenzione.

A ciò si aggiunge il potenziale costo legato a eventuali licenze per alcuni componenti, che avrebbe ulteriormente aumentato l'impatto economico e gestionale della soluzione.

L'integrazione di Debezium, inoltre, avrebbe introdotto un ulteriore carico sul database sorgente a causa del monitoraggio dei log delle transazioni, aumentando il consumo complessivo di risorse. Considerando che il database legato al prodotto eSOLVER non è distribuito e non utilizza container Docker per la gestione dei servizi, l'impiego di tecnologie di messaggistica e CDC così complesse risulterebbe eccessivo rispetto alle reali esigenze.

Con l'obiettivo di avvicinarsi a una sincronizzazione quasi "real-time", inviando le modifiche ai dipartimentali immediatamente dopo un aggiornamento delle tabelle di prodotto e superando i limiti dell'approccio schedulato senza adottare soluzioni esterne complesse, si è deciso di valorizzare strumenti già presenti in azienda, adattandoli alle nuove esigenze architetturali. In particolare, sono stati utilizzati il **DMS** e la **Coda Lavori**.

Entrambi gli strumenti offrono funzionalità affidabili per l'elaborazione e la gestione dei dati all'interno dell'ecosistema aziendale.

Il DMS consente di orchestrare le trasformazioni e i trasferimenti di dati tra sistemi eterogenei, mentre la Coda Lavori permette di gestire in modo controllato l'esecuzione di operazioni asincrone, diventando il motore principale della nuova logica di propagazione.

La soluzione proposta rappresenta dunque un compromesso efficace tra innovazione e sostenibilità operativa, permettendo di ottenere un miglioramento significativo

nel tempo di propagazione dei dati senza introdurre tecnologie distribuite non necessarie o difficilmente gestibili nell'attuale contesto aziendale.

Questi strumenti verranno descritti nel dettaglio nei capitoli successivi, insieme alle modifiche e agli sviluppi che hanno permesso di integrarli efficacemente nel nuovo processo di sincronizzazione.

Capitolo 3

Architettura del sistema

Il presente capitolo illustra l'architettura progettata per supportare l'integrazione applicativa e la comunicazione quasi in tempo reale tra eSOLVER e i sistemi dipartimentali. Dopo una panoramica della soluzione proposta e del confronto con l'architettura esistente, vengono descritti i principali componenti del sistema e il loro ruolo all'interno del flusso di integrazione. Il capitolo si conclude con l'analisi del flusso di sincronizzazione dei dati e con alcune considerazioni complessive sull'efficacia e sull'estendibilità dell'architettura.

3.1 Architettura scelta

Lo schema seguente illustra l'architettura che si vuole produrre per soddisfare i requisiti di comunicazione in tempo reale.

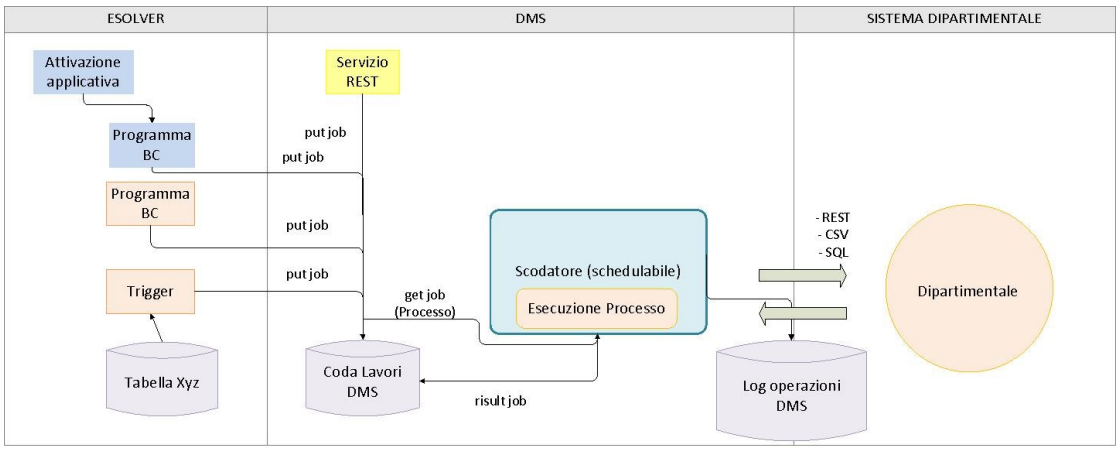


Figura 3.1: Architettura scelta

Come si può osservare, i componenti principali coincidono con quelli attualmente in uso: eSOLVER, DMS e il Sistema Dipartimentale. Nel presente elaborato, quest'ultimo sarà indicato come eSOLVER CRM, poiché è il sistema utilizzato per l'attività di tesi. A questa architettura viene inoltre aggiunta la Coda Lavori che, sebbene nella figura 3.1 sia rappresentata all'interno del DMS, viene considerata come un componente indipendente.

La presenza di una Coda Lavori già esistente ha rappresentato una motivazione rilevante nella scelta architetturale, poiché ha consentito di valorizzare l'infrastruttura presente, riutilizzando tale componente e introducendo esclusivamente i moduli e le funzionalità mancanti. Su questa base, l'architettura è stata progettata per rispondere alle necessità comunicative in tempo reale attraverso il disaccoppiamento tra produttori e consumatori, riducendo l'accoppiamento temporale e permettendo la gestione asincrona delle operazioni.

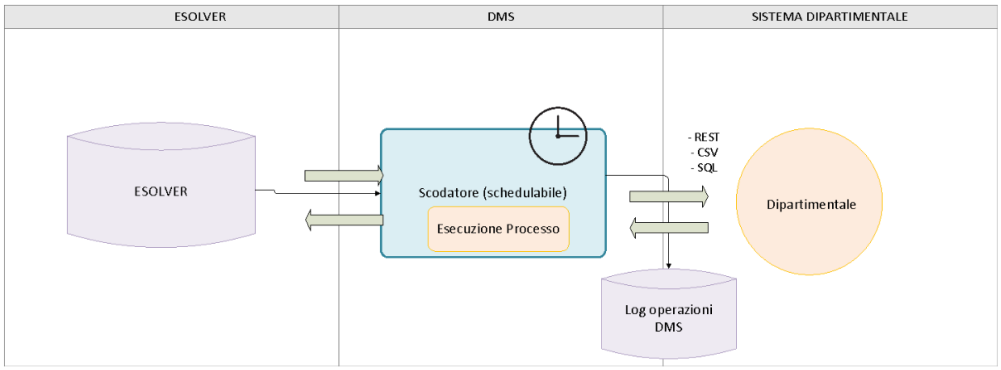
La Coda Lavori svolge inoltre la funzione di buffer, consentendo di assorbire eventuali picchi di carico generati dai produttori e di distribuirli in modo controllato verso i consumatori, migliorando la resilienza complessiva del sistema. L'integrazione di log operativi garantisce affidabilità e tracciabilità delle elaborazioni, rendendo possibili meccanismi di retry, audit e monitoraggio. La flessibilità dei canali di comunicazione esposti dal DMS, tramite REST, CSV o SQL, permette infine di interfacciarsi con sistemi dipartimentali eterogenei, sia legacy sia moderni.

L'adozione di questa architettura introduce tuttavia una maggiore complessità nella gestione dei flussi applicativi e nel monitoraggio dei job, dovuta alla presenza di componenti intermedi e di processi asincroni. Tale compromesso è stato ritenuto accettabile in quanto compensato dai benefici in termini di robustezza, scalabilità ed estendibilità del sistema.

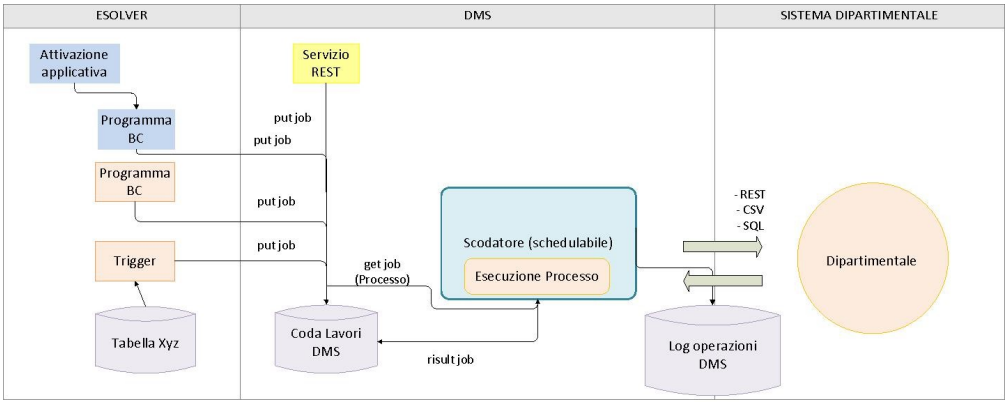
Sulla base di queste considerazioni, gli obiettivi di progettazione sono stati definiti per ottenere un'architettura in grado di supportare comunicazioni in tempo reale, facilitare l'integrazione tra sistemi, garantire affidabilità e monitorabilità, e favorire un'evoluzione modulare e scalabile nel tempo.

La differenza sostanziale tra l'architettura attuale e quella proposta risiede nella modalità di comunicazione tra eSOLVER e DMS. In particolare, la comunicazione non avviene più in modo diretto, ma viene mediata dalla Coda Lavori, che riceve i job attraverso differenti modalità, quali servizio REST, trigger o lo stesso eSOLVER. Ad ogni caricamento di un job nella Coda Lavori viene invocato un meccanismo di schedulazione che avvia l'esecuzione del processo specifico associato.

Di seguito sono riportate le architetture a confronto.



(a) Attuale



(b) Proposta

Figura 3.2: Confronto architetture

Alla luce delle considerazioni esposte, la soluzione architetturale adottata risponde in modo adeguato ai requisiti funzionali e non funzionali individuati.

3.2 Componenti principali del sistema

Di seguito vengono presentati i principali componenti evidenziati nella Figura 3.1, descrivendone innanzitutto la struttura e approfondendone successivamente il ruolo e il comportamento all'interno dell'architettura.

3.2.1 Il prodotto eSOLVER

eSOLVER è il software gestionale ERP di punta sviluppato da Sistemi, progettato specificamente per le piccole e medie imprese italiane che necessitano di una

soluzione completa per gestire tutti i processi aziendali, dall'amministrazione alla produzione, fino alla logistica.

Essendo un ERP modulare, copre diverse aree dell'azienda. Non si tratta di un semplice programma di contabilità, bensì di un sistema integrato:

- gestisce i processi amministrativi aziendali, includendo ciclo attivo, ciclo passivo e logistica, con funzionalità avanzate di contabilità generale e analitica;
- supporta la gestione dei processi produttivi aziendali attraverso funzionalità specifiche per le diverse tipologie di attività, offrendo inoltre strumenti di analisi economica della produzione;
- consente all'azienda di accedere in modo chiaro e immediato alle informazioni necessarie per il controllo dei processi aziendali, mettendo a disposizione strumenti per la creazione e la modifica di report e analisi.

Un'altra caratteristica rilevante di eSOLVER riguarda l'organizzazione dei dati, che sono suddivisi per mantenere una chiara distinzione tra le logiche di elaborazione e gli adempimenti normativi:

- *DBGruppo* (Gruppo di Lavoro):Identificato da un codice alfanumerico di due caratteri, è l'entità che consente di effettuare dei raggruppamenti degli archivi specifici, costituendo i cosiddetti gruppi di elaborazione. Contiene i dati di base e le tabelle generali;
- *DB Ditta* (Dati Specifici): Rappresenta il soggetto fiscalmente rilevante a cui fanno capo gli adempimenti normativi. In questo livello risiedono i dati specifici della partita IVA, come fatture, movimenti contabili e carichi di magazzino.

Il principio strutturale fondamentale di questa installazione è che a ogni gruppo corrisponde una sola ditta. Di conseguenza, la separazione tra DBGruppo e DB Ditta non ha lo scopo di condividere anagrafiche tra aziende diverse, ma serve a mantenere un ordine rigoroso all'interno della singola realtà aziendale, separando l'architettura generale di elaborazione (il Gruppo) dai dati prettamente transazionali e fiscali (la Ditta).

In Figura 3.3 è riportata la schermata principale di eSOLVER.

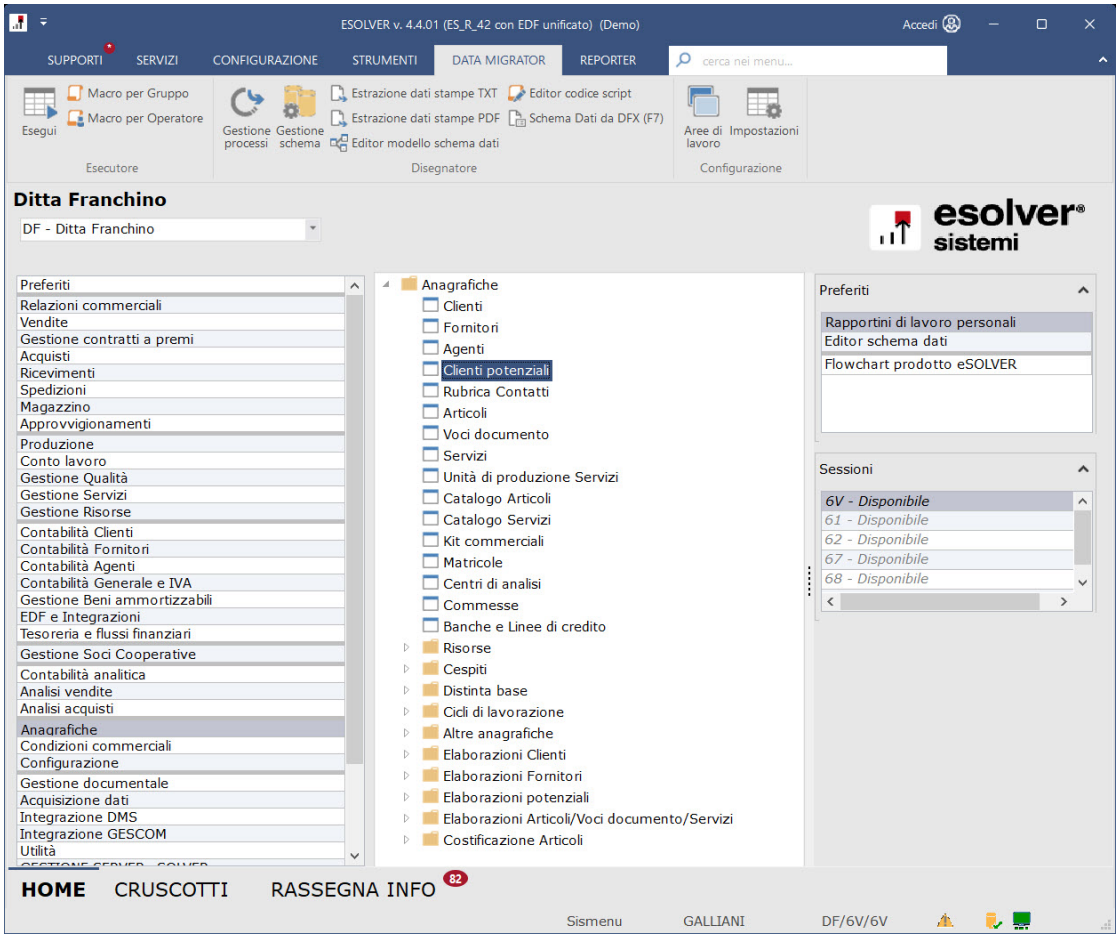


Figura 3.3: Schermata principale di eSOLVER

Sotto il profilo tecnico, si basa su un ambiente proprietario che sfrutta un linguaggio ricco di funzioni e oggetti grafici per creare interfacce chiare e di facile utilizzo.

Il *Linguaggio BC (Basic Codifier)* è un DSL (Domain Specific Language), ovvero un linguaggio creato su misura che serve a scrivere le regole di gestione, ad esempio come si calcola l'IVA, come si scarica il magazzino, come si valida una fattura. Il codice BC è "astratto" rispetto alla tecnologia sottostante. Questo significa che lo sviluppatore scrive la regola una volta sola in BC, e poi il "motore" di Sistemi (il framework) si occupa di tradurla ed eseguirla sia in ambiente Windows (Client-Server) che in ambiente Web/Cloud.

Al livello inferiore dell'architettura è presente il *database Microsoft SQL Server*, che custodisce i dati grezzi e garantisce prestazioni elevate e affidabilità.

Inoltre, l'architettura del software è concepita per essere aperta, permettendo integrazioni strutturate con applicativi dipartimentali o specifici già presenti in

azienda.

L'integrazione tra eSOLVER e i software di terze parti avviene attraverso il DMS, la cui architettura e funzionamento sono descritti in dettaglio nella Sezione 3.2.2.

Per gestire un'integrazione sono necessari tre pilastri fondamentali:

- *Dipartimentali*: Software specializzati in singoli processi o settori verticali (es. CRM o trasporti);
- *Ambiti di Integrazione*: Rappresentano l'attivazione effettiva dell'integrazione, definendo se il collegamento con il dipartimentale è valido per l'intera installazione di eSOLVER o solo per specifici gruppi;
- *Flussi di Integrazione*: Sono i processi operativi che eseguono la sincronizzazione dei dati, ricalcando l'organizzazione dei progetti nel DMS per un'esigenza di coerenza strutturale e operativa.

Nell'architettura proposta, le sue principali responsabilità consistono nella generazione di job in corrispondenza di eventi applicativi rilevanti (ad esempio la creazione di un'anagrafica cliente o l'aggiornamento di un record) e nella definizione di meccanismi di schedulazione che inoltrano tali job verso la coda del DMS.

3.2.2 Il Data Migrator Sistemi

Data Migrator Sistemi (DMS) è una suite di strumenti sviluppata da Sistemi per la gestione dei processi di migrazione e integrazione dei dati, supportando in modo strutturato la configurazione, il disegno e l'esecuzione dei processi ETL (Extract, Transform, Load). Il sistema consente l'estrazione dei dati da sorgenti eterogenee, la loro trasformazione secondo regole configurabili e la successiva importazione nei sistemi di destinazione.

Il principale ambito di utilizzo di DMS è la migrazione dati da soluzioni concorrenti verso i prodotti di Sistemi; tuttavia, lo strumento è progettato per supportare più in generale processi di integrazione applicativa e di trasformazione dei dati in contesti gestionali complessi.

Un elemento distintivo di DMS è la possibilità di riutilizzare schemi e regole di trasformazione, riducendo l'effort nei progetti successivi. Lo strumento è inoltre pensato per essere utilizzato anche da utenti applicativi, pur richiedendo un minimo di competenze tecniche nei casi di trasformazioni più complesse.

Dal punto di vista architetturale, DMS si basa su una libreria di connettori dati che rende indipendenti le sorgenti e le destinazioni dalle regole di trasformazione intermedie. Le sorgenti possono includere file strutturati e non strutturati (CSV, testi a lunghezza fissa, XML, JSON, PDF), database relazionali (tramite accesso

nativo o ODBC), fogli Excel, servizi web REST e aree di cache temporanea. Analogamente, i dati trasformati possono essere scritti verso file, database o servizi web, mantenendo invariato il modello di trasformazione.

Il processo ETL in DMS si articola in tre fasi principali:

- **Estrazione**, che comprende la selezione dei dati dalle sorgenti, l'eventuale applicazione di filtri e ordinamenti;
- **Trasformazione**, che include la mappatura dei campi sorgente-destinazione, la normalizzazione dei dati e la definizione delle regole di generazione dei record di output (uno-a-uno, aggregati, multi-record o strutture per web service);
- **Acquisizione**, fase finale in cui i dati trasformati vengono importati nei sistemi di destinazione, tipicamente tramite le procedure standard dei prodotti di Sistemi.

La suite prevede innanzitutto una **gestione delle aree di lavoro**, organizzata su due livelli gerarchici (*Macro Aree* e *Aree*), che consente di strutturare e centralizzare i progetti, offrendo una visione complessiva delle attività sviluppate e un accesso ordinato ai relativi contenuti.

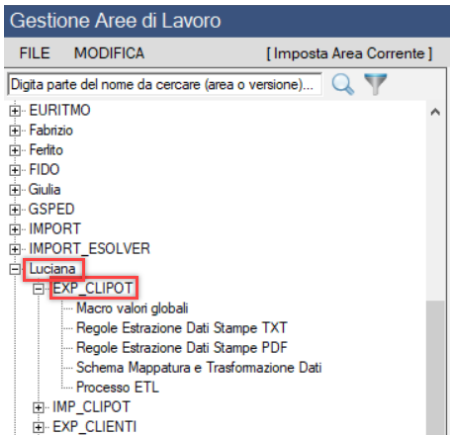


Figura 3.4: Esempio di Macro Area e Area

Il nucleo operativo del sistema è rappresentato dagli **strumenti di disegno**, che permettono di definire in modo formale le regole di estrazione, trasformazione e orchestrazione dei flussi:

- specifici editor consentono di progettare le regole di estrazione dei dati da stampe testuali e PDF, trasformando le informazioni estratte in strutture record assimilabili a tabelle;

- l'**Editor Regole Processo Migrazione Dati** permette di modellare il flusso di esecuzione complessivo, definendo la sequenza delle operazioni ETL e l'eventuale interazione con l'utente per la parametrizzazione del processo; i progetti sono salvati in file XML con estensione *MPRules* e di seguito verrà indicato come *MP Editor*. Vedi Figura 3.5;
- l'**Editor Regole Mappatura e Trasformazione Dati** consente di definire le sorgenti, le destinazioni e le regole di conversione dei dati, specificando le relazioni tra campi e le logiche di normalizzazione; i relativi progetti sono salvati in file XML con estensione *DTMRules* e tale strumento verrà di seguito indicato come *DTM Editor*. Vedi Figura 3.6.

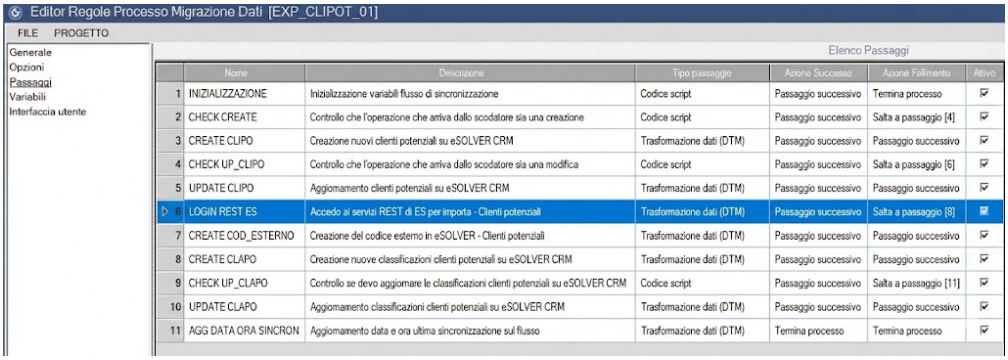


Figura 3.5: Esempio di Editor Regole Processo Migrazione Dati

A completamento della fase di progettazione, DMS fornisce una serie di **strumenti di utilità**, tra cui l'**Editor Schema Dati** per la gestione degli schemi dati standard, la **Generazione Schema Dati da DFX** per la creazione automatica degli schemi a partire dalle strutture di prodotto, l'**Editor Codice Script** per la definizione di script riutilizzabili a supporto dell'implementazione di logiche applicative comuni, le cui funzioni sono salvate in file con estensione *DMScript*, e l'**Editor Macro Valori Globali** per la configurazione delle variabili macro utilizzate negli schemi di progetto dell'area di lavoro, la cui definizione è salvata nel file *SisDM_MacroValues.xml*.

L'esecuzione dei processi avviene tramite **Esecutore del processo di migrazione**, applicazione incaricata dell'esecuzione delle regole definite in fase di progettazione; tale componente, indicato anche come *runtime DMS*, rappresenta l'eseguibile distribuito ed eseguito presso l'utente finale.

Nel contesto dell'architettura complessiva, il Data Migrator Sistemi ha il compito di elaborare e normalizzare i dati provenienti dai sistemi sorgente e di consegnarli ai sistemi dipartimentali destinatari. Tra questi rientra eSOLVER CRM, oggetto di analisi nella Sezione 3.2.4.

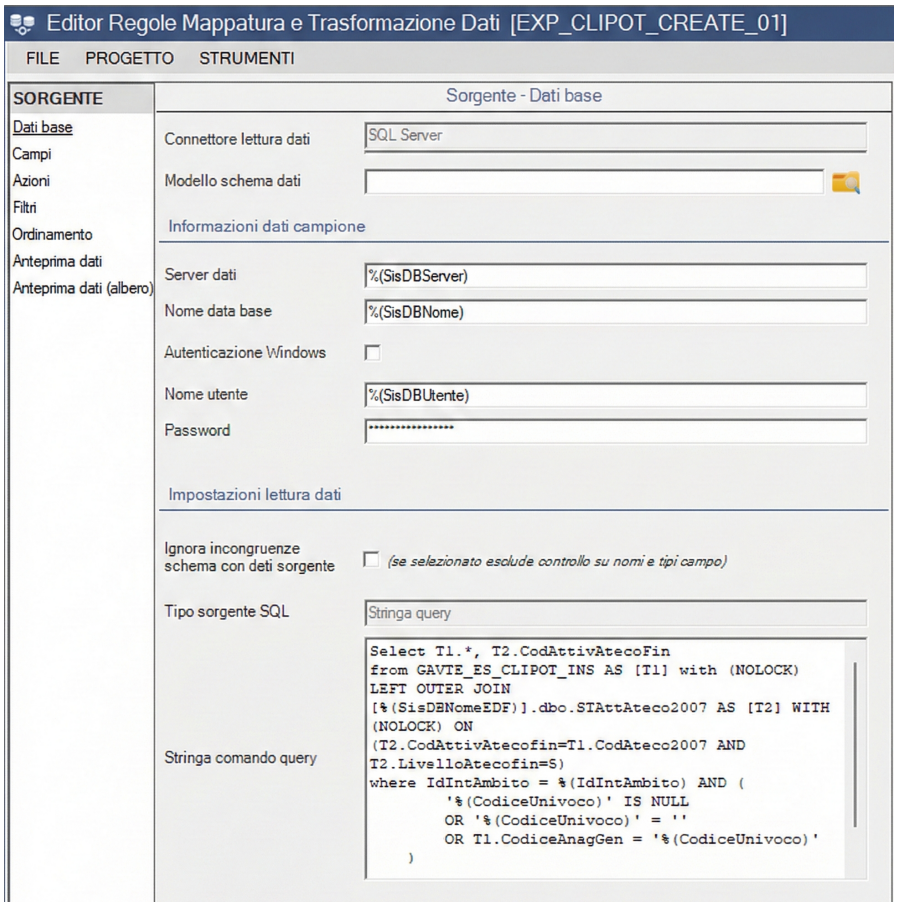


Figura 3.6: Esempio di Editor Regole Mappatura e Trasformazione Dati

3.2.3 La Coda Lavori

La Coda Lavori costituisce un elemento centrale dell'architettura del sistema, operando come punto di snodo critico tra la generazione dei dati, la loro orchestrazione e i servizi di elaborazione. La sua funzione principale è quella di fungere da intermediario affidabile tra i processi produttori di dati e i processi consumatori. Questo strumento, sviluppato di recente, è stato ulteriormente affinato e completato nel corso delle attività descritte nel presente elaborato, migliorandone l'usabilità e l'affidabilità complessiva.

In termini concreti, la struttura altro non è che una tabella, denominata **SIDMSCodaLavori**, implementata secondo un modello di accodamento FIFO (First In, First Out), in cui le richieste vengono inserite in ordine cronologico e processate secondo la sequenza di arrivo.

Ogni elemento della coda, denominato *job*, rappresenta un'unità atomica di elaborazione, contenente tutte le informazioni necessarie affinché si possa completare correttamente l'attività richiesta. Queste informazioni includono dati relativi al processo da elaborare, tra cui la macro area, l'area e il nome del flusso di esecuzione.

Per gestire in modo ottimale l'esecuzione asincrona dei processi, alla Coda Lavori è affiancato uno strumento dedicato, lo **Scodatore**. Questo componente si attiva automaticamente ogni volta che un nuovo job viene inserito nella coda, e si occupa di iterare tutti i processi considerati validi, ossia quelli da eseguire o da rieseguire. Per ciascun job valido, lo Scodatore richiama il DMS per avviare l'elaborazione del processo specifico. Al termine dell'esecuzione, il risultato viene riportato direttamente in SIDMSCodaLavori, aggiornando in maniera puntuale i campi di firma relativi all'esecuzione e consentendo un tracciamento completo dello stato dei processi. Grazie a questo meccanismo, l'inserimento di nuovi lavori avviene in modo asincrono e completamente automatizzato, riducendo l'intervento manuale e garantendo che tutti i job vengano elaborati secondo l'ordine corretto.

Nell'architettura pensata per soddisfare le esigenze dell'azienda, la Coda Lavori può essere popolata attraverso diverse modalità: mediante un **servizio REST**, tipicamente utilizzato in contesti web; tramite **trigger**, nel caso in cui si voglia implementare un'integrazione personalizzata direttamente all'interno del sistema; oppure tramite **eSOLVER stesso**, utilizzando il linguaggio BC, che rappresenta il caso d'uso più comune.

Tra queste modalità, il servizio REST risultava già implementato al momento dell'inizio delle attività. Il lavoro di tesi ha avuto come obiettivo l'analisi e l'ottimizzazione di tale servizio, insieme alla definizione progettuale delle altre due soluzioni di integrazione, sviluppate e documentate nel corso dell'elaborato.

Un altro aspetto rilevante è la gestione degli errori. Infatti c'è la possibilità di ripetere automaticamente i job falliti, riducendo il rischio di perdite di dati o elaborazioni incomplete.

Quindi, questo strumento non è semplicemente un buffer, ma un componente architetturale strategico, capace di coordinare processi eterogenei, garantire la continuità operativa e assicurare l'affidabilità dei flussi di dati nell'intero ecosistema.

3.2.4 Il sistema eSOLVER CRM

eSOLVER CRM è una soluzione software di Customer Relationship Management progettata per supportare le aziende di produzione, commercio e servizi nella gestione strutturata delle relazioni con il mercato [7]. Il sistema è nativamente integrato con l'ERP eSOLVER, garantendo coerenza e continuità dei dati tra area commerciale, amministrativa e operativa.

La piattaforma consente di gestire in modo completo tutti i flussi legati al cliente, dall'acquisizione iniziale fino all'assistenza post-vendita, favorendo al contempo una comunicazione efficace sia interna sia esterna all'azienda. Tutte le attività e le comunicazioni con i clienti vengono archiviate e rese consultabili, permettendo la centralizzazione delle informazioni sul singolo cliente. In particolare, eSOLVER CRM valorizza anche le informazioni non strutturate, quali eventi, e-mail, documenti e messaggistica, incrementando il valore informativo dello storico aziendale.

Dal punto di vista tecnologico, è basato su *VTENEXT*, una piattaforma CRM open source evoluta che fornisce funzionalità avanzate per la gestione dei processi commerciali, delle relazioni con i clienti e dei flussi di lavoro. Su tale base sono state sviluppate specifiche personalizzazioni per allineare il modello dati e le funzionalità alle esigenze e alle logiche applicative di eSOLVER. Rispetto alla versione standard, sono state adattate e integrate le principali entità gestionali, tra cui ditte, clienti potenziali, clienti effettivi (con relative sedi e informazioni di classificazione e condizioni commerciali), contatti commerciali, articoli e servizi, nonché i preventivi di vendita.

Il CRM consente di gestire in modo distinto ma coordinato *clienti potenziali* e *clienti effettivi*.

I clienti potenziali, comprendenti lead e prospect, rappresentano soggetti con cui l'azienda ha instaurato un primo contatto commerciale ma che non hanno ancora generato una transazione. Per tali entità, il sistema supporta la raccolta e l'organizzazione delle informazioni, il monitoraggio delle opportunità e la pianificazione delle attività commerciali e di marketing finalizzate alla conversione.

Ad esempio, un cliente potenziale può essere un'azienda intercettata tramite una campagna marketing, per la quale vengono registrati nel CRM i dati anagrafici di base, il contatto commerciale di riferimento, l'origine del lead e le prime opportunità commerciali. Tali informazioni possono essere importate verso eSOLVER per garantire la disponibilità del dato anche ai processi gestionali e amministrativi.

Invece, i clienti effettivi sono quei soggetti con cui esiste già un rapporto commerciale attivo; per questi, eSOLVER CRM permette una gestione completa di relazioni,

attività, comunicazioni e servizi di assistenza, mantenendo un collegamento diretto con i dati amministrativi e di fatturazione presenti nell'ERP.

Oltre alla gestione dei clienti, il sistema consente anche la consultazione delle anagrafiche e dei contatti dei fornitori ampliando così la visione complessiva delle relazioni aziendali. Tra le funzionalità principali rientrano:

- la gestione automatizzata dei processi tramite strumenti basati su BPMN;
- la sincronizzazione delle anagrafiche con eSOLVER;
- il supporto alla collaborazione interna attraverso messaggistica, calendario e chat;
- la gestione dell'assistenza clienti mediante una piattaforma di ticketing;
- funzionalità avanzate di reporting e analisi, con la possibilità di utilizzare report standard o personalizzati e di condividerli tra gli utenti.

A supporto della descrizione funzionale, in Figura 3.7 è riportata la schermata principale di eSOLVER CRM, che illustra visivamente l'organizzazione delle informazioni e delle funzionalità disponibili per l'utente.

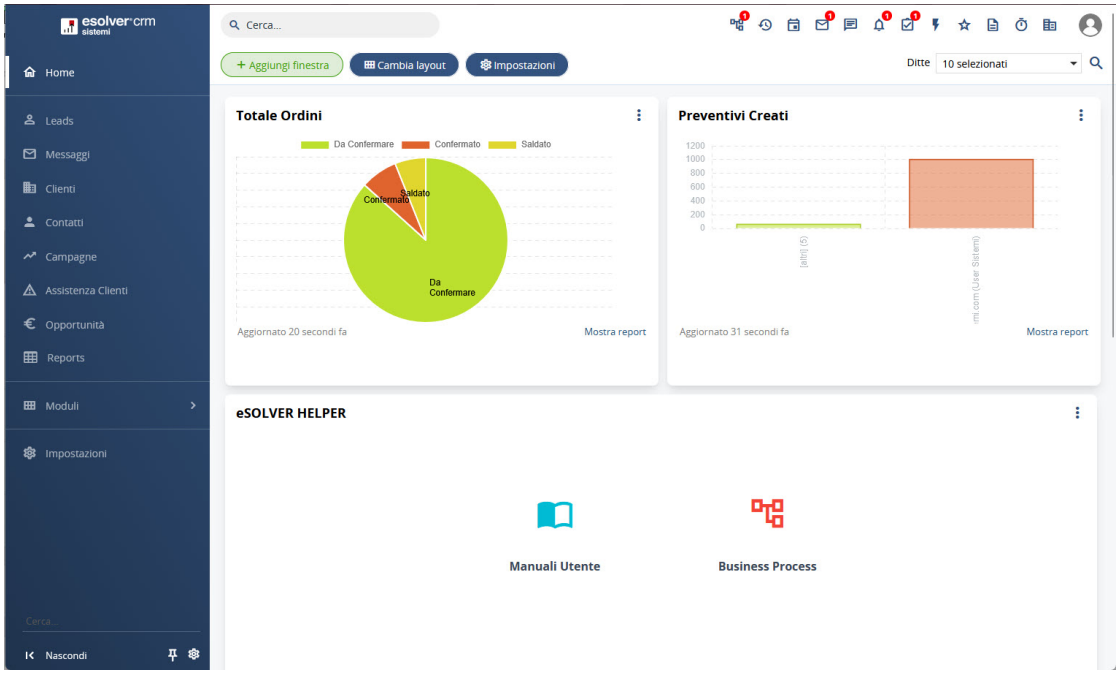


Figura 3.7: Home eSOLVER CRM

A partire da questa struttura, eSOLVER CRM interagisce con il sistema gestionale eSOLVER attraverso specifici flussi di integrazione applicativa, finalizzati a

garantire l'allineamento e la coerenza delle informazioni anagrafiche e documentali tra i due sistemi.

Nel contesto dell'*integrazione applicativa tra eSOLVER ed eSOLVER CRM*, sono coinvolte diverse entità anagrafiche e documentali, fondamentali per garantire la continuità dei dati all'interno dell'architettura complessiva del sistema informativo aziendale.

A fini del progetto di tesi, l'analisi si concentra esclusivamente sull'entità dei **clienti potenziali**.

Per tale entità è prevista un'integrazione bidirezionale: da eSOLVER verso eSOLVER CRM, tramite processo di esportazione, e da eSOLVER CRM verso eSOLVER, tramite processo di importazione. In particolare, sul gestionale vengono acquisiti sia i nuovi clienti potenziali generati dalle attività di CRM, sia eventuali aggiornamenti relativi a potenziali già esistenti.

Contestualmente, vengono esportati verso il CRM i clienti potenziali attivi codificati sul gestionale, completi delle relative classificazioni per ditta, tipicamente nella fase di avviamento dell'integrazione, oltre alle successive modifiche dei potenziali acquisiti dal CRM.

Nell'ambito del presente elaborato, l'attenzione è rivolta unicamente al **flusso di esportazione**.

3.3 Flusso di sincronizzazione dei dati

Come anticipato nelle sezioni precedenti, la Coda Lavori può essere popolata attraverso modalità differenti; tuttavia, indipendentemente dal meccanismo di inserimento, le informazioni che devono essere fornite risultano sempre le stesse. In particolare, per ogni processo è necessario specificare:

- la `Macro area`, l'`Area` e il file `.MPRules` associato al processo, così da identificare correttamente il contesto applicativo e le regole da utilizzare per l'esecuzione;
- il `tipo di operazione` da eseguire, ossia inserimento, variazione o eliminazione, ciascuna delle quali è codificata rispettivamente con i valori 1, 2 e 3;
- il `DBGruppo`;
- l'`ID del flusso di integrazione`;
- l'`ID dell'anagrafica da esportare`.

La presenza sia del file .MPRules associato al processo sia dell'ID del flusso di integrazione potrebbe apparire ridondante; tuttavia, entrambe le informazioni

risultano necessarie per consentire il collegamento tra la nuova modalità di integrazione e quella standard. Nel Capitolo 4 verrà approfondito il significato di tale collegamento e le modalità con cui esso viene realizzato.

L'inserimento di un nuovo record all'interno della tabella della Coda Lavori comporta l'attivazione di una schedulazione precedentemente configurata. L'utilizzo di una schedulazione consente di disaccoppiare il momento di generazione del dato dal momento della sua effettiva elaborazione, permettendo al sistema di gestire i processi in modo asincrono e controllato. Questo approccio garantisce una maggiore robustezza dell'architettura, evitando l'esecuzione immediata dei processi in contesti potenzialmente critici e consentendo una gestione più efficiente dei carichi di lavoro, nonché il recupero automatico in caso di errori o interruzioni, consentendo inoltre di rispettare i requisiti di quasi real-time richiesti, senza compromettere la stabilità del sistema gestionale primario.

In fase iniziale, il record viene inserito con Stato pari a zero (*Da eseguire*); è proprio questa condizione che innesca il funzionamento dello Scodatore. Quest'ultimo è un programma che scorre ciclicamente tutti i processi con Stato pari a zero (*Da eseguire*) o sette (*Da rieseguire*) e provvede a richiamare il DMS per l'esecuzione del processo associato.

Il processo standard di esportazione dei clienti potenziali, denominato **EXP_CLIPOT**, è strutturato come un normale flusso di esportazione delle entità anagrafiche tramite il DMS. Questo processo garantisce che i clienti potenziali attivi codificati sul gestionale siano correttamente esportati verso eSOLVER CRM con le relative classificazioni per ditta, assicurando la continuità delle informazioni nella fase di avviamento dell'integrazione e nelle successive sincronizzazioni.

Il sistema eSOLVER CRM è il destinatario finale dei dati elaborati dal DMS. L'interazione può avvenire tramite API REST, con endpoint esposti dal dipartimentale, scambi di file CSV o accesso SQL, a seconda delle interfacce disponibili. Il DMS adatta il formato e gestisce la consegna, assicurando un esito standardizzato e registrando l'esecuzione nei log operativi.

3.4 Considerazioni sull'architettura

L'architettura descritta evidenzia come eSOLVER sia concepito come un sistema ERP in grado di supportare in modo strutturato i principali processi aziendali e di garantire coerenza e riutilizzo dei dati attraverso una chiara separazione tra dati comuni e dati specifici di ditta. L'adozione di un linguaggio proprietario di alto livello, indipendente dalla tecnologia sottostante, consente inoltre di mantenere le regole di business centralizzate e riutilizzabili, facilitando l'evoluzione del sistema nel tempo.

In questo contesto, il DMS assume un ruolo chiave come elemento di collegamento tra eSOLVER e i sistemi esterni, permettendo l'integrazione controllata dei dati mediante flussi strutturati, job schedulati e una gestione asincrona basata su coda lavori. Le scelte architetturali adottate consentono di garantire affidabilità, tracciabilità e disaccoppiamento tra i diversi componenti coinvolti nei processi di integrazione.

Nel complesso, l'architettura proposta risponde efficacemente ai principali requisiti non funzionali individuati, quali scalabilità, resilienza e manutenibilità, consentendo al sistema di gestire in modo affidabile l'aumento dei volumi di dati e l'evoluzione dei flussi applicativi nel tempo.

L'intervento progettuale descritto ha inoltre permesso di estendere un'infrastruttura esistente, adattandola a nuovi scenari di integrazione applicativa senza introdurre dipendenze rigide o modifiche invasive ai sistemi coinvolti.

Sulla base di queste considerazioni, il capitolo successivo si concentra sull'implementazione concreta dell'architettura proposta, analizzando nel dettaglio le soluzioni tecniche adottate, i meccanismi operativi del DMS e le modalità con cui i flussi di integrazione vengono effettivamente realizzati.

Capitolo 4

Implementazione

Questo capitolo descrive le soluzioni progettate e implementate per il popolamento della coda lavori. Dopo alcune modifiche preliminari alla struttura esistente, sono state analizzate e realizzate diverse modalità di inserimento dei job nella coda, con l'obiettivo di rendere il meccanismo flessibile e adattabile a differenti scenari applicativi. In particolare, vengono presentate tre principali strategie: l'utilizzo di un servizio REST per l'inserimento diretto delle richieste, l'integrazione con il gestionale tramite eventi generati durante le operazioni applicative e, infine, l'impiego di trigger a livello di database per intercettare automaticamente le modifiche ai dati. Per ciascun approccio vengono illustrati il funzionamento, le motivazioni progettuali e le principali caratteristiche implementative.

4.1 Modifiche preliminari

4.1.1 Analisi tecnica

Prima di procedere con la progettazione della soluzione proposta, è stata condotta un'analisi tecnica della struttura dati esistente utilizzata dal sistema per la gestione dei job. In particolare, l'attenzione si è concentrata sulla tabella **SIDMSCodaLavori**, già presente nell'infrastruttura Sistemi.

L'analisi ha avuto l'obiettivo di comprendere il ruolo della tabella all'interno del sistema, identificare i campi maggiormente rilevanti e valutare come poterla riutilizzare o estendere per supportare il nuovo meccanismo di integrazione tra sistemi nel prodotto eSOLVER.

La Tabella 4.1 riporta la struttura della tabella analizzata.

Campo	Descrizione
IdLavoroDMS	Identificativo univoco del job
TipoOperazione	Tipo di operazione da eseguire: 1- inserimento, 2-variazione, 3-eliminazione
TipoEntita	Codice per identificare il tipo di entità (es: PRO per i clienti potenziali)
GruppoArchivi	Valorizzato con il DBGruppo
DataInserimento	Data in cui è stata inserita l'operazione da eseguire
OraInserimento	L'orario in cui è stata inserita l'operazione da eseguire
DataOraUltimaEsecuzione	Data in cui è stata eseguita l'ultima volta questa operazione
StatoLavoro	Indica lo stato di esecuzione del job (es: 0-da eseguire, 1-in esecuzione, etc...)
NumeroTentativiEseguiti	Il numero di tentativi eseguiti
EsitoCodiceErrore	Codice dell'errore eventualmente restituito dal DMS
EsitoMsgErrore	Testo dell'errore eventualmente restituito dal DMS
Argomento2	Campo libero
Argomento3	Campo libero
Argomento4	Campo libero
Argomento5	Campo libero
MacroArea	Macro area del DMS
Area	Area del DMS
Regole	Regole del DMS

Tabella 4.1: Struttura della tabella SIDMSCodaLavori

Tra i campi presenti nella tabella, quelli più rilevanti per il funzionamento del sistema sono i seguenti:

- **MacroArea, Area e Regole**: consentono di identificare quale processo del DMS deve essere eseguito e dove esso è definito all'interno del sistema.
- **TipoOperazione**: indica la tipologia di operazione da eseguire.
- **GruppoArchivi**: permette di individuare il gruppo di archivi coinvolti nell'elaborazione del lavoro.
- i **Campi Liberi**: campi utilizzati per memorizzare parametri aggiuntivi necessari all'esecuzione del lavoro.

Dall'analisi sono emersi alcuni aspetti che hanno influenzato le scelte adottate nella realizzazione della soluzione proposta. In particolare, la struttura della tabella e le modalità con cui essa viene utilizzata dai processi applicativi esistenti hanno imposto alcuni vincoli che non potevano essere modificati senza compromettere il funzionamento del sistema.

Uno degli elementi più rilevanti riguarda il campo **TipoOperazione**. Nella configurazione attuale del sistema, tale campo viene utilizzato per determinare la logica di elaborazione del job presente nella coda ed è strettamente collegato al campo **ARG1**, che viene valorizzato in funzione della tipologia di operazione da eseguire. Questa relazione rappresenta un elemento strutturale del meccanismo di gestione dei job e, per tale motivo, non è stato possibile modificarne il comportamento.

Nel contesto delle integrazioni realizzate in eSOLVER, tuttavia, il campo ARG1 viene normalmente valorizzato direttamente dal flusso di integrazione. Questa differenza di utilizzo ha reso necessario individuare una soluzione che consentisse di mantenere la compatibilità con il funzionamento della tabella *SIDMSCodaLavori* senza alterare la logica già presente nel sistema.

Per questo motivo, sfruttando la presenza di campi ad uso libero (ARG2-ARG5), si è scelto di utilizzare il campo **ARG2** per memorizzare l'identificativo del flusso di integrazione.

Inoltre è emersa un'ulteriore esigenza legata alla gestione delle anagrafiche da sincronizzare. Il meccanismo di integrazione esistente, infatti, non prevedeva l'esportazione di una singola anagrafica, ma operava su insiemi di dati.

Per poter gestire anche il caso di sincronizzazione puntuale di un' anagrafica, è stato necessario individuare un modo per trasmettere al processo l'identificativo dell'entità da elaborare. Anche in questo caso si è scelto di utilizzare uno dei campi liberi disponibili nella tabella, in particolare il campo **ARG3**.

Il valore contenuto in ARG3 rappresenta il codice univoco dell'anagrafica da esportare e viene assegnato alla variabile macro **CodiceUnivoco**.

In questo modo è stato possibile estendere il comportamento del sistema introducendo la possibilità di sincronizzare singole entità, mantenendo al tempo stesso la compatibilità con il funzionamento preesistente del meccanismo di integrazione.

4.1.2 Processo di esportazione delle anagrafiche

Innanzitutto è stato necessario intervenire sul processo che gestisce il flusso di esportazione dei clienti potenziali verso il sistema CRM, denominato *EXP_CLIPOT_01.MPRules*.

Il processo originario era stato progettato per gestire l'esportazione di insiemi di anagrafiche e non prevedeva la possibilità di esportare una singola entità. L'introduzione della coda lavori ha quindi richiesto un adattamento del processo, al fine

di supportare anche il caso di sincronizzazione puntuale.

Durante la fase di progettazione sono state valutate due possibili alternative per integrare questa nuova funzionalità all'interno dell'architettura esistente.

Alternativa 1: processo unico con passaggi condizionali

La prima soluzione prevedeva il mantenimento di un unico processo generale, introducendo al suo interno logiche condizionali in grado di distinguere i diversi casi di esecuzione (ad esempio tramite condizioni *if/else*).

Vantaggi:

- Minore duplicazione della logica applicativa, in quanto tutte le operazioni sono centralizzate all'interno dello stesso flusso.
- Gestione unificata delle modifiche: eventuali aggiornamenti alla logica comune si applicano automaticamente a tutti i casi.
- Riduzione dell'overhead di manutenzione, non essendo necessario gestire più processi separati.

Svantaggi:

- Maggiore complessità interna del processo, che può risultare meno leggibile e più difficile da mantenere.
- Maggiore rischio di regressioni, in quanto modifiche introdotte per supportare nuovi casi potrebbero impattare il comportamento esistente.

Alternativa 2: creazione di nuovi processi dedicati

La seconda soluzione prevedeva di mantenere invariato il processo esistente e introdurre due nuovi processi dedicati alla gestione della sincronizzazione puntuale delle anagrafiche, uno per le operazioni di creazione (*CREATE*) e uno per le operazioni di aggiornamento (*UPDATE*).

Vantaggi:

- Isolamento completo della nuova funzionalità rispetto al flusso esistente.
- Maggiore chiarezza logica, in quanto ogni processo rappresenta un caso d'uso ben definito.
- Migliore manutenibilità nel caso in cui i nuovi flussi evolvano indipendentemente nel tempo.

Svantaggi:

- Duplicazione di logica comune tra i diversi processi.

- Possibile rischio di incoerenza nel tempo qualora modifiche comuni non vengano propagate correttamente in tutti i processi.

Considerando che i nuovi processi avrebbero avuto una struttura sostanzialmente identica a quella del processo esistente, si è ritenuto preferibile adottare la prima soluzione. In questo modo è stato possibile evitare ridondanze nella logica applicativa e mantenere centralizzata la gestione del flusso di integrazione.

Le seguenti Figure mostrano il confronto tra il processo originale di esportazione e la versione modificata, evidenziando i nuovi passaggi introdotti per gestire anche la sincronizzazione di una singola anagrafica.

	Nome	Descrizione	Tipo passaggio	Azione Successo	Azione Fallimento	Attivo
1	INIZIALIZZAZIONE	Inizializzazione variabili flusso di sincronizzazione	Codice script	Passaggio successivo	Termina processo	☑
2	CREATE CLIPO	Creazione nuovi clienti potenziali su eSOLVER CRM	Trasformazione dati (DTM)	Passaggio successivo	Passaggio successivo	☑
3	UPDATE CLIPO	Aggiornamento clienti potenziali su eSOLVER CRM	Trasformazione dati (DTM)	Passaggio successivo	Passaggio successivo	☑
4	LOGIN REST ES	Accedo ai servizi REST di ES per importa - Clienti potenziali	Trasformazione dati (DTM)	Passaggio successivo	Salta a passaggio [6]	☑
5	CREATE COD_ESTERNO	Creazione del codice esterno in eSOLVER - Clienti potenziali	Trasformazione dati (DTM)	Passaggio successivo	Passaggio successivo	☑
6	CREATE CLAPO	Creazione nuove classificazioni clienti potenziali su eSOLVER CRM	Trasformazione dati (DTM)	Passaggio successivo	Passaggio successivo	☑
7	UPDATE CLAPO	Aggiornamento classificazioni clienti potenziali su eSOLVER CRM	Trasformazione dati (DTM)	Passaggio successivo	Passaggio successivo	☑
8	AGG DATA ORA SINCRON	Aggiornamento data e ora ultima sincronizzazione sul flusso	Trasformazione dati (DTM)	Termina processo	Termina processo	☑

Figura 4.1: Processo originale

	Nome	Descrizione	Tipo passaggio	Azione Successo	Azione Fallimento	Attivo
1	INIZIALIZZAZIONE	Inizializzazione variabili flusso di sincronizzazione	Codice script	Passaggio successivo	Termina processo	☑
2	CHECK CREATE	Controllo che l'operazione che arriva dallo scodatore sia una creazione	Codice script	Passaggio successivo	Salta a passaggio [4]	☑
3	CREATE CLIPO	Creazione nuovi clienti potenziali su eSOLVER CRM	Trasformazione dati (DTM)	Passaggio successivo	Passaggio successivo	☑
4	CHECK UP_CLIPO	Controllo che l'operazione che arriva dallo scodatore sia una modifica	Codice script	Passaggio successivo	Salta a passaggio [6]	☑
5	UPDATE CLIPO	Aggiornamento clienti potenziali su eSOLVER CRM	Trasformazione dati (DTM)	Passaggio successivo	Passaggio successivo	☑
6	LOGIN REST ES	Accedo ai servizi REST di ES per importa - Clienti potenziali	Trasformazione dati (DTM)	Passaggio successivo	Salta a passaggio [8]	☑
7	CREATE COD_ESTERNO	Creazione del codice esterno in eSOLVER - Clienti potenziali	Trasformazione dati (DTM)	Passaggio successivo	Passaggio successivo	☑
8	CREATE CLAPO	Creazione nuove classificazioni clienti potenziali su eSOLVER CRM	Trasformazione dati (DTM)	Passaggio successivo	Passaggio successivo	☑
9	CHECK UP_CLAPO	Controllo se devo aggiornare le classificazioni clienti potenziali su eSOLVER CRM	Codice script	Passaggio successivo	Salta a passaggio [11]	☑
10	UPDATE CLAPO	Aggiornamento classificazioni clienti potenziali su eSOLVER CRM	Trasformazione dati (DTM)	Passaggio successivo	Passaggio successivo	☑
11	AGG DATA ORA SINCRON	Aggiornamento data e ora ultima sincronizzazione sul flusso	Trasformazione dati (DTM)	Termina processo	Termina processo	☑

Figura 4.2: Processo modificato

Tutti i cambiamenti successivamente descritti sono stati effettuati nella macro area personale “Luciana”.

Per comprendere meglio le modifiche introdotte, è utile analizzare più nel dettaglio le principali fasi del processo di esportazione.

4.1.3 Modifica dello script di integrazione

La prima fase del processo, denominata *Inizializzazione*, ha il compito di preparare il contesto di esecuzione del flusso. In questa fase viene eseguito lo script *eSOLVERCRM.DMScript*, il quale consente di valorizzare le variabili macro utilizzate dal processo e restituisce la variabile Abilitato (0 o 1). Se il flusso non risulta abilitato (0), allora il processo termina, altrimenti procede al passaggio successivo.

A seguito delle considerazioni descritte nella sottosezione 4.1.1, il primo intervento effettuato ha riguardato la modifica di tale script.

In particolare, è stata introdotta una logica condizionale che consente di distinguere le due diverse modalità di esecuzione.

Se il campo **ARG2** risulta valorizzato, significa che l'operazione è stata generata tramite la coda lavori e che il valore contenuto in tale campo rappresenta l'identificativo del flusso di integrazione. Questo valore viene quindi assegnato alla variabile macro **IdFlusso**, che consente allo script di determinare il flusso di integrazione.

Se invece il campo **ARG2** non è valorizzato, il sistema continua a utilizzare la modalità di funzionamento già esistente, basata sul campo **ARG1**.

La Figura 4.3 mostra il confronto tra la versione originale dello script e quella modificata.

```
'=========================================================================
'  Funzioni  eSOLVER CRM
'=========================================================================

Function leggiConfigCRM ()

    leggiConfigCRM = 0

    'inizializzo la connessione al DB_ESOLVER
     DataReader.InitBySQLServer "ESOLVER_DB", Macro("SisDBServer"), Macro("SisDBNome"), Macro("SisDBUtente"), Macro("SisDBPassword")

    Macro("IdFlusso") = Macro("CmdArg1")
```

(a) Prima

```
'=========================================================================
'  Funzioni  eSOLVER CRM
'=========================================================================

Function leggiConfigCRM ()

    leggiConfigCRM = 0

    'inizializzo la connessione al DB_ESOLVER
     DataReader.InitBySQLServer "ESOLVER_DB", Macro("SisDBServer"), Macro("SisDBNome"), Macro("SisDBUtente"), Macro("SisDBPassword")

    if  Macro("CmdArg2") <> "" then
      Macro("IdFlusso") = Macro("CmdArg2")

      if  Macro("CmdArg3") <> "" then
         Macro("CodiceUnivoco") = Macro("CmdArg3")
      else
         Macro("CodiceUnivoco") = ""
      end if
    else
      Macro("IdFlusso") = Macro("CmdArg1")
    end if
```

(b) Dopo

Figura 4.3: Modifiche allo script di eSOLVER CRM

Di conseguenza, quando il processo viene attivato tramite la coda lavori, lo script verifica la presenza del valore nel campo ARG2 per poi determinare se il flusso di integrazione da eseguire è abilitato e, successivamente, controlla il contenuto del campo ARG3 per stabilire se l'operazione riguarda una specifica anagrafica.

4.1.4 Introduzione dei nuovi passaggi nel processo

Oltre alle modifiche apportate allo script di integrazione eseguito nella fase 1, nel processo sono stati aggiunti tre nuovi passaggi (Figura 4.2):

- **CHECK CREATE**
- **CHECK UP_CLIPO**
- **CHECK UP_CLAPO**

Tutti e tre i passaggi sono implementati tramite script che utilizzano blocchi condizionali *if-else* per determinare il percorso di esecuzione del processo. In

particolare, questi controlli permettono di verificare che l'esecuzione del flusso avvenga tramite lo *Scodatore* della coda lavori e di stabilire il tipo di operazione da eseguire sull'anagrafica.

Le condizioni verificate consentono infatti di distinguere due possibili azioni:

- **Inserimento** dell'anagrafica, identificato dal valore `=1`;
- **Aggiornamento** dell'anagrafica, identificato dal valore `=2`.

Il caso di **Eliminazione**, identificato dal valore `=3` nel campo *TipoOperazione*, non è gestito dal sistema eSOLVER nell'ambito delle integrazioni considerate e pertanto non è stato incluso tra gli scenari supportati dal processo.

Se tali condizioni sono vere, si passa alla fase successiva del processo, altrimenti si salta alla prossima condizione da verificare.

Il passaggio **CHECK CREATE** ha il compito di determinare se il processo deve proseguire con la creazione di una singola anagrafica nel sistema CRM.

Successivamente, i passaggi **CHECK UP_CLIPO** e **CHECK UP_CLAPO** gestiscono invece il caso di aggiornamento dei dati già presenti nel sistema di destinazione. Le due verifiche risultano identiche dal punto di vista della condizione di controllo, ma differiscono nella logica successiva, che dipende dal passaggio del processo a cui conducono.

È importante sottolineare che non è stato necessario introdurre un controllo condizionale prima del passaggio **CREATE CLAPO**. Una classificazione può infatti essere creata sia nel caso in cui l'anagrafica corrispondente venga inserita per la prima volta, sia nel caso in cui venga successivamente aggiornata (se precedentemente non era stata creata). Per questo motivo, tale operazione è stata mantenuta come passaggio comune ai due scenari.

Le Figure 4.4 e 4.5 mostrano nel dettaglio le condizioni di controllo dei nuovi passaggi.

4.1.5 Modifiche agli schemi dati

A seguito degli interventi sul processo, è stato necessario apportare alcune modifiche anche agli schemi dati utilizzati dal sistema nelle fasi 3 e 5 (Figura 4.2) per supportare la nuova funzionalità di sincronizzazione.

In particolare, gli schemi interessati sono: **EXP_CLIPOT_CREATE_01.DTMRules** e **EXP_CLIPOT_UPDATE_01.DTMRules**.

La modifica risulta identica per entrambi gli schemi. Specificatamente, è stato necessario aggiornare la query di recupero dei dati affinché sia in grado di gestire due modalità di estrazione: l'estrazione completa di tutte le anagrafiche, prevista

```
if Macro("CmdArg2") <> "" then

   if Macro("CmdArg1") <> "" and Macro("CmdArg1") = "1" then
      return true

   else
      return false

   end if

else
   return true

end if
```

Figura 4.4: Script del passaggio CHECK CREATE

```
if Macro("CmdArg2") <> "" then

   if Macro("CmdArg1") <> "" and Macro("CmdArg1") = "2" then
      return true

   else
      return false

   end if

else
   return true

end if
```

Figura 4.5: Script del passaggio CHECK UP_CLIPO

dal comportamento originale del flusso, e l'estrazione puntuale di una singola anagrafica.

Per mantenere entrambe le modalità all'interno dello stesso flusso di integrazione, è stata definita un'unica query che applica o meno il filtro sull'anagrafica in base al valore del parametro **ARG3**.

In particolare:

- `%(CodiceUnivoco) IS NULL OR %(CodiceUnivoco) = ''` disattiva il filtro nel caso in cui il parametro non sia valorizzato, consentendo l'estrazione di tutte le anagrafiche.
- `T1.CodiceAnagGen = %(CodiceUnivoco)` viene invece applicato quando il parametro è valorizzato, permettendo la selezione di una singola anagrafica.

Le Figure 4.6 e 4.7 mostrano il confronto tra la versione originale e quella modificata dello schema dati *EXP_CLIPOT_CREATE_01*, evidenziando la modifica apportata alla query di estrazione.

```
Select T1.*, T2.CodAttivAtecoFin
from GAVTE_ES_CLIPOT_INS AS [T1] with (NOLOCK)
LEFT OUTER JOIN [%(SisDBNomeEDF)].dbo.STAttAteco2007 AS
[T2] WITH (NOLOCK) ON
(T2.CodAttivAtecofin=T1.CodAteco2007 AND
T2.LivelloAtecofin=5)
where IdIntAmbito = %(IdIntAmbito)
order by CodiceAnagGen asc
```

Figura 4.6: Query originale

4.2 Popolamento della coda lavori tramite servizio REST

La prima modalità prevista per il popolamento della coda lavori consiste nell'utilizzo di un servizio REST. I principali casi d'uso riguardano il mondo web, nel quale un'applicazione o un portale online può utilizzare il servizio per inviare nuove richieste al sistema non appena queste vengono generate. Inoltre, tale meccanismo potrà essere utilizzato in futuro anche per gestire l'importazione di dati da eSOLVER CRM verso eSOLVER.

Attualmente il servizio espone due endpoint principali: uno dedicato alla lettura dello stato di un determinato job, denominato *GetJobStatus*, e uno dedicato alla creazione di un nuovo job, denominato *PostJob*.

```
Select T1.*, T2.CodAttivAtecoFin
from GAVTE_ES_CLIPOT_INS AS [T1] with (NOLOCK)
LEFT OUTER JOIN
[%(SisDBNomeEDF)].dbo.STAttAteco2007 AS [T2] WITH
(NOLOCK) ON
(T2.CodAttivAtecofin=T1.CodAteco2007 AND
T2.LivelloAtecofin=5)
where IdIntAmbito = %(IdIntAmbito) AND (
        '%(CodiceUnivoco)' IS NULL
        OR '%(CodiceUnivoco)' = ''
        OR T1.CodiceAnagGen = '%(CodiceUnivoco)'
    )
```

Figura 4.7: Query modificata

Per quanto riguarda il servizio *GetJobStatus*, esso prevede un unico parametro di richiesta obbligatorio, **IdLavoroDMS**, che rappresenta l'identificativo di un job già presente nella coda lavori. La risposta del servizio restituisce il valore del campo **StatoLavoro**, espresso tramite un numero compreso tra 0 e 8.

Nella tabella 4.2 è riportata la codifica dei possibili valori.

Questo servizio risulta utile quando un sistema esterno necessita di verificare lo stato di avanzamento dell'elaborazione di un job precedentemente inserito nella coda lavori.

Nelle Figure 4.8 e 4.9 vengono mostrati, rispettivamente, un esempio di richiesta e uno di risposta relative a tale endpoint.

Si specifica che, in tutti gli esempi, il DBGruppo di riferimento è **C8**, mentre gli ID delle anagrafiche utilizzate si riferiscono a clienti potenziali creati precedentemente all'interno di eSOLVER, secondo la procedura prevista.

Parametri

Parametro	Valore
value	{ "IdLavoroDMS": 1 }

Figura 4.8: Richiesta dell'endpoint GetJobStatus

Valore	Descrizione	Note
0	Da eseguire	Job mai processato dallo scodatore
1	In esecuzione	Job attualmente in gestione dallo scodatore
2	Eseguito correttamente	Job eseguito dal DMS senza rilasciare errori
3	Eseguito con warning	Job eseguito dal DMS rilasciato con warning
4	Eseguito con errori	Job eseguito dal DMS rilasciando errori
5	Errore di timeout	Job andato in timeout durante l'esecuzione dal DMS
6	Errore tecnico	Anomalia tecnica riscontrata dal DMS in fase di esecuzione del job
7	Da rieseguire	Job in riesecuzione quando il DMS rilascia allo scodatore lo Stato 4 o 5
8	Scaduto	Job scaduto quando in Stato 7 per un numero di giorni superiori a quelli configurati

Tabella 4.2: Possibili valori del parametro StatoLavoro

Figura 4.9: Risposta dell'endpoint GetJobStatus

L'endpoint *PostJob*, invece, consente di aggiungere un nuovo job alla coda lavori. I parametri richiesti sono più numerosi e corrispondono ai campi della tabella *SIDMSCodaLavori* (4.1). Tra questi, i parametri obbligatori sono *TipoOperazione*, che indica la causa scatenante del job, e *TipoEntita*.

La risposta del servizio restituisce l'identificativo del job creato e una variabile denominata **Esito**, il cui valore può essere 1 oppure 0, a seconda che l'operazione sia stata eseguita con successo oppure si sia verificato un errore.

Qui sotto viene mostrato un esempio di richiesta e di risposta relative a questo endpoint.

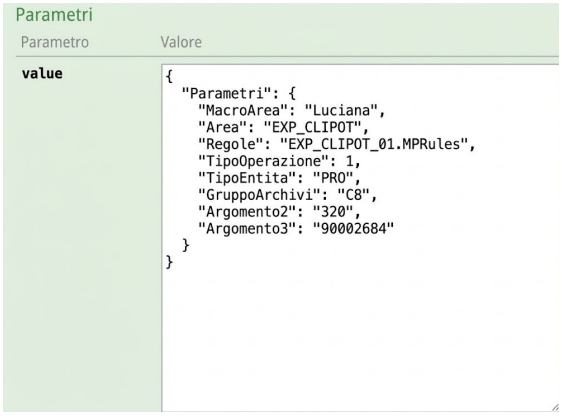


Figura 4.10: Richiesta dell'endpoint PostJob

Figura 4.11: Risposta dell'endpoint PostJob

Accedendo alla pagina **Cruscotto scodatore processi DMS** è possibile monitorare lo stato dei job presenti nella coda lavori. Viene riportato di seguito un esempio della situazione visualizzata all'interno del cruscotto a seguito della chiamata (Figura 4.10) del servizio.

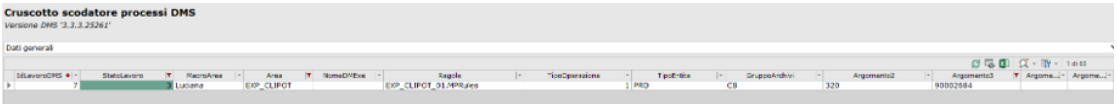


Figura 4.12: Visualizzazione dei job

Ogni job parte da 0, passa a 1 e, auspicabilmente, raggiunge 3.

Gli endpoint descritti erano già presenti nel sistema, ma sono stati utilizzati come casi di studio e di approfondimento al fine di comprendere la logica di funzionamento della coda lavori. Essi hanno inoltre rappresentato un utile punto di partenza per lo sviluppo della soluzione proposta.

4.3 Popolamento della coda lavori tramite trigger

In questa sezione viene analizzata la modalità di popolamento della coda lavori tramite trigger.

Questa soluzione è stata pensata principalmente per chi desidera gestire un'integrazione personalizzata direttamente all'interno del sistema. I trigger risultano particolarmente adatti per la gestione di entità meno comuni e, di conseguenza, meno soggette a frequenti sincronizzazioni.

Tuttavia, l'utilizzo del solo trigger non è sufficiente. Un semplice `INSERT` nella tabella `SIDMSCodaLavori`, infatti, non è in grado di "attivare" la schedulazione del sistema. Come evidenziato nell'analisi precedente, era il servizio REST che, oltre a inserire il record nella tabella `SIDMSCodaLavori`, si occupava anche di avviare la schedulazione, avviando così l'intero processo di elaborazione.

Per questo motivo, affinché il meccanismo funzioni correttamente, è necessario affiancare ai trigger una schedulazione temporizzata attiva quotidianamente, in grado di intercettare i nuovi record inseriti nella coda lavori e avviare il processo.

Per comprendere meglio che tipo di trigger fosse necessario implementare, è stato prima analizzato il comportamento del sistema durante la gestione di un'anagrafica cliente potenziale, osservando le istruzioni SQL generate automaticamente dal programma.

Le schermate di interesse sono riportate di seguito.

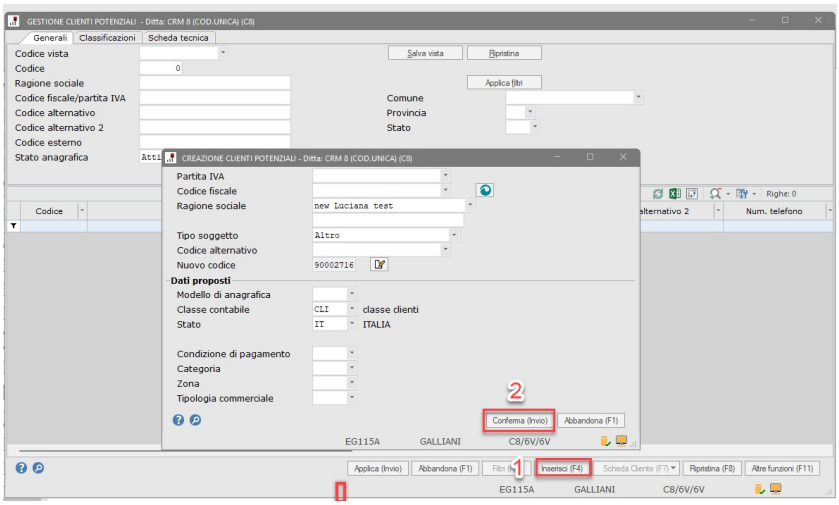


Figura 4.13: Gestione cliente potenziale

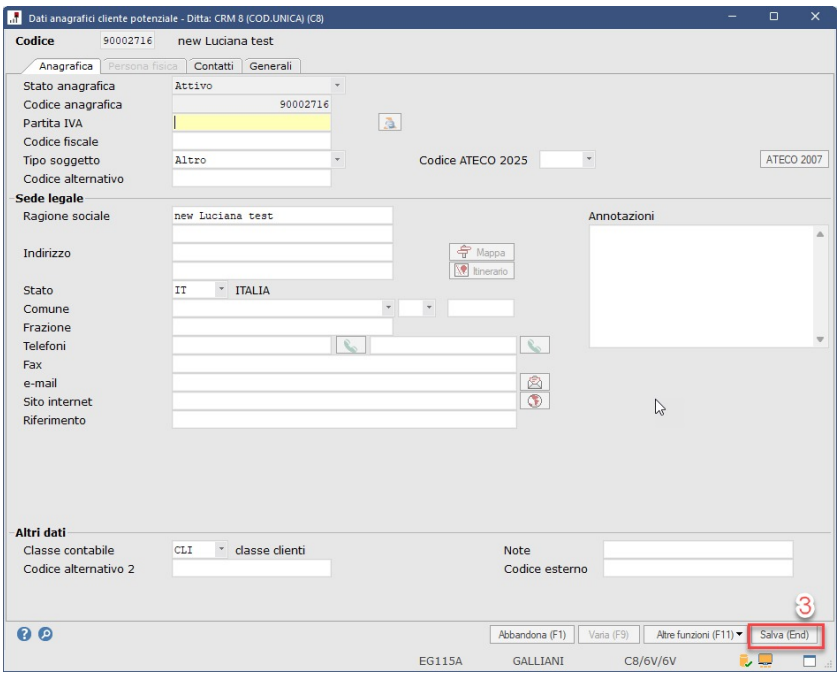


Figura 4.14: Salvataggio cliente potenziale

Quando si clicca sul pulsante *Conferma* (2), viene eseguita un'operazione di INSERT che salva i *dati parziali*.

Durante la creazione di un'anagrafica, infatti, il sistema genera un record provvisorio con lo scopo di "prenotare" il numero identificativo disponibile. Al

momento del salvataggio definitivo (3), l'operazione viene però gestita comunque come una variazione.

Se, durante il flusso di creazione, l'utente seleziona *Abbandona*, viene eseguita un'operazione di `DELETE` per eliminare il record temporaneo. La modifica di un'anagrafica già esistente è invece gestita tramite un'operazione di `UPDATE`, mentre la sua eliminazione comporta l'esecuzione di una `DELETE`.

A partire da questa analisi si è giunti alla conclusione di implementare tre trigger, in grado di coprire tutte le principali operazioni eseguite sulla tabella.

Di seguito è riportato un diagramma di flusso che riassume la logica alla base della progettazione dei trigger.

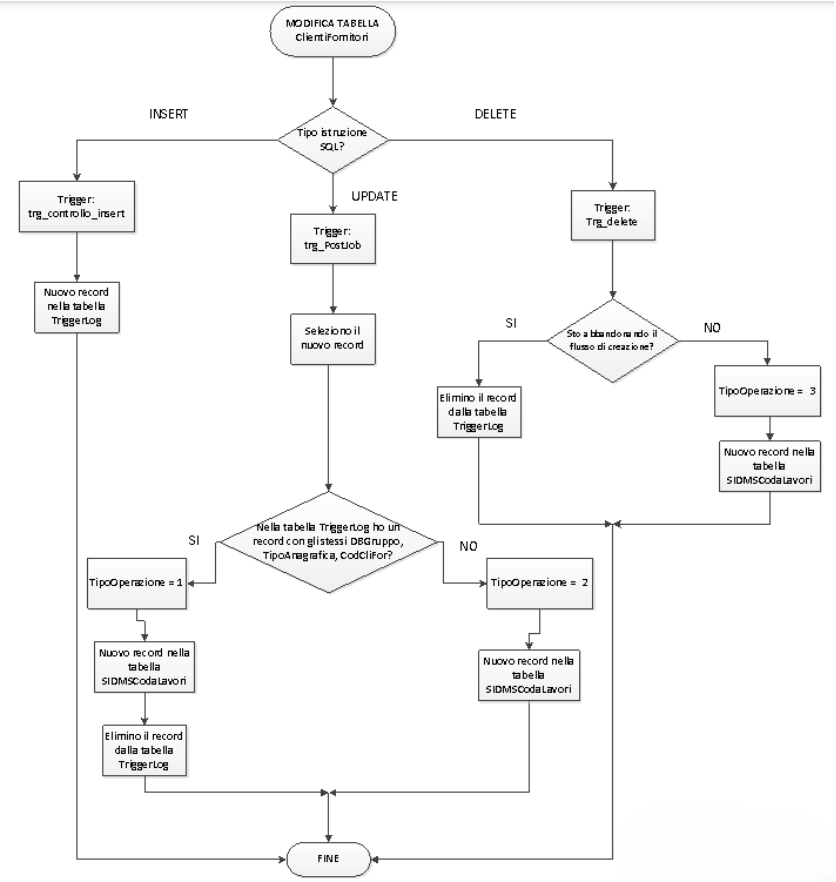


Figura 4.15: Diagramma di flusso trigger

Per gestire correttamente l'operazione di `INSERT` è stata introdotta una tabella di supporto denominata `TriggerLog`. Questa tabella consente di distinguere in modo chiaro tra una reale creazione e un aggiornamento del record, evitando ambiguità nella logica dei trigger.

Dopo un'operazione di inserimento sulla tabella `ClientiFornitori`, viene eseguito il trigger **trg_controllo_insert**, il cui compito è popolare la tabella di supporto. Tale tabella viene successivamente interrogata dal trigger associato all'operazione di `UPDATE`, per determinare se l'operazione in corso corrisponde alla creazione di un nuovo record oppure alla modifica di uno esistente.

La scelta di introdurre una tabella di supporto è stata effettuata con l'obiettivo di rendere la logica il più possibile generalizzabile, favorendone il riutilizzo e mantenendola indipendente dal contesto specifico.

Il trigger **trg_PostJob** viene eseguito dopo un aggiornamento sulla tabella `ClientiFornitori`. Il suo scopo è gestire la sincronizzazione dei dati modificati con i sistemi dipartimentali, distinguendo tra nuove creazioni e aggiornamenti effettivi, e mantenendo aggiornata la tabella di supporto `TriggerLog`.

Un aspetto rilevante riguarda la valorizzazione del campo `TipoOperazione`, che viene determinato in base al contenuto della tabella di supporto: se è presente un record corrispondente a quello appena inserito, l'operazione viene classificata come creazione; in caso contrario, viene interpretata come aggiornamento.

È stata inoltre implementata una stored procedure denominata **sp_InsertSIDMSCodaLavori**, incaricata di inserire il job nella coda lavori. L'utilizzo di una stored procedure al posto di un'istruzione `INSERT` diretta consente di centralizzare la logica applicativa, introdurre eventuali controlli e validazioni, offrire maggiore flessibilità in caso di modifiche future e migliorare complessivamente la sicurezza e la manutenibilità del sistema.

Al termine dell'elaborazione, la tabella di supporto `TriggerLog` viene aggiornata rimuovendo i record temporanei relativi alle operazioni già processate.

Infine, il trigger **trg_delete** viene eseguito dopo un'operazione di eliminazione sulla tabella `ClientiFornitori`. Il suo scopo è aggiornare la tabella di supporto eliminando i record associati a creazioni effettivamente completate e inserire nella coda lavori le informazioni necessarie per sincronizzare la cancellazione con i sistemi dipartimentali.

Il codice completo dei trigger implementati è riportato in Appendice A.

4.4 Popolamento della coda lavori tramite eSOLVER

Per consentire l'attivazione automatica dei flussi di esportazione in seguito alla creazione, modifica o eliminazione di un'anagrafica o di un documento nel gestionale, è stata progettata una soluzione che permette di popolare la coda lavori direttamente da eSOLVER. Questo meccanismo sfrutta le logiche applicative implementate nelle classi BC del sistema per intercettare gli eventi e generare le relative richieste.

L'obiettivo è intercettare gli eventi generati durante le operazioni di data management e trasformarli in richieste di elaborazione da inserire nella coda lavori, che verranno successivamente evase dal sistema di integrazione.

La soluzione è stata progettata per gestire la sincronizzazione delle principali entità, in particolare:

- **Anagrafiche soggetto**: come clienti, fornitori, clienti potenziali e altre anagrafiche;
- **Anagrafiche oggetto**: come articoli, servizi, voci documento e cespiti;
- **Documenti**: ad esempio documenti di Vendite (preventivi e ordini), Acquisti, Contabilità Clienti/Fornitori, Gestione Servizi, Ricevimenti e Spedizioni.

4.4.1 Interfaccia e Tabella di Configurazione Ipotizzate

È stato ipotizzato che il punto di accesso per la gestione di queste regole si troverà nel percorso di menu **Integrazione DMS > Configurazione**, sotto la voce **Sincronizzazioni dati con coda lavori**.

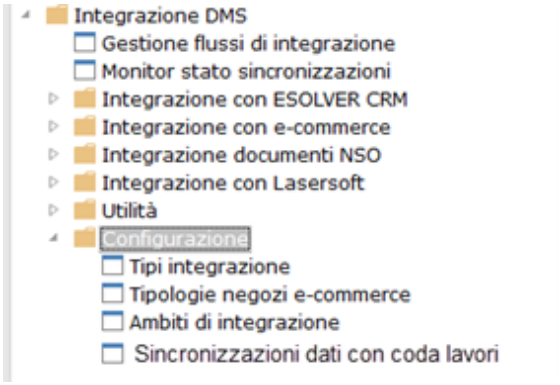


Figura 4.16: Menu Configurazione

Attraverso un'apposita schermata a griglia, in cui sarà possibile inserire del testo in quasi tutte le celle, l'utente potrà definire le regole che stabiliranno quali flussi di integrazione attivare.

La configurazione prevede i seguenti parametri:

- **Entità di prodotto e Tipo di operazione**: gestiti tramite menu a tendina (combo box). Le entità selezionabili saranno Anagrafica soggetto, Anagrafica oggetto e Documento, mentre le operazioni previste saranno Creazione, Variazione ed Eliminazione;

- **ID Processo**: se specificato, permette di valorizzare automaticamente le colonne MacroArea, Area e Processo;

- **Priorità**: Valore numerico (da 0 a 100) che indicherà l'ordine di esecuzione delle configurazioni riguardanti la stessa entità e operazione. Ad esempio, con priorità 1 e 2, il sistema rispetterà rigorosamente l'ordine numerico; il valore 0 indica invece un ordine indifferente;

- **GruppoArchivi**: Campo opzionale. Cliccando sull'apposita freccia, si aprirà una schermata di Ricerca Gruppi per facilitarne la selezione;

- **Attivo**: Casella di controllo che abiliterà o disabiliterà il flusso. Al momento della spunta, si prevede che il sistema mostri prima un messaggio di conferma dell'azione.

Di seguito è riportata un'ipotesi della schermata di configurazione:

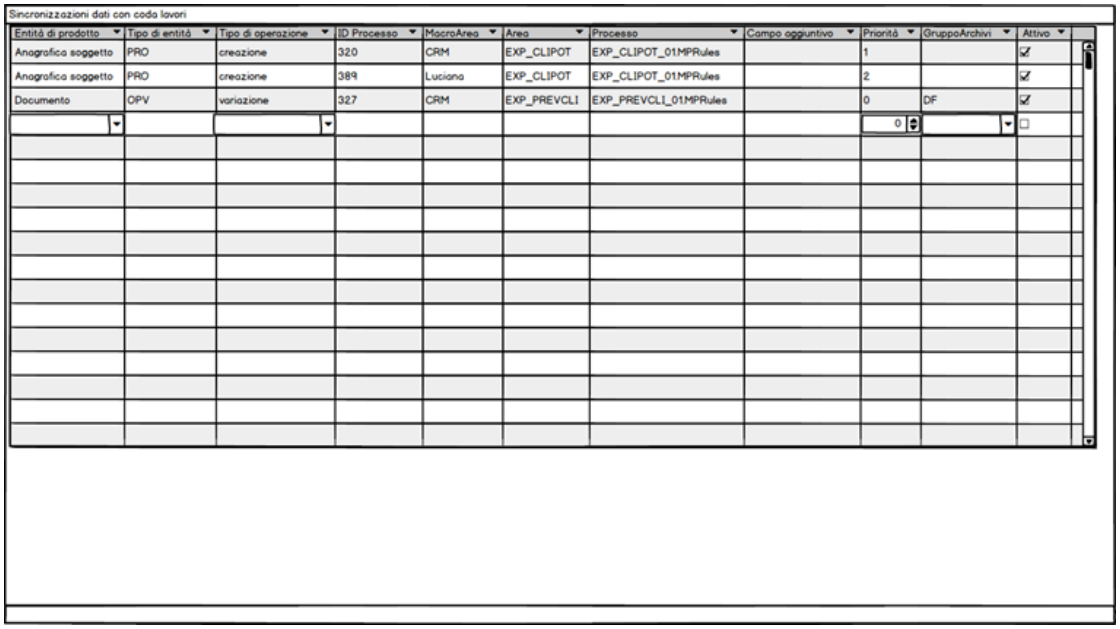


Figura 4.17: Esempio di schermata di configurazione

L'interfaccia riflette la struttura della tabella di configurazione sottostante, i cui campi sono dettagliati nella tabella seguente:

Campo	**Tipo**	**Descrizione**	**Come ricavarlo**	**Obbl**
Codice configurazione	PK, int	Identificativo univoco	Auto	/
Entità di prodotto	int	18 = Anagrafica oggetto 19 = Anagrafica soggetto 32 = Documento	Fisso dalla configurazione	SI
Tipo di entità	varchar(50)	Tipo di entità (es. PRO per i clienti potenziali, OPV per i preventivi)	Fisso dalla configurazione	SI
Tipo di operazione	int	1 = Creazione 2 = Variazione 3 = Eliminazione	Dal programma	SI
ID Processo	varchar(50)	ID del flusso di integrazione	Fisso dalla configurazione	NO
MacroArea	varchar(50)	Macro-area del flusso	Fisso dalla configurazione	SI
Area	varchar(50)	Area funzionale	Fisso dalla configurazione	SI
Processo	varchar(50)	Regole o processo da eseguire	Fisso dalla configurazione	SI
Campo aggiuntivo	varchar(50)	contenuto libero	Fisso dalla configurazione	NO
Priorità	int	valori da 0 a 100 0 = nessuna priorità 1 = priorità alta 2 = priorità media e così via	Fisso dalla configurazione	SI
GruppoArchivi	varchar(2)	DBGruppo	Fisso dalla configurazione	NO
Attivo	bit	1 = attiva 0 = non attiva	Gestito da interfaccia utente	SI

Tabella 4.3: Descrizione campi di configurazione

Note tecniche sulla valorizzazione dei parametri ARG

L'ID Processo non è strettamente obbligatorio, poiché può essere derivato dalla terna MacroArea/Area/Processo; in ogni caso, la sua presenza valorizza il parametro *ARG2*. Il campo "Tipo Entità" (o ID Documento) valorizza *ARG3*, mentre l'ID specifico dell'anagrafica soggetto popola *ARG4*. Infine, il "Campo aggiuntivo" è destinato ad *ARG5*.

Per quanto riguarda il GruppoArchivi, se non viene specificato, la configurazione si applica a qualsiasi DBGruppo che esegue quell'operazione su quel tipo di entità. Se invece viene indicato, la configurazione si applica solo al DBGruppo specificato.

4.4.2 Il Processo di Salvataggio e Intercettazione Atteso

Il funzionamento del meccanismo si baserà sull'intercettazione delle operazioni effettuate dagli utenti all'interno del gestionale. In particolare, quando un utente crea, modifica o elimina un'anagrafica o un documento e conferma l'operazione tramite il comando di salvataggio, il sistema esegue una serie di controlli e operazioni automatiche.

Il primo passaggio consiste nell'**intercettazione dell'evento** applicativo generato dal salvataggio dei dati. Le logiche implementate nelle classi BC permettono di identificare il tipo di operazione effettuata, distinguendo tra inserimento (INSERT), aggiornamento (UPDATE) ed eliminazione (DELETE).

Successivamente, verrà **letta la tabella di configurazione** per verificare la presenza di una configurazione attiva associata all'entità coinvolta e al tipo di operazione eseguita. Se nella configurazione sarà stato indicato un GruppoArchivi, questo dovrà corrispondere al DBGruppo con cui l'operatore avrà effettuato l'accesso e consentirà di limitare l'esecuzione del flusso a determinati contesti applicativi.

Nel caso in cui venga individuata una configurazione valida, il sistema procederà al **recupero dei parametri necessari** e invocherà la classe BC, la quale preparerà il **record del job** e lo inserirà nella tabella SIDMSCodaLavori.

Per validare il modello, sono stati ipotizzati i seguenti scenari di accodamento:

1. **Anagrafiche (Gestione delle Priorità)**: Si supponga che un utente configuri sull'anagrafica Clienti potenziali, per l'operazione di INSERT, l'avvio di due flussi: il primo ("ES CRM - Esp. Clienti Potenziali") con priorità 1 e il secondo ("ES CRM – Esp. Clienti potenziali (Luciana)") con priorità 2. Al salvataggio del cliente, il sistema verificherà le configurazioni attive e, controllando il valore della priorità, invocherà la classe BC per inserire prima il job relativo alla macro area CRM e, successivamente, quello relativo all'area Luciana.

Codice configurazione	Entità di prodotto	Tipo di entità	Tipo di operazione	ID Processo	MacroArea	Area	Regole	Campo aggiuntivo	Priorità	Gruppo Archivi	Attivo
1	19	PRO	1	320	CRM	EXP_CLIPOT	EXP_CLIPOT_01.MPRules		1		1
2	19	PRO	1	389	Luciana	EXP_CLIPOT	EXP_CLIPOT_01.MPRules		2		1

Figura 4.18: Esempio tabella Configurazione per Anagrafiche

2. **Documenti (Filtro per GruppoArchivi)**: Nel caso della modifica (UPDATE) di un documento Preventivi, potrebbe essere impostata una regola per far partire il flusso "ES CRM - Esp. Preventivi clienti", ma limitata al solo gruppo archivi "DF". Al salvataggio, il sistema costruirà la richiesta e invocherà la classe BC per l'inserimento nella coda lavori solo se il DBGruppo ottenuto dal programma corrisponderà a "DF".

Codice configurazione	Entità di prodotto	Tipo di entità	Tipo di operazione	ID Processo	MacroArea	Area	Regole	Campo aggiuntivo	Priorità	Gruppo Archivi	Attivo
1	19	PRO	1	320	CRM	EXP_CLIPOT	EXP_CLIPOT_01.MPRules		1		1
2	19	PRO	1	389	Luciana	EXP_CLIPOT	EXP_CLIPOT_01.MPRules		2		1
3	32	OPV	2	327	CRM	EXP_PREVCLI	EXP_PREVCLI_01.MPRules		0	DF	1

Figura 4.19: Esempio tabella Configurazione per Documenti

4.4.3 Vantaggi Architetturali

Grazie a questo meccanismo, ogni modifica rilevante effettuata all'interno di eSOLVER può generare automaticamente una richiesta di elaborazione nella coda lavori, permettendo al sistema di integrazione di reagire agli eventi applicativi in modo asincrono e configurabile.

Il principale vantaggio di questo approccio è rappresentato dalla separazione tra configurazione e logica applicativa: i flussi da attivare e le relative condizioni possono essere definiti attraverso la tabella di configurazione, senza necessità di intervenire direttamente sul codice applicativo. Questo rende il sistema più flessibile ed estendibile, consentendo di aggiungere nuovi scenari di integrazione semplicemente configurando nuove regole.

Capitolo 5

Conclusioni e sviluppi futuri

5.1 Sintesi del lavoro e risultati

L'integrazione tra sistemi informativi rappresenta oggi un elemento centrale per il corretto funzionamento degli ecosistemi applicativi aziendali, sempre più caratterizzati dalla presenza di software eterogenei che devono collaborare e condividere informazioni in modo tempestivo e affidabile.

In questo contesto si inserisce il lavoro presentato in questo elaborato, sviluppato durante il tirocinio presso Sistemi S.p.A., con riferimento al gestionale ERP eSOLVER e alle sue integrazioni con i sistemi dipartimentali. L'analisi iniziale ha evidenziato come l'attuale meccanismo di integrazione, basato sul Data Migrator Sistemi (DMS) e su un modello di interrogazione periodica tramite polling, rappresenti una soluzione consolidata ma con alcune limitazioni. In particolare, tale approccio introduce una latenza tra la modifica dei dati nell'ERP e la loro propagazione verso i sistemi esterni, oltre a generare un carico costante sul database dovuto alle interrogazioni cicliche.

A partire da queste considerazioni, l'obiettivo principale del lavoro è stato quello di progettare un'evoluzione dell'architettura di integrazione esistente, introducendo un modello più reattivo e orientato agli eventi. È stata quindi definita una nuova architettura in grado di intercettare le modifiche ai dati all'interno dell'ERP e attivare in modo quasi immediato i processi di sincronizzazione verso i sistemi esterni.

Il modello progettato si basa sull'introduzione di una coda lavori transazionale, che rappresenta il punto centrale di raccolta degli eventi generati dalle modifiche ai dati. Tale coda può essere popolata attraverso diverse modalità, tra cui servizi REST, trigger SQL o direttamente dal sistema eSOLVER, consentendo una maggiore flessibilità nella generazione degli eventi. Il DMS assume quindi un ruolo più reattivo, monitorando la coda e attivando i processi di sincronizzazione in risposta

agli eventi registrati.

Il lavoro si è concentrato in particolare sul caso d'uso relativo all'integrazione tra eSOLVER ed eSOLVER CRM, con riferimento ai flussi di esportazione delle anagrafiche dei clienti potenziali. Questo scenario rappresenta un esempio significativo di integrazione tra sistemi aziendali e ha consentito di valutare concretamente l'applicabilità del nuovo modello architetturale.

L'implementazione realizzata ha permesso di dimostrare la fattibilità tecnica della soluzione proposta e di evidenziare i vantaggi derivanti dall'introduzione di un paradigma event-driven all'interno del processo di integrazione.

L'architettura progettata introduce infatti un approccio più dinamico e reattivo rispetto al modello tradizionale basato su polling. La possibilità di intercettare direttamente le modifiche ai dati e di generare eventi che attivano i processi di sincronizzazione consente infatti di ridurre significativamente il tempo che intercorre tra l'aggiornamento delle informazioni nel sistema ERP e la loro disponibilità nei sistemi dipartimentali.

Uno dei principali benefici della soluzione proposta riguarda proprio la riduzione della latenza nella propagazione dei dati. A differenza del modello schedulato, nel quale gli aggiornamenti vengono rilevati solo durante le interrogazioni periodiche del database, l'approccio adottato consente di reagire quasi immediatamente alle modifiche effettuate, migliorando la tempestività complessiva del sistema.

Un ulteriore vantaggio riguarda la riduzione del carico elaborativo sul database dell'ERP. Eliminando la necessità di interrogazioni cicliche in assenza di modifiche, il sistema evita operazioni ridondanti e utilizza le risorse in modo più efficiente.

Dal punto di vista architetturale, la soluzione introduce inoltre un maggiore livello di modularità e flessibilità. La presenza di una coda lavori dedicata alla gestione degli eventi consente infatti di disaccoppiare il sistema ERP dai meccanismi di sincronizzazione, rendendo più semplice l'introduzione di nuove integrazioni o l'estensione della soluzione ad altri sistemi aziendali.

Sebbene la funzionalità sviluppata non fosse ancora stata messa in produzione al termine del periodo di tirocinio, essa è stata inserita tra le novità previste per il rilascio del software programmato per giugno 2026. Questo aspetto rappresenta un ulteriore elemento di validazione del lavoro svolto, evidenziando l'interesse concreto dell'azienda verso l'architettura proposta.

5.2 Limiti della soluzione

Nonostante i risultati ottenuti e i benefici evidenziati, la soluzione proposta presenta alcune limitazioni che devono essere considerate nell'ottica di una futura evoluzione del sistema.

In primo luogo, il lavoro si è concentrato su un caso d'uso specifico, relativo alla sincronizzazione delle anagrafiche dei clienti potenziali tra eSOLVER ed eSOLVER CRM. Sebbene tale scenario rappresenti un esempio significativo di integrazione tra sistemi aziendali, l'applicabilità del modello ad altri flussi informativi richiederà ulteriori attività di analisi e adattamento.

Un secondo aspetto riguarda il fatto che la soluzione non è stata ancora sottoposta a un utilizzo estensivo in ambiente di produzione. Sarà quindi necessario condurre ulteriori test, in particolare in presenza di elevati volumi di dati o di frequenti aggiornamenti, al fine di valutare il comportamento del sistema in scenari operativi reali.

Un ulteriore limite riguarda la gestione dei trigger eventualmente definiti sulle tabelle standard del database. Attualmente tali trigger risultano preservati unicamente in caso di operazioni di *ALTER TABLE*. Al contrario, operazioni che comportino la rigenerazione completa della tabella potrebbero determinare la perdita dei trigger precedentemente definiti, rendendo necessario un loro successivo ripristino.

Infine, l'introduzione di un'architettura orientata agli eventi comporta inevitabilmente un aumento della complessità gestionale del sistema. Aspetti quali il monitoraggio dei processi, la gestione degli errori e il controllo delle code di elaborazione richiederanno strumenti adeguati e una progettazione attenta delle strategie di logging e recupero in caso di anomalie.

5.3 Sviluppi futuri

Alla luce dei risultati ottenuti, la soluzione sviluppata rappresenta un primo passo verso un modello di integrazione basato sulla gestione degli eventi. In prospettiva futura, sono possibili diverse linee di sviluppo che potrebbero contribuire a consolidare e ampliare il modello proposto.

Una prima direzione riguarda l'implementazione del flusso inverso di integrazione, ovvero l'importazione di informazioni da eSOLVER CRM verso eSOLVER. Attualmente la soluzione sviluppata si concentra principalmente sull'esportazione dei dati dall'ERP verso il sistema dipartimentale; l'estensione del modello anche ai flussi in ingresso permetterebbe di realizzare un'integrazione bidirezionale, migliorando ulteriormente la coerenza e l'allineamento delle informazioni tra i diversi sistemi.

Un ulteriore sviluppo potrebbe riguardare l'estensione dell'approccio adottato ad altri flussi informativi gestiti dal sistema ERP. Il modello basato sulla coda lavori potrebbe infatti essere applicato non solo alla sincronizzazione delle anagrafiche dei clienti, ma anche ad altri ambiti funzionali, come la gestione degli ordini, delle fatture o delle informazioni relative agli articoli.

Un evoluzione successiva potrebbe consistere nell'integrazione della soluzione con altri sistemi dipartimentali presenti nell'ecosistema applicativo aziendale, come PEOPLELINK, utilizzato per gestire in un unico ambiente di lavoro i flussi dei processi HR. In questo caso, il modello architetturale potrebbe essere adattato alle specifiche esigenze dei diversi contesti applicativi: se nella soluzione sviluppata la coda lavori può essere popolata attraverso tre modalità differenti (servizio REST, trigger SQL o intervento diretto di eSOLVER), in scenari differenti potrebbe risultare più opportuno adottare una sola di queste modalità, selezionata in base alle caratteristiche del sistema da integrare e ai requisiti operativi.

Dal punto di vista tecnico, sarà inoltre possibile introdurre strumenti avanzati di monitoraggio e osservabilità, al fine di migliorare il controllo dei processi di integrazione e facilitare l'individuazione di eventuali anomalie. L'adozione di meccanismi di gestione degli errori, sistemi di retry automatico e strategie di gestione delle code contribuirà a rendere l'architettura ancora più robusta e affidabile.

Inoltre, ulteriori attività di sperimentazione e valutazione delle prestazioni potranno essere condotte per analizzare il comportamento del sistema in condizioni di carico elevato e per confrontare in modo più approfondito l'efficacia dell'approccio volto agli eventi rispetto al modello tradizionale basato su polling.

In conclusione, il lavoro svolto introduce in azienda un approccio innovativo al tema dell'integrazione tra sistemi, aprendo nuove prospettive per la gestione dei flussi informativi in modo più reattivo ed efficiente. L'adozione di questo modello potrebbe rappresentare un importante passo avanti nell'evoluzione delle soluzioni di integrazione offerte, migliorando le prestazioni dei sistemi e rispondendo in modo più efficace alle esigenze di immediatezza richieste dai clienti.

Appendice A

Implementazione dei trigger

A.1 Trigger trg_controllo_insert

```
1     ALTER TRIGGER [dbo].[trg_controllo_insert]
2         ON [dbo].[ClientiFornitori]
3         AFTER INSERT
4     AS
5     BEGIN
6         SET NOCOUNT ON;
7
8         INSERT INTO TriggerLog (
9               DBGruppo,
10            CodEntitaNum1,
11            CodEntitaNum2,
12            WsInUso,
13            FirmaCreazData,
14            FirmaCreazOra,
15            FirmaUltVarData,
16            FirmaUltVarOra)
17        SELECT i.DBGruppo, i.TipoAnagrafica, i.CodCliFor, i.
    WsInUso, i.FirmaCreazData, i.FirmaCreazOra, i.FirmaUltVarData,
    i.FirmaUltVarOra
18        FROM INSERTED i;
19    END
```

A.2 Trigger trg_PostJob

```
1     ALTER TRIGGER [dbo].[trg_PostJob]
2         ON  [dbo].[ClientiFornitori]
```

```
3         AFTER UPDATE
4     AS
5     BEGIN
6         -- SET NOCOUNT ON added to prevent extra result sets from
7         -- interfering with SELECT statements.
8         SET NOCOUNT ON;
9
10        DECLARE @DBGruppo VARCHAR(2),
11          @Argomento3 VARCHAR(256),
12          @TipoOperazione TINYINT,
13          @DataInserimento DATE,
14          @OraInserimento INT;
15
16        SELECT
17            @DBGruppo = i.DBGruppo,
18            @Argomento3 = i.CodCliFor,
19            @TipoOperazione = CASE WHEN
20                ISNULL(t.CodEntitaNum2, 0) <> 0 THEN 1  -- create
    definitiva
21                                                ELSE 2  -- update
    effettivo
22              END,
23            @DataInserimento = i.FirmaUltVarData,
24            @OraInserimento = i.FirmaUltVarOra
25         FROM INSERTED i
26        INNER JOIN DELETED d ON d.DBGruppo = i.DBGruppo AND
27            d.TipoAnagrafica = i.TipoAnagrafica  AND d.CodCliFor =
     i.CodCliFor
28         LEFT JOIN TriggerLog t  ON t.DBGruppo = i.DBGruppo AND
29            t.CodEntitaNum1 = i.TipoAnagrafica  AND t.
    CodEntitaNum2 = i.CodCliFor
30        WHERE i.WsInUso = '' AND i.WsInUso <> d.WsInUso;
31
32        -- Chiamata alla stored procedure
33       if(ISNULL(@TipoOperazione, 0) >= 1)
34       BEGIN
35         EXEC [dbo].[sp_InsertSIDMSCodaLavori]
36           @MacroArea = 'Luciana',
37           @Area = 'EXP_CLIPOT',
38           @Regole = 'EXP_CLIPOT_01.MPRules',
39           @TipoEntita = 'PRO',
40           @TipoOperazione = @TipoOperazione,
41           @GruppoArchivi = @DBGruppo,
42           @Argomento2 = '320',
43           @Argomento3 = @Argomento3,
44           @CodOperatore = 'ADMIN',
45           @CodWS = 'ZZ',
46           @DataInserimento = @DataInserimento,
47           @OraInserimento = @OraInserimento;
```

```
48      END
49
50      -- la tabella di appoggio viene pulita dopo averla usata
51      DELETE t
52      FROM TriggerLog t
53      INNER JOIN INSERTED i ON t.DBGruppo = i.DBGruppo AND t.
    CodEntitaNum1 = i.TipoAnagrafica AND t.CodEntitaNum2 = i.
    CodCliFor
54      INNER JOIN DELETED d ON i.CodCliFor = d.CodCliFor
55      WHERE i.WsInUso = '' AND i.WsInUso <> d.WsInUso;
56   END
```

A.3 Trigger trg_delete

```
1   ALTER TRIGGER [dbo].[trg_delete]
2       ON  [dbo].[ClientiFornitori]
3       AFTER DELETE
4    AS
5    BEGIN
6      -- SET NOCOUNT ON added to prevent extra result sets from
7      -- interfering with SELECT statements.
8      SET NOCOUNT ON;
9
10     DECLARE @DBGruppo VARCHAR(2),
11         @Argomento3 VARCHAR(256),
12         @DataInserimento DATE,
13         @OraInserimento INT;
14     DELETE t
15     FROM TriggerLog t
16     INNER JOIN DELETED d ON t.DBGruppo = d.DBGruppo AND t.
    CodEntitaNum1 = d.TipoAnagrafica  AND t.CodEntitaNum2 = d.
    CodCliFor
17
18
19     SELECT
20       @DBGruppo = d.DBGruppo,
21       @Argomento3 = d.CodCliFor,
22       @DataInserimento = d.FirmaUltVarData,
23       @OraInserimento = d.FirmaUltVarOra
24     FROM DELETED d;
25
26       -- Chiamata alla stored procedure
27       EXEC [dbo].[sp_InsertSIDMSCodaLavori]
28           @MacroArea = 'Luciana',
29           @Area = 'EXP_CLIPOT',
```

```
30              @Regole = 'EXP_CLIPOT_01.MPRules',
31              @TipoEntita = 'PRO',
32          @TipoOperazione = 3,
33          @GruppoArchivi = @DBGruppo,
34              @Argomento2 = '',           -- id del flusso
35          @Argomento3 = @Argomento3,
36              @CodOperatore = 'ADMIN',
37              @CodWS = 'ZZ',
38              @DataInserimento = @DataInserimento,
39          @OraInserimento = @OraInserimento;
40      END;
```

Bibliografia

[1] IBM. *ETL: Extract, Transform, Load*. 2024. URL: https://www.ibm.com/it-it/think/topics/etl (cit. a p. vii).

[2] Gregor Hohpe e Bobby Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley Professional, 2004 (cit. a p. 5).

[3] Martin Fowler. «What do you mean by "Event-Driven"?» In: *MartinFowler.com* (2017). URL: https://martinfowler.com/articles/201701-event-driven.html (cit. a p. 6).

[4] Adam Bellemare. *Building Event-Driven Microservices: Leveraging Organizational Data at Scale*. O'Reilly Media, 2020 (cit. a p. 7).

[5] Neha Narkhede, Gwen Shapira e Todd Palino. *Kafka: The Definitive Guide: Real-Time Data and Stream Processing at Scale*. O'Reilly Media, 2017 (cit. alle pp. 7, 8).

[6] Debezium Community. *Debezium Architecture Documentation*. Red Hat. 2023. URL: https://debezium.io/documentation/ (cit. alle pp. 7, 8).

[7] Sistemi S.p.A. *eSolver CRM*. 2024. URL: https://www.sistemi.com/software-gestionali/esolver/esolver-crm/ (cit. a p. 22).