



UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

TESI DI LAUREA

I processi ETL e la loro applicazione

RELATORE

Prof. Fabio Palomba

Università degli Studi di Salerno

CANDIDATO

Simone Civale

Matricola: 0512105509

Anno Accademico 2023-2024

Questa tesi è stata realizzata nel

sesa^{lab}
SOFTWARE ENGINEERING
SALERNO

*"Everything negative - pressure, challenges - is all an opportunity for me to rise"-Kobe
Bryant*

Abstract

Nella tesi si analizzano i processi ETL con la loro definizione e i vari tool che permettono di realizzare i processi ETL. La tesi, inoltre, chiarisce il motivo per cui è stato scelto il tool Talend. Infine, viene esposto come si passa da Talend al microservizio e vengono definiti i microservizi stessi. È, inoltre, riportata una valutazione del microservizio.

Indice

Elenco delle Figure	iv
1 Introduzione	1
1.1 Contesto applicativo	1
1.2 Motivazioni e Obiettivi	1
1.3 Risultati ottenuti	2
1.4 Struttura della tesi	2
2 Background e stato dell'arte	3
2.1 ETL	3
2.1.1 A cosa servono i processi ETL	4
2.2 Funzionamento dei processi ETL	4
2.3 Come si è evoluto l'ETL?	6
2.3.1 ETL tradizionale e batch	6
2.3.2 ETL moderno	7
2.3.3 Data warehouse	8
2.4 Vantaggi dell'ETL	8
2.5 Caratteristiche di un ETL	9
2.6 Pipeline ETL	10
2.7 Tool analizzati	10

2.7.1	Talend Open Studio for Data Integration	10
2.7.2	Knime	11
2.7.3	PowerBI	11
2.7.4	Visual Studio con SSIS	12
2.7.5	Esempio di un processo ETL in Talend Open Studio	12
2.7.6	Esempio di processi ETL in Visual Studio con SSIS	19
2.8	Intelligenza artificiale (AI) e machine learning (ML) nell'ETL	27
2.8.1	ETL e democratizzazione dei dati	28
2.8.2	Automatizzare le pipeline ETL	28
2.8.3	Rendere operativi i modelli di AI e ML con l'ETL	28
2.8.4	Replicare il database con l'acquisizione dei dati di modifica (CDC)	28
2.8.5	Maggiore agilità aziendale grazie all'ETL per l'elaborazione dei dati	29
2.9	ETL e passaggio al cloud	29
3	Dal tool al microservizio	30
3.1	Scelta del tool	30
3.2	Che cos'è un microservizio?	30
3.2.1	Caratteristiche dei microservizi	31
3.2.2	Vantaggi dei microservizi	32
3.3	Architettura a microservizi utilizzata	34
3.3.1	Discovery Server	35
3.3.2	API gateway	35
3.3.3	Config server	35
3.3.4	Microservizio ETL	36
3.4	Come estrarre la classe Java da Talend	36
3.5	Realizzazione del microservizio	37
3.5.1	Classe OutputDB	38
3.5.2	Classe ExtractDB	39
3.5.3	Classe Proprietà	40
3.5.4	Classe Controller	40

3.6	JUnit	41
3.7	Docker	42
3.8	Metodologia Agile Scrum	43
4	Analisi dei risultati	46
4.1	Come veniva svolta l'operazione ETL	46
4.2	Valutazione utilità ed efficacia del microservizio ETL	46
4.3	Come si può migliorare il microservizio	47
4.4	Utilizzo del microservizio ETL a livello aziendale	47
4.5	Vantaggi del microservizio ETL	47
4.6	Risultati ottenuti	48
5	Conclusioni	49
	Bibliografia	50

Elenco delle figure

2.1	Funzionamento del processo ETL.	3
2.2	Connessione al database passo 1.	13
2.3	Connessione al database passo 2.	13
2.4	Recupero della tabella passo 1.	14
2.5	Recupero della tabella passo 2.	15
2.6	Creazione job.	16
2.7	Settaggio componente tDBInput.	16
2.8	Settaggio campi tabella.	17
2.9	Settaggio componente tMap.	18
2.10	Settaggio componente tDBOutput.	19
2.11	Componente Attività Esegui SQL.	20
2.12	Configurazione componente Attività Esegui SQL.	21
2.13	Componente Attività Flusso di dati.	22
2.14	Componente Origine ODBC.	22
2.15	Configurazione componente Origine ODBC.	23
2.16	Componente Colonna derivata.	24
2.17	Configurazione componente Colonna derivata.	24
2.18	Componente Destinazione ODBC.	25
2.19	Configurazione componente Destinazione ODBC.	26

2.20	Configurazione componente Destinazione ODBC Mappings.	27
3.1	Architettura Spring.	34
3.2	Esporta classe Java passo 1.	36
3.3	Esporta classe Java passo 2.	37
3.4	Esempio di report junit.	42
3.5	Esempio di board scrum.	45

CAPITOLO 1

Introduzione

1.1 Contesto applicativo

Nell'industria software di oggi, ogni azienda ha i dati disponibili su dei database esterni o su database differenti grazie all'avvento dei database relazionali (precedentemente i dati erano disponibili su fogli elettronici). I dati all'interno dei database relazionali sono rappresentati tramite tabelle. Spesso questi dati sono molto numerosi e sono disponibili su sorgenti diverse; questo rende l'analisi dei dati molto complessa. Per questo motivo sono subentrati i processi ETL (Extract Transform Load). Questi processi permettono di estrarre i dati da diverse sorgenti, trasformarli e caricarli in una sorgente dati. Questi processi permettono un'analisi più facile e veloce in modo tale che i dipendenti di un'azienda possano prendere delle decisioni tempestivamente.

1.2 Motivazioni e Obiettivi

Il motivo per il quale viene svolto questo lavoro è quello di descrivere i processi ETL, vista la loro rapida espansione. I processi ETL sono diventati essenziale dato che i tipi di dati e le loro origini si sono incrementate esponenzialmente (grazie anche

all'affermarsi della tecnologia cloud dove si hanno database di estese dimensioni). L'obiettivo è quello di far conoscere al lettore i processi ETL, i tool disponibili, e la successiva creazione di un microservizio.

1.3 Risultati ottenuti

In questa tesi, è stato prodotto un microservizio tramite Spring microservices per i processi ETL, partendo dalla classe Java generata dal tool Talend Open Studio. Questo microservizio è stato integrato nel progetto btrip dell'azienda EMM, che tratta la gestione delle trasferte dei dipendenti dell'azienda con i relativi costi, per poi effettuare i relativi rimborsi. Realizzare un processo ETL ha permesso di ottenere i dati di cui l'azienda aveva bisogno da un database esterno. Questi dati erano in formati differenti e, quindi, difficili da utilizzare, e presenti in diverse tabelle del database; grazie al processo ETL sono stati accorpati e resi disponibili. Infine, si è realizzata una valutazione tramite intervista a due dipendenti dell'azienda e dalle risposte ottenute si è riscontrato che i processi ETL sono stati utili al fine del progetto, inoltre, avendo creato un microservizio, questo può essere riutilizzato anche per altri progetti.

1.4 Struttura della tesi

La struttura della tesi è suddivisa in quattro capitoli: nel primo capitolo, viene presentato il contesto e la motivazione che ha portato alla creazione del microservizio sui processi ETL; nel secondo capitolo, si introduce il concetto dei processi ETL, esaminando vari tool; nel terzo capitolo, viene presentato il tool scelto tra quelli analizzati e come si è arrivati ad incorporare il tutto nel microservizio; nel quarto capitolo, vengono riportati il design delle interviste e i risultati che si sono ottenuti.

Background e stato dell'arte

2.1 ETL

L'ETL (Extract/Trasform/Load o Estrazione/Trasformazione/Caricamento) è il processo di selezione dei dati da diverse sorgenti e della loro successiva composizione e concentrazione in un solo repository.

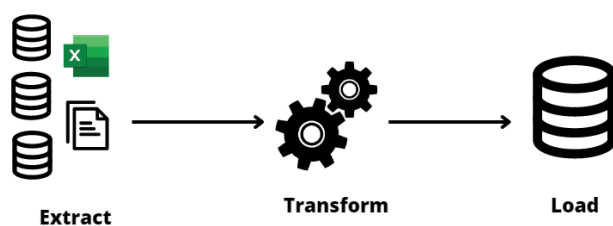


Figura 2.1: Funzionamento del processo ETL.

Il processo ETL estrae i dati da più sorgenti, li trasforma, cambiandoli in base alle necessità del cliente (come illustrato nella figura 2.1) e infine li carica in un data warehouse.

2.1.1 A cosa servono i processi ETL

I motivi per usare i processi ETL sono:

- Assiste le aziende quando bisogna prendere delle decisioni critiche attraverso le analisi dei dati aziendali.
- I processi ETL, a differenza dei database transazionali, hanno la capacità di rispondere alle domande aziendali.
- Il data warehouse mette a disposizione un repository di dati comuni e se i dati presenti nelle sorgenti variano, si aggiorna automaticamente.
- I processi ETL consentono di trasferire i dati da diverse sorgenti in un data warehouse.
- Il processo ETL permette di confrontare i dati tra la sorgente di origine e quella di destinazione.
- L'ETL nel data warehouse permette all'azienda di avere un contesto storico.
- Permette di aumentare la produttività perchè definisce e manipola i dati dalle sorgenti di origine nel database di destinazione.

2.2 Funzionamento dei processi ETL

Il processo ETL è diviso in tre fasi, quali:

1. Estrazione: è la prima fase del processo ETL e consiste nell'estrazione dei dati dal sistema di origine. Questa fase è fondamentale perchè bisogna estrarre i dati in maniera accurata per le fasi successive. Il processo ETL gestisce i dati provenienti da sistemi di origine differenti (come file flat, file xml e altri) e ogni sorgente può impiegare un formato di dati differenti (come database relazionali, XML, JSON e file flat, strutture di database non relazionali). Un'operazione cruciale in questa fase è verificare se i dati ottenuti dalle sorgenti sono validi, altrimenti i dati errati vengono rimandati alla sorgente di origine per un'ulteriore verifica per individuare i record errati. Esistono tre metodi di estrazione

dati: estrazione completa, estrazione fisica ed estrazione incrementale. I dati estratti vengono poi trasferiti nella destinazione (data warehouse).

2. Trasformazione: è la seconda fase del processo ETL e si occupa del procedimento di pulizia e unione dei dati estratti che possono essere impiegati per l'analisi. Questa fase può essere effettuata in due modi:

- (a) trasformazione dei dati in più fasi: questo è il classico processo di estrazione, trasformazione e caricamento. I dati estratti sono trasferiti in un'area di staging dove si effettuano le trasformazioni prima di caricare i dati nel data warehouse.
- (b) trasformazione dei dati all'interno del data warehouse: i dati vengono estratti e caricati direttamente nel data warehouse, dove vengono eseguite le operazioni di trasformazione.

Esistono vari tipi di trasformazione, quali:

- trasformazioni di base, ovvero:
 - pulizia: rimozione dei formati di dati superflui o errati.
 - Risistemazione delle chiavi: esecuzione di relazioni tra chiavi.
- trasformazioni avanzate, ovvero:
 - filtraggio: selezione di alcune righe o colonne in base a criteri specifici.
 - unire: combinazione di dati provenienti da diverse fonti in una sola tabella.
 - dividere: separazione di una colonna in più colonne.
 - convalida dei dati: approvazioni complesse o semplici (ad esempio la riga viene scartata se le prime due colonne sono vuote o nulle per l'elaborazione).
 - aggregazione: gli elementi sono raggruppati da differenti tabelle e database di partenza.
 - integrazione: assegnazione di un nome e una descrizione comuni a ogni elemento di dati univoco (per evitare ambiguità o incongruenze tra nomi e valori diversi).

Una volta finita questa fase i dati sono pronti per l'ultima fase (caricamento).

3. Caricamento: è la terza e ultima fase del processo ETL e consiste nel caricare i dati estratti e trasformati nella sorgente di destinazione. Questa fase può essere usata per: stratificazione dei dati logici, sviluppo di un algoritmo per individuare le duplicazioni e l'esecuzione di un sistema in tempo reale. Ci sono due modi per caricare i dati nel data warehouse, quali:

- (a) caricamento completo: quando una sorgente di dati viene caricata nel database, l'intero dato viene scaricato completamente.
- (b) caricamento incrementale: si scaricano solo i dati che sono cambiati tra la fonte e la destinazione, a intervalli regolari. Si tiene traccia dell'ultima data di estrazione, per caricare solo i dati relativi a quel periodo.

Inoltre, abbiamo anche due tipi di caricamento: caricamento incrementale in streaming e caricamento incrementale in batch [1] [2].

2.3 Come si è evoluto l'ETL?

In epoca pre-cloud, quando le risorse erano scarse e costose, le tre fasi del processo ETL erano eseguite come una pipeline e un malfunzionamento in una fase si ripercuoteva spesso sulle altre. L'ETL tradizionale si basava essenzialmente su due approcci: ETL tradizionale e ETL batch.

2.3.1 ETL tradizionale e batch

Per realizzare l'ETL tradizionale, si utilizzano SQL e script in vari linguaggi, come Python o Hive, per adattarsi al data warehouse di destinazione. Questo metodo garantisce una buona compatibilità e usabilità, ma richiede molto tempo, molta manodopera e può causare errori. Invece, le caratteristiche dell'ETL batch sono: elaborazione dei dati in blocchi, esecuzione regolare, alta latenza, trasformazioni incentrate su disco, originariamente progettata per i database, molteplici copie dei dati dalle sorgenti alla destinazione. Questa è una buona soluzione per un'azienda che non ha fretta di ottenere e analizzare i dati. Tuttavia, poiché questa non è la situazione

più comune al giorno d'oggi, sono stati fatti dei progressi per rendere questi parametri più flessibili. Per rispondere alla necessità di aggregare e processare i dati appena vengono modificati, sono state sviluppate delle soluzioni di integrazione dei dati in tempo reale. Il primo approccio si chiama Change Data Capture (CDC) o anche replica logica, e consiste nel rilevare (di solito tramite i log del database) e trasferire in tempo reale solo i dati modificati, invece di trasferire le snapshot complete dei dati. Questa soluzione offre un trasferimento di dati quasi in tempo reale, con una bassa latenza, ma ha una capacità di trasformazione limitata e supporta principalmente fonti di database. In CDC si possono fare delle trasformazioni semplici e per questo motivo CDC potrebbe essere considerato come una sorta di soluzione ETL. Il metodo ETL sposta la fase di trasformazione dalla piattaforma di integrazione (che ora aggrega e fornisce solo i dati) alla piattaforma dati di destinazione. Così, l'ETL carica i dati direttamente nel sistema host desiderato ed esegue le trasformazioni sul posto. In molti casi reali, "EL" significa in realtà replica dei dati e la sfida è farla in modo valido, sicuro e ad alta accuratezza.

2.3.2 ETL moderno

L'ETL è diventato sempre più popolare per vari motivi: i dati sono generati in quantità sempre maggiori (spesso senza l'intervento umano), il costo dello spazio di archiviazione è sempre più conveniente, sia in locale che nel cloud, il costo del calcolo si è abbassato nel tempo grazie alla varietà di strumenti open source (come Apache Spark, Apache Hadoop, Apache Beam) e alle offerte cloud (come AWS, Microsoft Azure e Google Cloud). Le moderne piattaforme di dati cloud offrono soluzioni a basso costo per analizzare fonti di dati diverse, remote e distribuite in un unico ambiente. Insieme all'integrazione dei dati in tempo reale che permette di trasformare e processare i dati in streaming al volo, i dati possono essere organizzati per l'analisi nel punto stesso in cui arrivano alla destinazione finale. Il passaggio alla tecnologia ETL si è verificato quando le aziende hanno iniziato a migrare dai data warehouse on-prem basati su database relazionali alle implementazioni Map-Reduce, agli ambienti NoSQL e alle piattaforme di streaming dei dati (come Kafka, Apache Flink, Apache Storm), soprattutto quando tutti questi elementi hanno aumentato la

loro presenza nel cloud[3].

2.3.3 Data warehouse

Un data warehouse è una raccolta di dati integrati e coerenti che provengono da diverse sorgenti e che sono memorizzati in una grande base di dati centralizzata. Questi dati sono accessibili ai clienti o agli utenti per supportare le loro decisioni e le loro analisi di business intelligence. Il data warehouse offre dati consolidati e aggregati in una prospettiva multidimensionale ed anche strumenti di analisi online (OLAP). Nella dimensione multidimensionale si applicano varie funzioni come l'estrazione, la pulizia e la trasformazione dei dati. Il caricamento dei dati è la funzione degli strumenti e delle utility del data warehouse. Il suo ambiente comprende un data store, un data mart e i metadati. La funzione principale di un data store è quella di fornire i dati a un data warehouse per scopi di analisi aziendale. Un data mart è una porzione di un data warehouse in cui i dati sono accessibili in modo veloce e con tempi di elaborazione ridotti. I motivi principali per lo sviluppo di un data warehouse sono la ridondanza dei dati e il tipo di utenti finali. L'idea del data warehouse si è sviluppata notevolmente grazie all'integrazione di differenti tecnologie. È importante capire e analizzare le necessità aziendali per progettare un data warehouse efficace ed efficiente. I data warehouse sono largamente impiegati nei seguenti settori: servizi finanziari, servizi bancari, beni di consumo, vendita al dettaglio, produzione controllata [2].

2.4 Vantaggi dell'ETL

I vantaggi dell'ETL sono i seguenti:

- **Dati consolidati:** di frequente le organizzazioni trovano problemi con i dati originari da più sorgenti. I dati di sorgenti diverse possono differenziarsi nel volume, formato e nella complessità. L'ETL li standardizza e provvede ad un'unica visione dei dati. Inoltre, l'ETL permette alle organizzazioni di recuperare e analizzare velocemente i dati. Infine, aiuta a adottare decisioni migliori e più rapide.

- **Contesto storico:** numerose organizzazioni hanno dati storici archiviati in sistemi di archiviazione dati legacy. L'ETL è capace di estrarre i dati dai sistemi legacy e accorparli con quelli correnti. Questo crea un contesto storico mediante il quale le organizzazioni possono distinguere le tendenze a lungo termine. Il contesto storico assiste le aziende a derivare informazioni utili e a sviluppare la business intelligence.
- **Efficienza e produttività:** l'ETL amplia l'efficienza dei team dando loro un semplice accesso ai dati e rimuove l'obbligo di scrivere script personale per il trasferimento dei dati, ampliando così la produttività. Nel momento in cui i dati sono immediatamente disponibili, i dipendenti possono prendere decisioni ben aggiornate e destinare più tempo all'analisi e meno tempo a incarichi di minor valore. [4]

2.5 Caratteristiche di un ETL

Le caratteristiche di un ETL sono:

- **Connettività:** un ETL si collega a tutte le origini di dati già impiegate in azienda. L'ETL è provvisto di connettori inseriti per database, differenti applicazioni di vendita e marketing, differenti formati di file, così da facilitare l'interoperabilità fra sistemi e il cambiamento e la trasformazione dei dati fra l'origine e la destinazione a seconda delle necessità.
- **Interfaccia facile da usare:** un'interfaccia senza bug e semplice da configurare è fondamentale per migliorare tempi e costi di utilizzo e inoltre per facilitare la visualizzazione della pipeline.
- **Gestione degli errori:** l'ETL controlla gli errori in modo efficiente, assicurando la coerenza e l'accuratezza dei dati. Inoltre, mette a disposizione la capacità di trasformazione dei dati fluide e produttive, negando le perdite.
- **Accesso ai dati in tempo reale:** l'ETL recupera i dati in tempo reale, per assicurare aggiornamenti immediati rispetto ai processi correnti.

- Monitoraggio incorporato: l'ETL è fornito di un sistema di monitoraggio integrato che garantisce un'esecuzione fluida del processo. [5]

2.6 Pipeline ETL

L'obiettivo delle pipeline ETL è quello di trasferire (prima di averli estratti ed elaborati) i dati dalle sorgenti dati a un data layer che possa supportare lo sviluppo di applicazioni future. Una pipeline ETL è una particolare data pipeline, ma non vale il contrario; questo dipende dal fatto che le data pipeline possono essere usate per spostare dati da un sistema all'altro senza necessariamente modificarli. Invece, una pipeline ETL prevede sempre tre fasi distinte: estrazione, trasformazione e caricamento dei dati; in questo tipo di pipeline c'è sempre una fase di elaborazione. Un'altra differenza tra i due tipi di pipeline riguarda il modo in cui vengono eseguite: mentre le pipeline ETL di solito processano i dati a intervalli regolari (chiamata esecuzione batch), le data pipeline spesso funzionano come processi real-time per processare gli eventi non appena vengono generati [6].

2.7 Tool analizzati

Per realizzare i processi ETL sono stati analizzati i seguenti tool: Talend, Knime, PowerBI e Visual Studio con l'estensione SQL Server Integration Service (SSIS). Ho analizzato i tool attraverso degli esempi e poi realizzato i processi ETL per i dipendenti, i clienti e i progetti col tool Talend.

2.7.1 Talend Open Studio for Data Integration

È un software open source ed è uno strumento di integrazione dei dati per effettuare operazioni di ETL; questo ci permette di trascinare tramite Drag and Drop vari componenti messi a disposizione. Successivamente verrà generata una classe in linguaggio Java, permettendoci più portabilità tra vari sistemi operativi ma anche l'integrazione della classe in progetti esterni o la modifica della classe per nostri scopi. Con questo tool ho realizzato alcuni esempi per comprendere meglio il

suo funzionamento. Per realizzare i processi ETL sono stati usati principalmente i seguenti componenti:

- **tDBInput**: ci permette di estrarre i dati dal database di input configurandolo con la connessione e la tabella da dove estrarre i dati.
- **tMap**: ci permette di mappare i dati della tabella di input nella tabella di output. Inoltre, ci permette anche di creare variabili o tramite delle funzioni messe a disposizione da Talend oppure si possono creare inserendo il codice Java, per ottenere, ad esempio, la data odierna.
- **tDBOutput**: ci permette di inserire o aggiornare i dati nella tabella di output, configurando la connessione e la tabella di output.

2.7.2 Knime

KNIME è una piattaforma di analisi dei dati open source con una ampia gamma di elementi costitutivi e di strumenti di terze parti (tra cui componenti di machine learning e data mining). È possibile utilizzarla dal caricamento dei dati a un report finale o per predire nuovi valori utilizzando un modello trovato in precedenza. Inoltre, permette anche di eseguire i processi ETL [7]. In questo tool sono stati eseguiti soltanto esempi per capire meglio il funzionamento del tool. Anche questo tool, come il tool Talend, mette a disposizione dei componenti che poi vanno configurati opportunamente. Negli esempi sono stati usati componenti per leggere e poi caricare i dati sia con file che con database, componenti per filtrare colonne e righe e per rappresentare i dati in grafici a barre, a torte, ecc, componenti per convertire le stringhe, per ordinare, raggruppare e per unire più dati.

2.7.3 PowerBI

Power BI è una soluzione di Microsoft che permette di creare e condividere report e dashboard interattivi per l'analisi dei dati aziendali. Power BI può connettersi a diverse fonti di dati, come fogli di calcolo, file di testo, database, ecc., e trasformarli in visualizzazioni dinamiche e personalizzabili. Power BI aiuta gli utenti a scoprire informazioni utili ed a prendere decisioni basate sui dati [8]. Questo tool non è

stato preso in considerazione perché analizzandolo si è visto che la connessione ai database è possibile solo attraverso SQL server e avendo il database in PostgreSQL questo risultava un po' difficile. Inoltre, questo tool ci permette di rappresentare i dati attraverso i grafici rispetto agli altri tool; questo perché è basato sull'analisi dei dati piuttosto che sui processi ETL. Infatti, in questo tool ho usato come componente solo una scheda con due campi e poi ho analizzato la connessione tramite SQL Server con PostgreSQL.

2.7.4 Visual Studio con SSIS

In questo tool si usa l'estensione SQL Server Integration Service (SSIS) che è un'evoluzione di Data Transformation Services (DTS), già presente in SQL Server 2000. Si tratta di un potente strumento di estrazione, trasformazione e caricamento (ETL), che permette di trasferire, modificare e pulire i dati prima di inserirli in un database transazionale, in un database di staging o in qualsiasi altra destinazione [9]. In questo tool ho creato degli esempi per capire meglio il suo funzionamento. I componenti utilizzati sono:

- Attività Esegui SQL: è il componente per configurare la connessione di input.
- Attività Flusso Dati: è il componente per il flusso dati dove si possono estrarre i dati e poi caricarli nel database attraverso i componenti Origine ODBC e Destinazione ODBC.

Di seguito vediamo degli esempi su Talend Open Studio for Data Integration e su Visual Studio con SSIS.

2.7.5 Esempio di un processo ETL in Talend Open Studio

Il software Talend Open Studio è stato utilizzato per realizzare le operazioni di ETL e produrre la lista delle commesse, la lista dei dipendenti e la lista dei clienti. La prima cosa da fare è creare la connessione al database tramite il pannello a sinistra e andando in "Metadata" e poi in "Db Connections", facendo click col tasto destro, si seleziona "Crea connessione" e si aprirà una finestra di questo tipo (vedi immagine 2.2):

Connessione Database

Aggiorna connessione database - Passo 1/2

Aggiorna proprietà

Nome: Zucchetti

Scopo: Connessione al database zucchetti

Descrizione: Connessione al database zucchetti

Autore: user@talend.com

Bloccatore: user@talend.com

Versione: 0.1

Stato:

Percorso:

< Back Next > Finish Cancel

Figura 2.2: Connessione al database passo 1.

In questa schermata (immagine 2.2) si può specificare il nome, la descrizione e lo scopo della connessione. Fatto ciò, si clicca su "Next" e si avrà la seguente schermata (vedi immagine 2.3):

Connessione Database

Aggiorna connessione database - Passo 2/2

Devi premere il bottone verifica per verificare l'impostazione del database

Tipo DB: PostgreSQL

Versione Db: v9 and later

Stringa di connessione: jdbc:postgresql://localhost:2222/zucchetti?

Login: pgadmin

Password:

Server: localhost

Porta: 2222

DataBase: zucchetti

Schema: public

Parametri aggiuntivi:

Test connection

Esportare come contesto Ripristina contesto

[How to install a driver](#)

< Back Next > Finish Cancel

Figura 2.3: Connessione al database passo 2.

In questa schermata (immagine 2.3) si può specificare il tipo di database (nel nostro caso PostgreSQL), la sua versione, i vari campi per l'autenticazione, il database e

lo schema del database. Una volta impostati i vari parametri, si può verificare che la connessione funzioni facendo click su "Test connection". Dopo aver creato la connessione bisogna recuperare le tabelle che ci serviranno per produrre le liste delle commesse e dei clienti; per fare questo si va nel pannello a sinistra e sulla connessione creata, facendo click col tasto destro si preme "Recupera schema" e si aprirà la seguente finestra (vedi immagine 2.4):

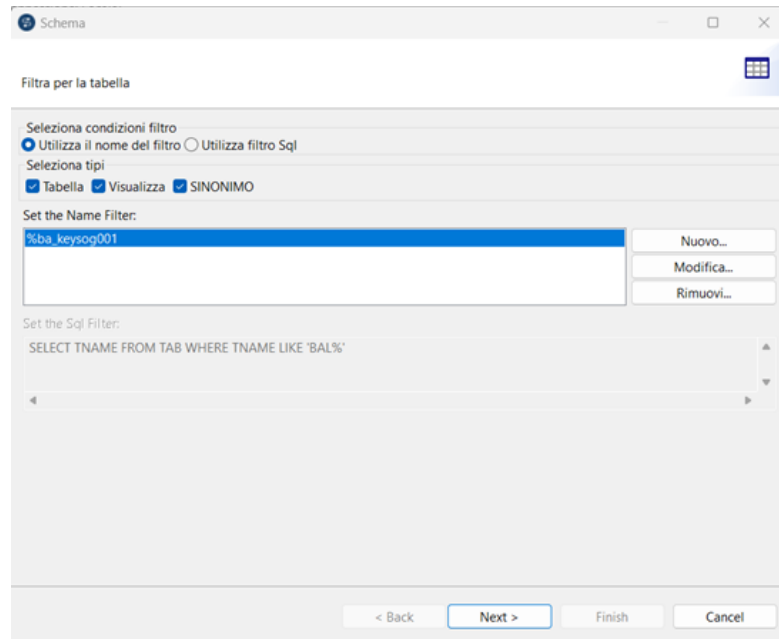


Figura 2.4: Recupero della tabella passo 1.

In questa finestra (immagine 2.4) si specifica il nome della tabella che serve, facendo click su "Next" e si ha la seguente schermata (vedi immagine 2.5):

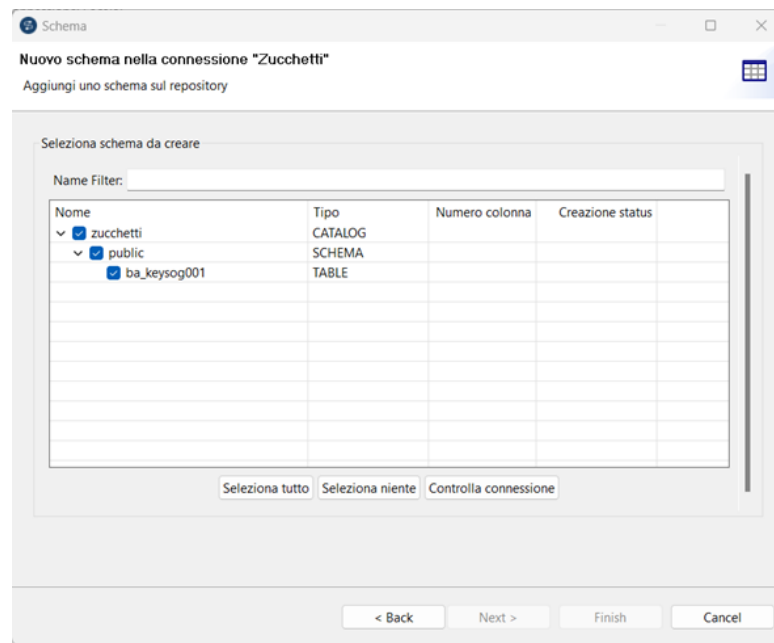


Figura 2.5: Recupero della tabella passo 2.

Questa finestra (immagine 2.5) riporta le tabelle con il nome specificato precedentemente e qui si selezionano le tabelle e si preme su "Finish". La stessa cosa va fatta per la connessione al database di destinazione, per cui si crea un'altra connessione e, dopo questo, si selezionano le tabelle di destinazione necessarie. Prima di creare le varie liste, si analizzano i componenti che si utilizzeranno. Per i nostri esempi useremo i componenti: tDBInput, tMap, tDBOutput.

Lista clienti

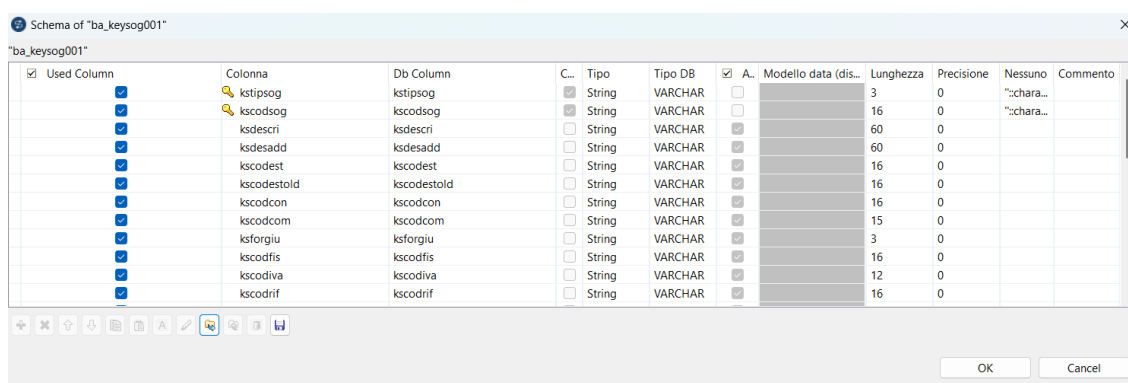
Innanzitutto bisogna creare il job e per fare ciò si crea prima una cartella, dove si inseriscono tutti i job, chiamata Zucchetti, facendo click col tasto destro sulla cartella si seleziona "Creare job" e si aprirà la seguente finestra (vedi immagine 2.6):

Figura 2.6: Creazione job.

In questa finestra (immagine 2.6) si specifica il nome, lo scopo e la descrizione del job. Una volta creato il job, premendoci sopra due volte si aprirà lo spazio di lavoro, dove si possono trascinare i vari componenti presenti nel pannello a destra. Il primo componente che si inserisce è tDBInput, trascinato dal pannello a destra; dopodiché, cliccandoci sopra, nel pannello "Componente" sottostante, si setta la connessione al database, in particolare la tabella da dove si devono estrarre i dati. Si ottiene la seguente schermata (vedi immagine 2.7):

Figura 2.7: Settaggio componente tDBInput.

In questa schermata (immagine 2.7) in "Database" si seleziona il tipo di database (nel mio caso seleziono PostgreSQL), in "Tipo proprietà" si seleziona "Repository" e con i tre punti presenti a fianco si seleziona la connessione creata all'inizio con il database Zucchetti. Successivamente i campi dell'autenticazione verranno compilati automaticamente perché vengono presi dai valori che abbiamo specificato in fase di creazione della connessione. Per lo schema si seleziona "Repository" e, con i tre punti presenti a fianco, si seleziona la tabella che serve; cliccando i tre punti a fianco di "Edit schema" si può verificare se la tabella è stata caricata o meno, dato che si apre una finestra e, cliccando "View schema", si osserva un'ulteriore finestra che fa vedere tutti i campi della tabella caricata, il tipo di dati, la sua lunghezza e altri parametri, come si può vedere di seguito (vedi immagine 2.8):



The screenshot shows a window titled "Schema of 'ba_keysg001'" with a sub-header "ba_keysg001". It contains a table with columns: "Used Column", "Colonna", "Db Column", "C...", "Tipo", "Tipo DB", "A.", "Modello data (dis...", "Lunghezza", "Precisione", "Nessuno", and "Commento". The table lists various columns and their properties, including data types like VARCHAR and lengths like 3, 16, 60, 15, 3, 12, and 16.

Used Column	Colonna	Db Column	C...	Tipo	Tipo DB	A.	Modello data (dis...	Lunghezza	Precisione	Nessuno	Commento
<input checked="" type="checkbox"/>	kstipsog	kstipsog	<input checked="" type="checkbox"/>	String	VARCHAR	<input checked="" type="checkbox"/>		3	0	""chara...	
<input checked="" type="checkbox"/>	kscodsog	kscodsog	<input checked="" type="checkbox"/>	String	VARCHAR	<input checked="" type="checkbox"/>		16	0	""chara...	
<input checked="" type="checkbox"/>	ksdescri	ksdescri	<input checked="" type="checkbox"/>	String	VARCHAR	<input checked="" type="checkbox"/>		60	0		
<input checked="" type="checkbox"/>	ksdesadd	ksdesadd	<input checked="" type="checkbox"/>	String	VARCHAR	<input checked="" type="checkbox"/>		60	0		
<input checked="" type="checkbox"/>	kscodest	kscodest	<input checked="" type="checkbox"/>	String	VARCHAR	<input checked="" type="checkbox"/>		16	0		
<input checked="" type="checkbox"/>	kscodestold	kscodestold	<input checked="" type="checkbox"/>	String	VARCHAR	<input checked="" type="checkbox"/>		16	0		
<input checked="" type="checkbox"/>	kscodcon	kscodcon	<input checked="" type="checkbox"/>	String	VARCHAR	<input checked="" type="checkbox"/>		16	0		
<input checked="" type="checkbox"/>	kscodcom	kscodcom	<input checked="" type="checkbox"/>	String	VARCHAR	<input checked="" type="checkbox"/>		15	0		
<input checked="" type="checkbox"/>	ksforgiu	ksforgiu	<input checked="" type="checkbox"/>	String	VARCHAR	<input checked="" type="checkbox"/>		3	0		
<input checked="" type="checkbox"/>	kscodfis	kscodfis	<input checked="" type="checkbox"/>	String	VARCHAR	<input checked="" type="checkbox"/>		16	0		
<input checked="" type="checkbox"/>	kscodiva	kscodiva	<input checked="" type="checkbox"/>	String	VARCHAR	<input checked="" type="checkbox"/>		12	0		
<input checked="" type="checkbox"/>	kscodrif	kscodrif	<input checked="" type="checkbox"/>	String	VARCHAR	<input checked="" type="checkbox"/>		16	0		

Figura 2.8: Settaggio campi tabella.

Il parametro "Nome Tabella" verrà compilato automaticamente. In ultimo si specifica la query nel parametro "Query" e così facendo si configura il componente tDBInput per estrarre i dati dal database zucchetti. Successivamente si inserisce il componente "tMap" sempre trascinandolo per mappare i dati estratti dal database zucchetti, per caricarli poi nel database di destinazione. Per configurarlo si fa doppio click sul componente e si ha la seguente schermata (vedi immagine 2.9):

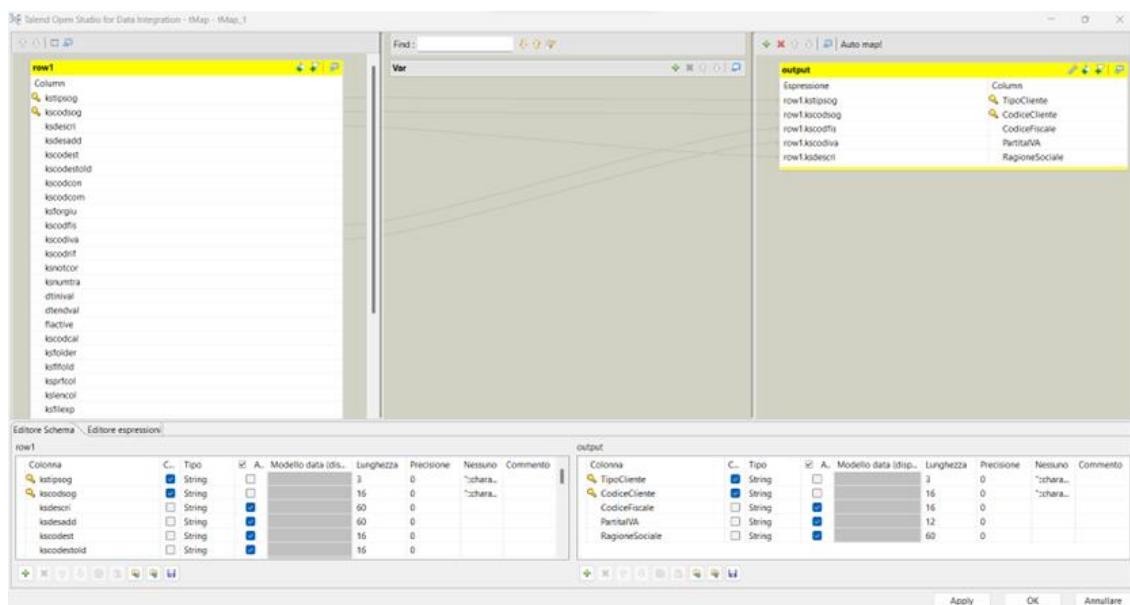


Figura 2.9: Settaggio componente tMap.

In questa schermata (immagine 2.9) si ha la tabella di input a sinistra (è la tabella ottenuta dal componente tDBInput), e a destra si ha la tabella di output. Al centro si possono creare delle variabili e si specifica una funzione come, ad esempio, la data odierna premendo il "+" e aggiungendo la relativa funzione. Per mappare le colonne della tabella di input nella tabella di output bisogna trascinarle. Nel nostro caso si trascina: "kstipsog" che rappresenta il tipo di cliente, "kscodsog" che rappresenta il codice cliente, "ksdescri" che rappresenta la ragione sociale, "kscodfis" che rappresenta il codice fiscale e "kscodiva" che rappresenta la partita IVA. Nella tabella di output nella sezione "Colonna", cliccandoci sopra, si può rinominarla; inoltre, si può anche specificare il tipo della colonna, la sua lunghezza e altri parametri come si può vedere dall'immagine. Successivamente si inserisce il componente "tDBOutput", trascinandolo e configurandolo, come fatto per il componente "tDBInput", premendoci sopra e andando nella scheda "Componente" si ottiene la seguente schermata (vedi immagine 2.10):

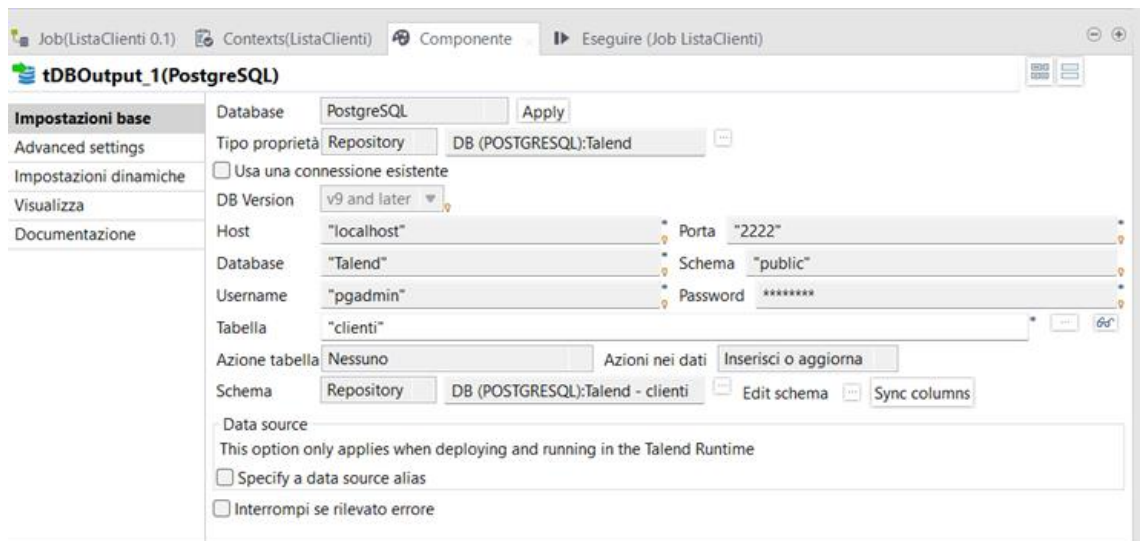


Figura 2.10: Settaggio componente tDBOutput.

Questo componente è simile al componente "tDBInput" e, infatti, si ha il database, la specifica del database, l'autenticazione presa dalla connessione creata all'inizio e la tabella specificata. La differenza tra i due componenti è che nel caso di tDBOutput vi è "l'azione tabella" come, ad esempio, "crea la tabella" (se esiste) ecc.; inoltre, si ha anche "l'azione nei dati" come, ad esempio, "Inserisci o aggiorna" (come nel mio caso). Nel campo "Schema", si specifica la tabella di destinazione dove si inseriscono i dati estratti col componente "tDBInput". Così facendo, si è configurato il componente "tDBOutput". Dopodiché, andando nella scheda sottostante "Eseguire", premendo su "Esegui", si esegue il job e si ottengono nella tabella del database di destinazione i dati estratti dal database zucchetti.

2.7.6 Esempio di processi ETL in Visual Studio con SSIS

Tramite il software Visual Studio si esamina un caso d'uso di un processo ETL, usando l'estensione SQL server integration services projects. Questa estensione è utile perché comprende le operazioni dell'ETL per il data warehouse. In questo caso d'uso si vedrà come prendere il codice cliente dalla relativa tabella presente nel database "zucchetti" e si inserirà questo codice in una tabella destinazione presente in un altro database. La prima cosa da fare è installare il componente per creare il progetto per

far funzionare i processi ETL e l'estensione che contiene gli strumenti ETL. Quindi, una volta aperto "visual studio" si clicca in "Strumenti" e in "Ottieni strumenti e funzionalità" e si aprirà una schermata dove si possono selezionare vari componenti; si selezionerà poi "Elaborazione ed archiviazione dati", che verrà installato facendo click su "Installa". Questo cosa servirà per creare il progetto che ci permette di usare gli strumenti ETL. Successivamente bisogna installare l'estensione "SQL server integration services projects": si clicca su "Estensioni", poi su "Gestisci le estensioni" e su "Visual Studio Marketplace", si seleziona l'estensione e si installa facendo click su "Installa". Così facendo, si è installato tutto quello che serviva; ora si può creare una pipeline ETL SSIS (SQL Server Integration Services): si crea un nuovo progetto scegliendo come modello "Integration Services Project". Una volta creato il progetto nella scheda "Control Flow", si inserisce, tramite "Drag and Drop", il componente "Attività Esegui SQL" presente nel pannello a sinistra; questo componente esegue un comando "SQL" e verrà usato per pulire la tabella di destinazione in modo da non avere valori duplicati (vedi immagine 2.11).

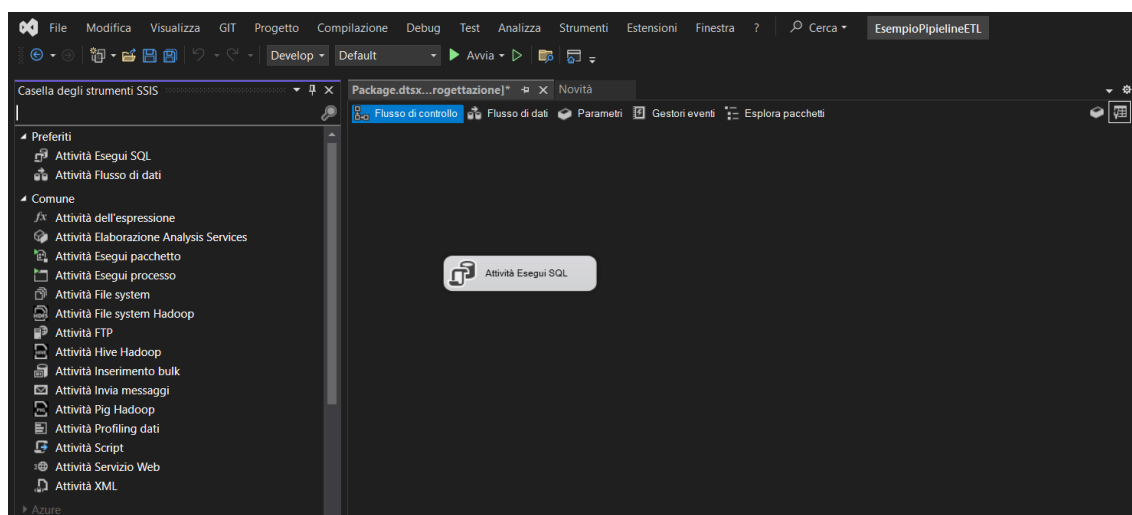


Figura 2.11: Componente Attività Esegui SQL.

Adesso bisogna configurare questo componente e per fare questo si fa doppio click sul componente e si aprirà la schermata seguente (vedi immagine 2.12):

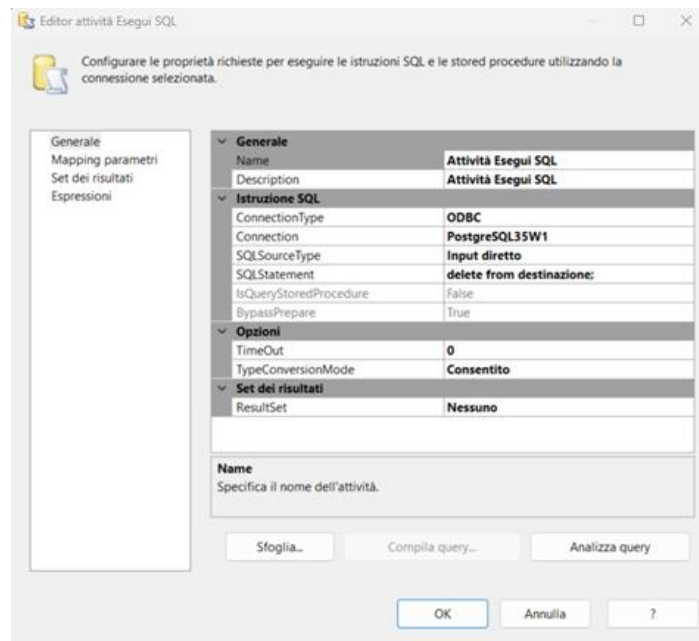


Figura 2.12: Configurazione componente Attività Esegui SQL.

In questa finestra (immagine 2.12) si hanno vari parametri tra cui: "ConnectionType" che indica il tipo di connessione (nel mio caso seleziono ODBC), "Connection" che indica la connessione e si seleziona la connessione alla tabella del database di destinazione, "SQLStatement" che indica il comando SQL e si compila con la query necessaria. Fatto ciò, si preme su "Ok" e si è così configurato il componente. Adesso si trascina sempre tramite "Drag and Drop", dal pannello a sinistra, il componente "Attività flusso di dati"; questo componente trasferisce i dati tra l'origine e la destinazione (vedi immagine 2.13).

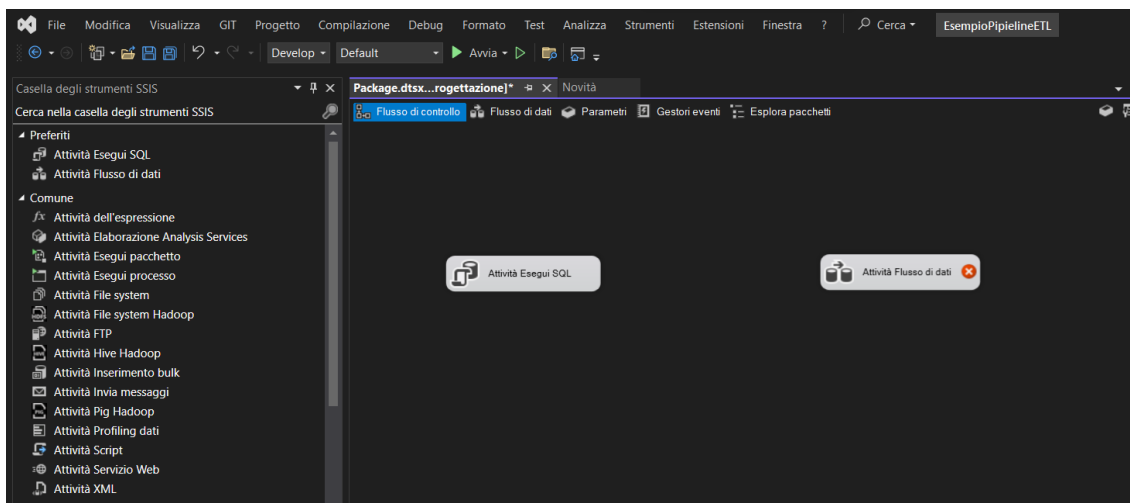


Figura 2.13: Componente Attività Flusso di dati.

Prima di configurare il componente "Attività Flusso di dati", si collegano i due componenti, andando sul componente "Attività Esegui SQL" e trascinando la freccia sul componente "Attività Flusso di dati". Dopo questa operazione, si può configurare il componente "Attività Flusso di dati": si fa doppio click su questo componente e si va nella scheda "Data Flow". In questa scheda si inserisce il componente "Origine ODBC" dal pannello a sinistra; questo componente ci permette, data una connessione al database e la relativa tabella, di estrarre i dati necessari (vedi immagine 2.14).

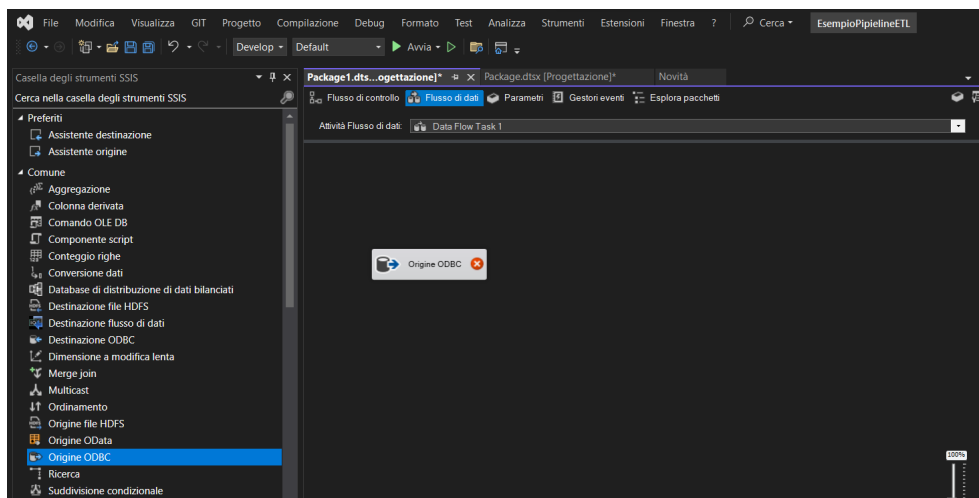


Figura 2.14: Componente Origine ODBC.

Una volta inserito si può rinominare facendo click col tasto destro sul componente e premendo "Rinomina". Successivamente bisogna configurare il componente con la connessione e la tabella per estrarre i dati e per fare ciò si fa doppio click sul componente e si aprirà la seguente finestra (vedi immagine 2.15):

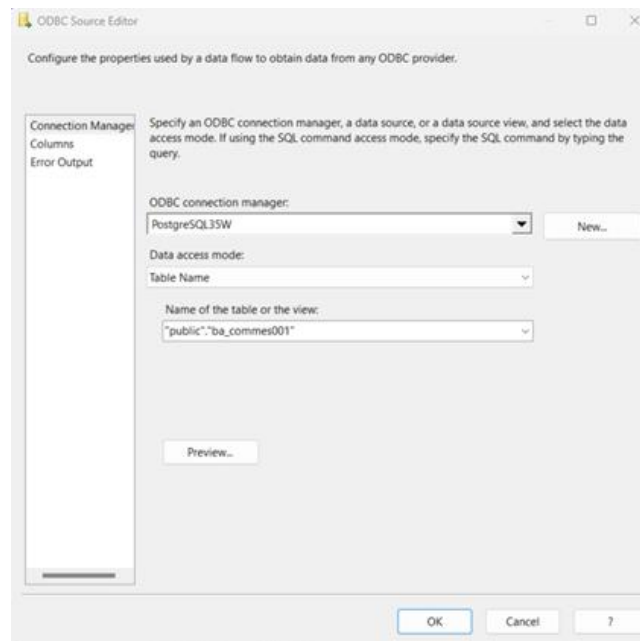


Figura 2.15: Configurazione componente Origine ODBC.

In "ODBC connection manager" si specifica il manager di connessione; cliccando "Data access mode", o tramite la tabella (opzione che seleziono) o tramite un comando SQL, e andando successivamente in "Name of the table or the view", si specifica la tabella da dove si vogliono estrarre i dati e, tramite il tasto "Preview", si potrà vedere l'anteprima della tabella con i relativi campi e dati. In questo modo sono estratti i dati dalla tabella di origine. In seguito, sempre tramite "Drag and Drop", dal pannello a sinistra si inserisce il componente "Colonna derivata"; questo componente permette di selezionare le varie colonne dalla tabella che servono e dà la possibilità di applicare delle funzioni (vedi immagine 2.16).

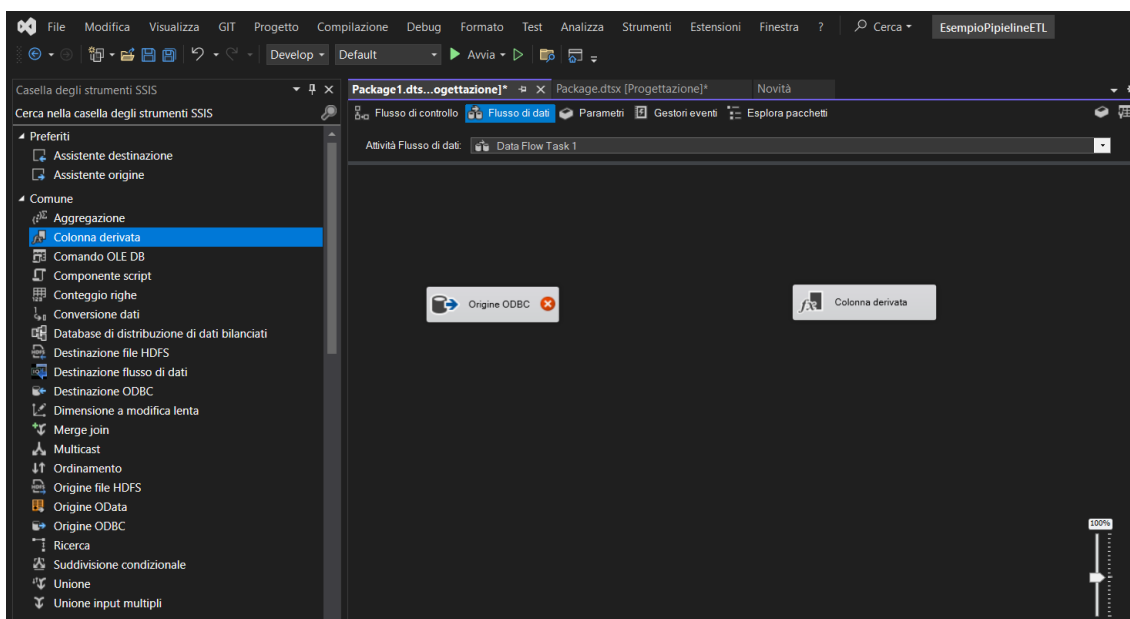


Figura 2.16: Componente Colonna derivata.

Ora bisogna collegare il componente "Origine ODBC" al componente "Colonna derivata" e per fare ciò si va sul componente "Origine ODBC" e si trascina la freccia blu al componente "Colonna derivata". Successivamente si inseriscono le colonne, estratte dalla tabella con il componente "Origine ODBC", che servono: si fa doppio click sul componente "Colonna derivata" e si aprirà la seguente schermata (vedi immagine 2.17):

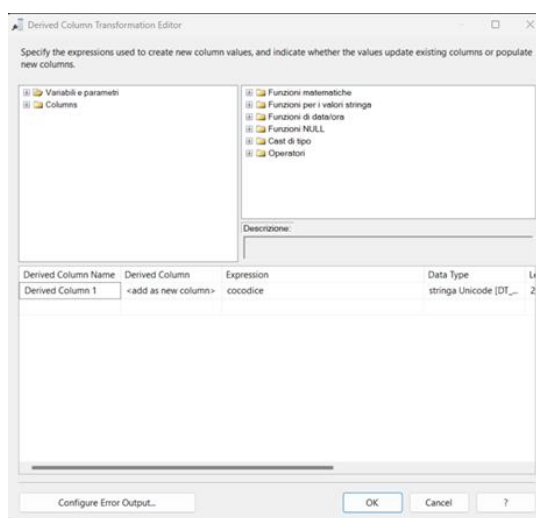


Figura 2.17: Configurazione componente Colonna derivata.

In "Columns" si hanno tutte le colonne della tabella, e trascinandole in basso si

possono inserire le colonne necessarie; nel mio caso inserisco "cocodice" che rappresenta il codice del cliente. Inoltre, si potrebbe configurare una colonna anche con le funzioni disponibili nel riquadro a destra. In ultimo si aggiunge il componente (sempre tramite "Drag and Drop") "Destinazione ODBC" presente nel pannello a sinistra; questo componente carica i dati in una tabella del database indicata (vedi immagine 2.18).

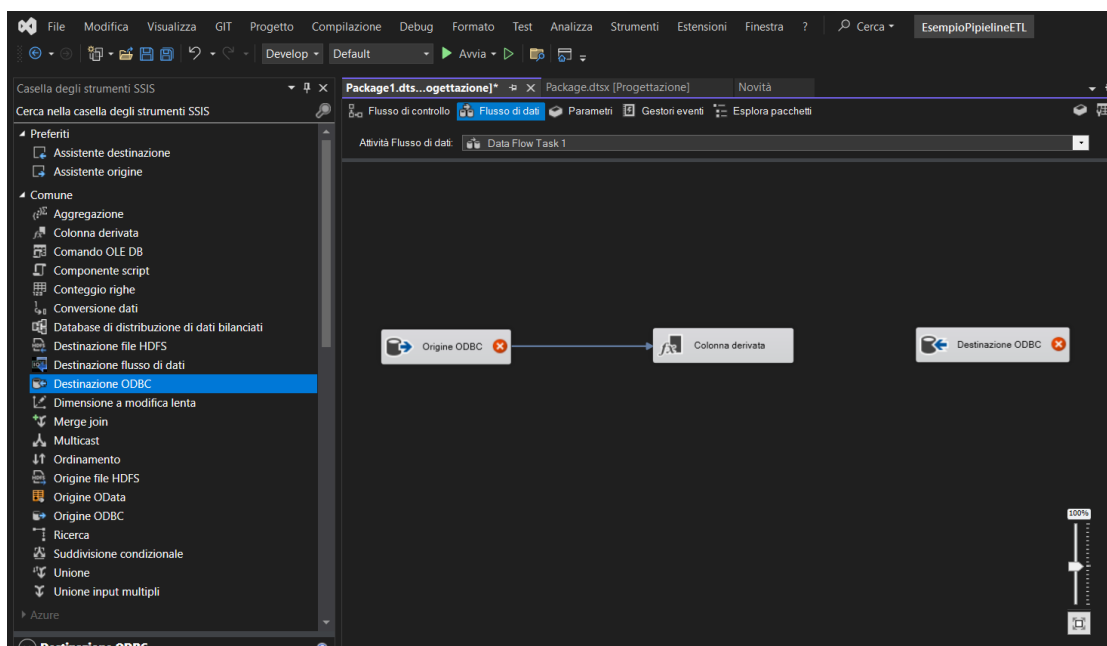


Figura 2.18: Componente Destinazione ODBC.

Ora bisogna configurare il componente "Destinazione ODBC" andando ad indicare il database e la tabella: si fa doppio click sul componente e si aprirà la seguente finestra (vedi immagine 2.19):

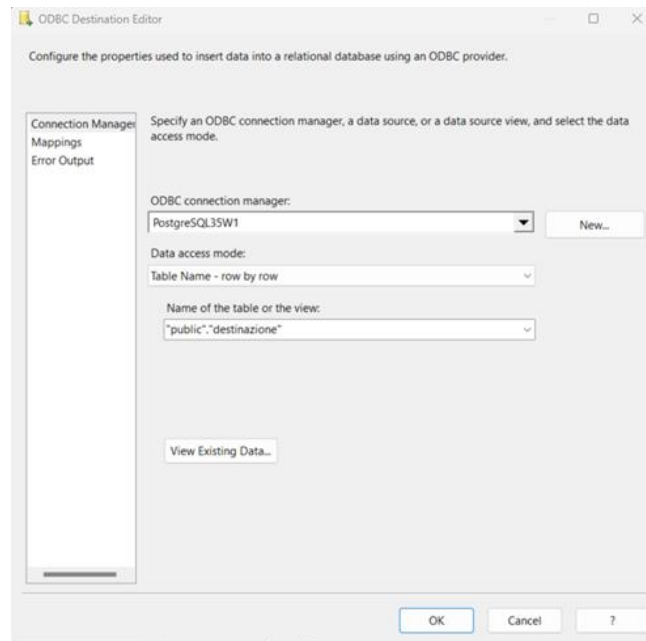


Figura 2.19: Configurazione componente Destinazione ODBC.

Questa schermata è molto simile alla schermata di configurazione del componente "Origine ODBC". In "ODBC connection manager" si ha il manager di connessione di destinazione; in "Data access mode", selezionando "tabella riga per riga" (oppure tramite tabella ma con operazioni batch) e poi "Name of the table or the view", si clicca la tabella di destinazione. Successivamente, attraverso il pulsante "View Existing Data", si possono vedere i dati esistenti nella tabella di destinazione. Nella stessa finestra, andando in "Mappings", nel pannello a destra, si possono mappare le colonne estratte nella tabella di destinazione e si otterrà la seguente schermata (vedi immagine 2.20):

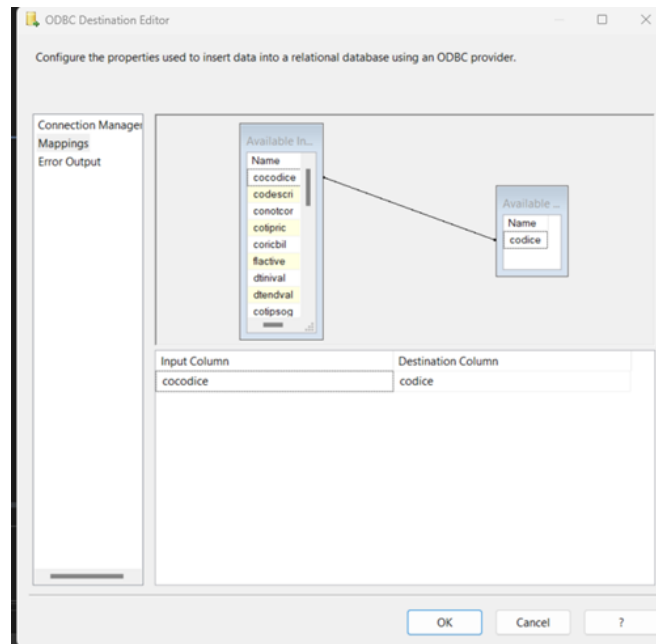


Figura 2.20: Configurazione componente Destinazione ODBC Mappings.

Nel mio caso bisogna mappare la colonna di input "cocode" nella colonna "codice" della tabella di output: si trascina la colonna "cocode" su "codice" e, fatto ciò, non rimane che collegare il componente "Colonne derivate" al componente "Destinazione ODBC" (come fatto prima dal componente "Origine ODBC" al componente "Colonna derivata"). Con questa operazione ho realizzato il caso d'uso e avviando l'esecuzione, si avrà la tabella destinazione, riempita con i codici dei clienti presenti nella tabella del database di destinazione indicata.

2.8 Intelligenza artificiale (AI) e machine learning (ML) nell'ETL

L'ETL incentrato su AI e ML automatizza le pratiche critiche inerenti ai dati, garantendo che i dati ricevuti per l'analisi rispettino gli standard di qualità necessari per fornire approfondimenti sicuri per il processo decisionale. Può essere unito ad altri strumenti di qualità dei dati per assicurare che i dati in uscita rispettino le specifiche.

2.8.1 ETL e democratizzazione dei dati

L'aggiunta dell'intelligenza artificiale nel processo ETL in fase di progettazione e di esecuzione semplifica il conseguimento di questo obiettivo. Gli strumenti ETL idonei all'AI e al ML possono apprendere dai dati storici. Quindi, gli strumenti possono indicare i migliori componenti adattabili per il contesto degli utenti aziendali e possono comprendere mappature dei dati, mapplet, trasformazioni, pattern, configurazioni e altro ancora per il contesto degli utenti aziendali. L'effetto è un incremento della produttività del team. Inoltre, l'automazione agevola la concordanza alle policy, visto che l'intervento umano è ristretto.

2.8.2 Automatizzare le pipeline ETL

Gli strumenti ETL incentrati sull'intelligenza artificiale permettono di automatizzare il lavoro di ingegneria dei dati impegnativo e ripetuto; permettono anche di incrementare l'efficacia della gestione dei dati e di sveltire la consegna dei dati. Inoltre, si può apprendere, sviluppare, integrare, accrescere, elaborare, mappare, stabilire e registrare automaticamente i dati per il data warehousing.

2.8.3 Rendere operativi i modelli di AI e ML con l'ETL

Gli strumenti ETL in cloud permettono di controllare in maniera efficiente gli ampi volumi di dati desiderato dalle pipeline di dati impiegati in AI e ML. Con lo strumento corretto, si possono trascinare e rilasciare le trasformazioni ML nelle mappature dei dati. Ciò produce i carichi di lavoro della scienza dei dati più resistente, efficienti e semplici da conservare. Inoltre, gli strumenti ETL alimentati dall'AI consentono di utilizzare comodamente l'integrazione continua/consegna continua (CI/CD), DataOps e MLOps per automatizzare la pipeline di dati.

2.8.4 Replicare il database con l'acquisizione dei dati di modifica (CDC)

L'ETL viene impiegato per ripetere e sincronizzare automaticamente i dati da diversi database sorgenti a un data warehouse nel cloud. Le sorgenti possono com-

prendere MySQL, PostgreSQL, Oracle e altri. Per risparmiare tempo e incrementare l'efficienza, ci si può servire della cattura dei dati di modifica per automatizzare il processo e aggiornare soltanto i set di dati cambiati.

2.8.5 Maggiore agilità aziendale grazie all'ETL per l'elaborazione dei dati

I team si muoveranno più velocemente perché tale processo diminuisce lo sforzo essenziale per raggruppare, predisporre e rafforzare i dati. L'automazione ETL incentrata sull'intelligenza artificiale migliora la produttività e permette ai professionisti di accedere ai dati di cui hanno necessità, senza dover scrivere codice o script. Questo permette di guadagnare tempo e risorse preziose [10].

2.9 ETL e passaggio al cloud

La connettività dei dispositivi, l'archiviazione economica e i connettori numerosi e veloci hanno creato una grande richiesta di dati da parte delle aziende per essere competitive. I dati provengono da molte fonti e non sono più concentrati in un solo data center. Molti studi mostrano che la maggior parte delle aziende usa o vuole usare più servizi cloud; questo significa che i processi ETL devono usare sistemi di archiviazione e/o elaborazione nel cloud o parzialmente nel cloud. L'ETL nel cloud (o ETL nativo nel cloud) cerca di usare i vantaggi della tecnologia cloud: scalabilità, parallelismo, flessibilità, autoscaling, serverless, ribilanciamento, pianificazione e altro. Ci sono diverse soluzioni commerciali che offrono alcune di queste funzionalità, ma come integrarle bene nel Cloud ETL è una sfida interessante per la ricerca futura [3].

Dal tool al microservizio

3.1 Scelta del tool

Dopo la fase di analisi dei tool, la scelta è ricaduta sul tool Talend Open Studio for Data Integration, che permette di estrarre la classe Java che viene generata dal job che si crea; così facendo si può successivamente inserirla in un microservizio, dedicato al processo ETL, e modificarla per i propri scopi. Grazie al tool, sono stati realizzati i processi ETL per i dipendenti, i clienti e le commesse e sono state estratte le classi Java relative.

3.2 Che cos'è un microservizio?

Un microservizio è un'applicazione autonoma e leggera, che può cambiare e crescere in modo autonomo, scegliendo liberamente la sua struttura, tecnologia e piattaforma; ha un proprio ciclo di vita, una propria metodologia di sviluppo e può essere gestito, distribuito e scalato in modo autonomo. Un'architettura basata su microservizi è un modello di architettura per applicazioni distribuite, in cui l'applicazione è formata da una collezione di componenti più piccoli e indipendenti (cioè sono delle piccole applicazioni a sé). In questo modo si creano applicazioni singole

che eseguono una funzione specifica e poi si combinano per formare l'applicazione principale. Un microservizio deve essere distribuito in modo indipendente e, inoltre, ogni microservizio ha una propria architettura di distribuzione e non condivide risorse (come container, cache, datastore) con altri componenti dell'organizzazione. Il ridimensionamento di un microservizio è indipendente dagli altri microservizi e l'incremento delle prestazioni di un microservizio non dovrebbe influenzare i suoi consumatori. Le applicazioni offrono interfacce che possono essere usate da altre applicazioni per interagire, mentre un microservizio espone le proprie interfacce per le comunicazioni. I microservizi, inoltre, sono realizzati usando protocolli di comunicazione internet come HTTP e seguono standard aperti come REST e usano tecnologie di trasferimento dati come JSON [11].

3.2.1 Caratteristiche dei microservizi

Le caratteristiche di un microservizio sono:

- **Piccoli e focalizzati:** i microservizi sono caratterizzati da una granularità fine e una forte coesione, che li rende focalizzati su una singola responsabilità aziendale. Questa responsabilità deve essere definita in base al dominio di business tecnologici (non in base ai vincoli organizzativi) o di comunicazione. Ogni microservizio è un'applicazione a sé stante, con il suo codice sorgente e il suo processo di distribuzione. Questo approccio favorisce l'indipendenza, la scalabilità e l'evoluzione dei servizi.
- **Accoppiamento debole:** ogni microservizio può essere rilasciato in base alle proprie esigenze, senza dover attendere o sincronizzarsi con gli altri servizi. Questo favorisce una maggiore frequenza e velocità di distribuzione, che si traduce in una migliore capacità di adattamento dell'applicazione alle richieste degli utenti.
- **Linguaggio indipendente:** i microservizi consentono agli sviluppatori di scegliere la tecnologia più adatta per ogni servizio, senza dover seguire uno standard imposto dall'architettura. Questo significa che i team di sviluppo possono utilizzare i linguaggi di programmazione con cui si sentono più a loro agio e che

meglio si adattano al problema da risolvere. In questo modo, i microservizi sfruttano la diversità delle tecnologie e delle competenze dei team, aumentando la qualità e l'efficienza del software prodotto.

- **Contesto delimitato:** è una parte dell'architettura a microservizi che si occupa di un aspetto specifico del dominio. All'interno di un contesto delimitato, il modello dei dati, il modello del dominio e le regole di business sono coerenti e isolati dagli altri contesti. Questo permette di semplificare la complessità e di favorire l'autonomia dei servizi. Un contesto delimitato ha anche delle interfacce ben definite per comunicare con gli altri contesti, rispettando i principi di integrazione. Nell'architettura a microservizi, è importante progettare i contesti delimitati in modo chiaro e consistente; questo aiuta a evitare ambiguità, conflitti e dipendenze tra i servizi. Alcuni modelli possono essere comuni a più contesti, mentre altri sono specifici di un solo contesto.

3.2.2 Vantaggi dei microservizi

I vantaggi di un microservizio sono:

- **Isolamento dei guasti:** i guasti sono inevitabili, ma si possono gestire in modo efficace e tempestivo. Quando un servizio si interrompe, bisogna essere in grado di individuare e risolvere il problema il prima possibile. Per fare ciò, è necessario applicare un approccio modulare, in cui si suddivide il sistema in componenti più piccoli e indipendenti. In questo modo, si possono usare o creare strumenti che permettano di rilasciare frequentemente piccole modifiche, in modo che lo sviluppatore possa modificare un solo elemento per volta e se si verifica un errore, si capisce esattamente cosa ha causato il guasto.
- **Osservabilità:** per realizzare un'architettura a microservizi efficace, è essenziale monitorare lo stato di salute di ogni servizio del sistema, per poter intervenire tempestivamente in caso di malfunzionamenti. Questo implica anche l'utilizzo di un sistema di logging avanzato per registrare, conservare e rendere ricercabili i log per una migliore analisi. Questo compito diventa più complesso quando si devono gestire molteplici componenti distribuiti e dinamici. Per questo motivo,

la progettazione dell'osservazione deve essere in grado di presentare i dati in modo chiaro e significativo, in modo da fornire informazioni utili per l'analisi.

- **Requisiti di automazione:** l'automazione è il componente più rilevante e critico da gestire. La diffusione del numero di servizi e delle relazioni tra di essi avverrà a un certo punto, probabilmente molto presto, e non potrà essere controllata senza un modo per automatizzare le cose.
- **Elevata indipendenza:** i microservizi si basano sul principio di disaccoppiamento dei servizi. Questo significa che ogni servizio deve essere progettato e implementato in modo da non dipendere da altri servizi, ma da comunicare con essi tramite interfacce ben definite. In questo modo, i servizi possono essere modificati e rilasciati in modo autonomo, senza impattare sul funzionamento degli altri servizi.
- **Testing:** i test sono una parte fondamentale dello sviluppo di un'architettura a microservizi. Con più componenti che interagiscono tra loro, aumenta il rischio di errori e malfunzionamenti. Per garantire la qualità del sistema, è necessario definire una strategia di test che copra tutti gli aspetti rilevanti, sia funzionali che non funzionali. I test automatizzati sono diventati sempre più avanzati e affidabili, grazie all'uso di nuovi strumenti e metodologie. Tuttavia, esistono ancora delle difficoltà nel testare in modo ottimale le caratteristiche e le prestazioni del sistema, soprattutto in un contesto distribuito. L'architettura a microservizi introduce una nuova sfida per i test, richiedendo di gestire la complessità derivante dalla modularità e dalla collaborazione dei componenti.
- **Scalabilità:** quando si utilizzano i microservizi, bisogna tenere conto della loro crescita esponenziale nel sistema, così come delle diverse versioni che possono coesistere. Questo comporta un aumento delle connessioni tra i servizi e una maggiore complessità dell'architettura. Per gestire questa situazione, si devono adottare soluzioni specifiche, come il service discovery per localizzare i servizi attivi, un meccanismo di routing per smistare il traffico tra le API dei servizi, una gestione della configurazione più efficace per aggiornare dinamicamente le impostazioni e applicare le modifiche al sistema e così via [12].

3.3 Architettura a microservizi utilizzata

Il microservizio è stato sviluppato con la tecnologia Spring microservices, che mette a disposizione già una sua architettura (vedi immagine 3.1):

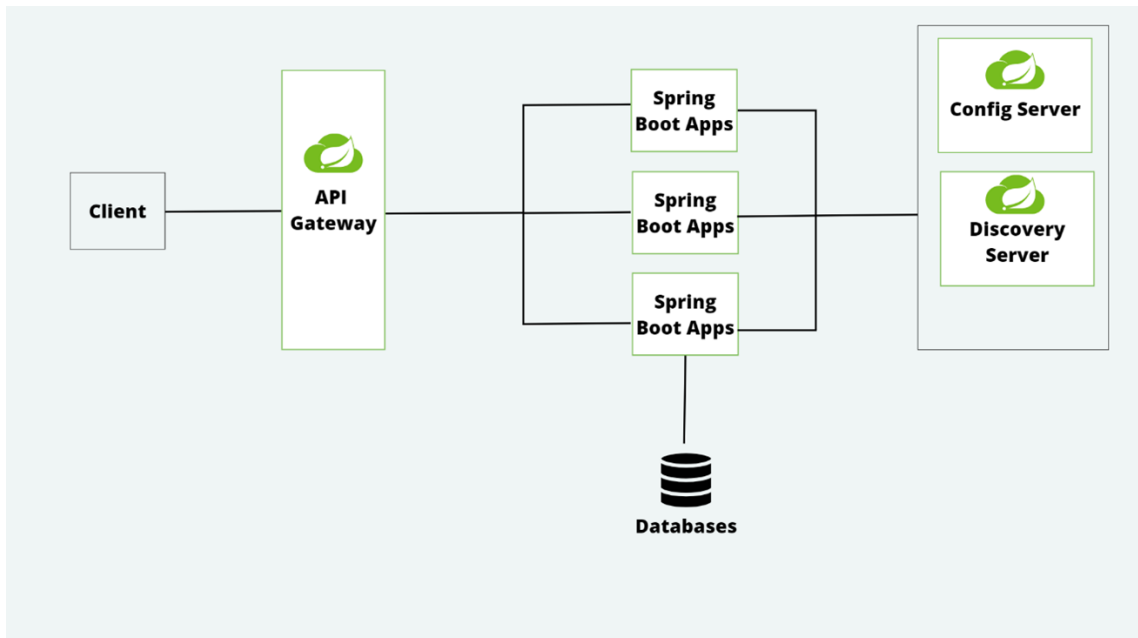


Figura 3.1: Architettura Spring.

Nell'architettura troviamo:

- **Discovery server:** è il registro dei servizi, che agisce come un elenco telefonico per i microservizi. Ogni microservizio si autoregistra, comunicando al registro dove risiede (host, nome del nodo, porta, ecc.). Il registro dei servizi mantiene un elenco aggiornato di tutti i microservizi disponibili e li monitora per verificare il loro stato. In questo modo, i microservizi possono scoprire e comunicare tra loro in modo efficiente e affidabile.
- **API gateway:** funge da punto di ingresso per i clienti che vogliono interagire con i servizi esposti. Il gateway API può filtrare le richieste in base alle autorizzazioni e alle regole di routing, e nascondere i dettagli interni dei microservizi non pubblici. In questo modo, il cliente ha un'interfaccia semplice e uniforme per accedere ai microservizi necessari.
- **Config server:** fornisce un meccanismo per gestire i parametri di configurazione.

3.3.1 Discovery Server

Per il discovery server in Spring cloud si utilizza Eureka: è una soluzione per la gestione dei servizi basata su REST che si adatta bene alle architetture cloud pubbliche. Con Eureka, i microservizi possono essere scoperti e utilizzati per il load balancing e il failover. Eureka fa parte di Spring Cloud, che offre un registro dei servizi dove i microservizi si registrano con le loro informazioni (host, nome del nodo, porta, ecc.). Eureka ha due componenti, il server Eureka e il client Eureka, che facilitano l'interazione con il registro dei servizi. Il client ha anche un load balancer integrato che fa un semplice load balancing round-robin. Uno dei vantaggi di usare un registro dei servizi come Eureka è che permette ai consumatori di servizi di usare un nome logico di servizio per cercare nel registro, invece di usare un indirizzo fisico. Questo nome logico di servizio può essere associato a un servizio effettivo che ha diverse istanze con diversi indirizzi fisici. In questo modo, i consumatori di servizi sono protetti da eventuali cambiamenti nei dettagli del servizio effettivo o nell'ambiente di distribuzione.

3.3.2 API gateway

Un modo per rendere accessibili i microservizi ai clienti al di fuori del perimetro è usare un gateway API, che permette di esporre solo quelli che si vogliono rendere pubblici e di nascondere gli altri. Il gateway API agisce come un punto di ingresso unico all'ambiente applicativo aziendale e fornisce gli URL ai clienti per accedere ai microservizi esposti. Il gateway API consulta il registro e poi reindirizza la richiesta al microservizio appropriato.

3.3.3 Config server

La gestione dei parametri di configurazione è un aspetto cruciale dei microservizi. Questi parametri riguardano sia l'applicazione che l'infrastruttura. Con molti microservizi, è necessario avere un modo efficace per gestire i parametri di configurazione esterni e il Config Server di Spring Cloud offre una soluzione per questo problema. Il Config Server di Spring Cloud conserva i parametri di configurazione in un repo-

sitory, che può essere locale o remoto, e può essere configurato in modalità a nodo singolo o a cluster per garantire l'alta disponibilità. Grazie al controllo di versione, è possibile avere un riferimento diretto ai parametri di configurazione dai server di produzione, evitando così errori di runtime causati da valori sbagliati o simili [13].

3.3.4 Microservizio ETL

È il microservizio creato per eseguire i processi ETL; contiene le classi Java generate ed esportate dal tool Talend e modificate opportunamente. Queste sono:

- ListaClienti: è la lista per ottenere i dati dei clienti;
- ListaCommesse: è la lista per ottenere i dati delle commesse;
- ListaDipendenti: è la lista per ottenere i dati dei dipendenti;

3.4 Come estrarre la classe Java da Talend

Una volta scelto il tool Talend e realizzato i processi ETL per i dipendenti, i clienti e le commesse non resta che estrarre la relativa classe Java. Per fare questo, si clicca in "File", poi in "Export" e si avrà la seguente schermata (vedi immagine 3.2):

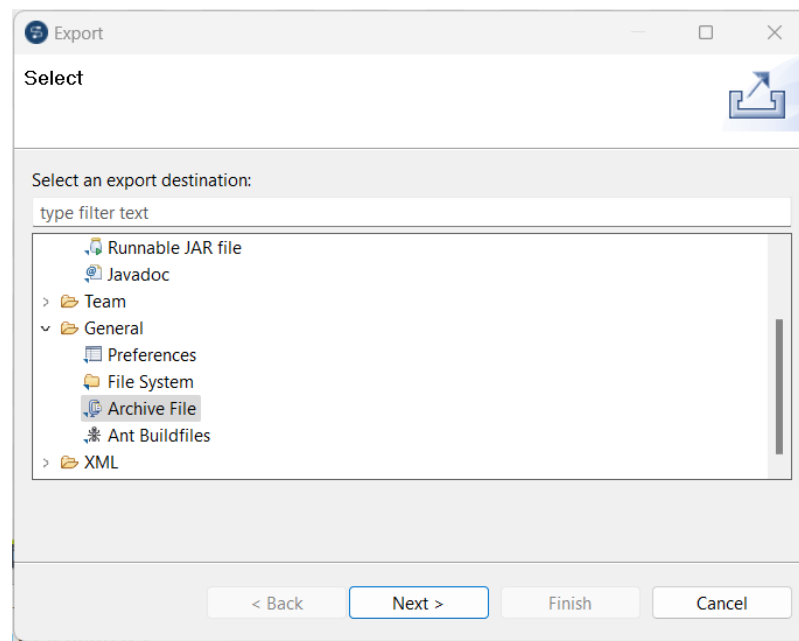


Figura 3.2: Esporta classe Java passo 1.

Facendo click su "Next" si ottiene (vedi immagine 3.3):

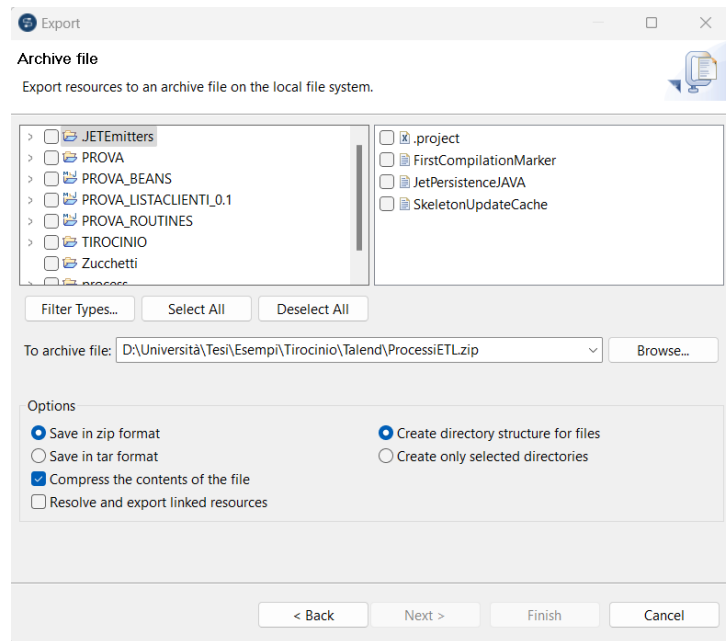


Figura 3.3: Esporta classe Java passo 2.

Si seleziona la cartella di interesse (ad esempio PROVA_LISTACLIENTI_0.1), il percorso dove salvare la cartella e si clicca su "Finish". La cartella è una zip, vado ad estrarla e poi in "src->main->java->prova->listaclienti_0_1"; qui trovo la classe Java relativa al processo ETL realizzato (questo procedimento è stato fatto per tutti e tre i job che ho realizzato). Fatto ciò inserisco ogni classe nel microservizio dedicato all'ETL.

3.5 Realizzazione del microservizio

Per realizzare il microservizio è stato usato l'IDE IntelliJ IDEA; innanzitutto bisogna creare il modulo del microservizio: si va nella cartella del microservizio, nel pannello a sinistra, si clicca il tasto destro, poi su "New", su "Module" e si apre una schermata dove si inserisce il nome del microservizio e si fa click su "Create". Una volta fatto ciò è stato creato il microservizio; al suo interno si inseriscono le classi Java che sono state estratte da Talend. In queste classi si modificano i parametri delle connessioni al database di input e output per renderle più dinamiche, in quanto

sono parametri statici. Le linee di codice seguenti servono ad ottenere i parametri del database di output:

```

1 OutputDB outputDB = new OutputDB();
2 dbschema_tDBOutput_1 = outputDB.getSchema();
3 String url_tDBOutput_1 = outputDB.getUrl();
4 dbUser_tDBOutput_1 = outputDB.getUsername();
5 final String decryptedPassword_tDBOutput_1 =
6 PasswordEncryptUtil.decryptPassword(outputDB.getPassword());

```

Mentre queste sono per il database di input:

```

1 ExtractDB input = new ExtractDB();
2 String dbUser_tDBInput_1 = input.getUsername();
3 final String decryptedPassword_tDBInput_1 =
4 PasswordEncryptUtil.decryptPassword(input.getPassword());
5 String url_tDBInput_1 = input.getUrl();
6 String schema = input.getSchema();

```

Queste linee di codice sono state inserite alla ListaClienti, ListaCommesse e ListaDipendenti.

3.5.1 Classe OutputDB

```

1 public void init() throws IOException
2 {
3     Proprietà proprietà=new Proprietà(urlProperties);
4     JSONObject jsonObject=proprietà.read();
5     url=jsonObject.get("spring.outputdatasource.jdbc-url").toString();
6     username=jsonObject.get("spring.outputdatasource.username").toString();
7     password=jsonObject.get("spring.outputdatasource.password").toString();
8     schema=jsonObject.get("spring.outputdatasource.schema").toString();
9 }
10 public void creazioneUrl() throws IOException
11 {
12     proprietà= PropertiesLoaderUtils.loadAllProperties
13         ("application.properties");
14     String configUri=proprietà.getProperty("spring.config.uri");
15     String applicationName="/".concat
16         (proprietà.getProperty("spring.application.name"));
17     String profilo="/".concat
18         (proprietà.getProperty("spring.profiles.active"));

```

```
19 urlProperties=configUri.concat(applicationName);
20 urlProperties=urlProperties.concat(profilo);
21 }
```

Questa classe con il metodo `creazioneUrl()` crea l'url per prendere i parametri di configurazione del database di output, che sono contenuti nel microservizio `ConfigServer`. Il metodo `init()` legge le proprietà dall'url creata e le memorizza nelle rispettive variabili.

3.5.2 Classe ExtractDB

```
1 public void init() throws IOException
2 {
3     Proprietà proprietà=new Proprietà(getUrlProperties());
4     JSONObject jsonObject=proprietà.read();
5     url=jsonObject.get("spring.datasource.jdbc-url").toString();
6     username=jsonObject.get("spring.datasource.username").toString();
7     password=jsonObject.get("spring.datasource.password").toString();
8     schema=jsonObject.get("spring.datasource.schema").toString();
9 }
10 public void creazioneUrl() throws IOException
11 {
12     proprietà= PropertiesLoaderUtils.loadAllProperties
13         ("application.properties");
14     String configUri=proprietà.getProperty("spring.config.uri");
15     String applicationName="/".concat
16         (proprietà.getProperty("spring.application.name"));
17     String profilo="/".concat
18         (proprietà.getProperty("spring.profiles.active"));
19     urlProperties=configUri.concat(applicationName);
20     urlProperties=urlProperties.concat(profilo);
21 }
```

Questa classe è come la classe `OutputDB`, ma questa serve per i parametri di configurazione al database di input.

3.5.3 Classe Proprietà

```
1 public JSONObject read()
2 {
3     if(url==null || url.equals("null"))
4     {
5         return null;
6     }
7     else
8     {
9         risposta = restTemplate.getForObject(url, String.class);
10        JSONObject jsonRisposta = new JSONObject(risposta);
11        JSONObject jsonObject = scompat(jsonRisposta);
12        return jsonObject;
13    }
14 }
```

Questa classe costruisce un oggetto JSON, data una url, dove si inseriscono i parametri di configurazione. Il metodo read serve per leggere la risposta dell'url data, mentre il metodo scompat costruisce il JSON con i valori di configurazione necessari.

3.5.4 Classe Controller

Questa classe serve per esporre le API per far sì di poter accedere alle varie liste, come ad esempio:

```
1 @GetMapping("/commesse")
2 public void commesse()
3 {
4     ListaCommesse commesse=new ListaCommesse();
5     commesse.runJob(new String[]{" "});
6 }
```

Se accedo a questa API, vado ad eseguire la lista delle commesse. Questo è stato fatto anche per la lista clienti e la lista dipendenti.

Tutte queste classi realizzate sono state testate con JUnit 5. Inoltre, per distribuire i microservizi è stato utilizzato Docker.

3.6 JUnit

JUnit è un framework open source per Java per la scrittura dei test di unità. Il principio è quello di testare e codificare allo stesso tempo: i programmatori scrivono delle righe di codice, e successivamente un test; se il test fallisce, aggiustano l'errore. Se tutto va bene, scrivono un altro codice e altri test, e così via. Tutti i test sono salvati insieme al codice, così se il codice viene modificato, è semplice fare test di regressione per verificare che le modifiche non abbiano compromesso il software [14]. Una volta realizzate le classi Java con le modifiche opportune, per verificare il loro funzionamento sono state testate con il framework JUnit. Per fare ciò sono state create delle classi di test per ogni classe da testare e sono state usate principalmente due annotazioni: `@ExtendWith` e `@SpringBootTest`, che permettono di eseguire i test con Spring e JUnit. Nelle classi di test, sono state controllate:

- Classi `ExtractDB` e `OutputDB`: sono state verificate se le variabili ottenute dal file `properties` non fossero nulle;
- Classe `Controller`: è stato verificato se l'URL dell'API, corrispondente alle varie classi estratte e modificate da Talend, funzionasse correttamente usando l'annotazione `@WebMvcTest` e l'oggetto `MockMvc`;
- Classi `ListaClienti`, `ListaDipendenti`, `ListaCommesse`: sono stati verificati se i dati che si estraggono con questi classi sono presenti nel database;
- Classe `Proprietà`: è stato verificato se la lettura del file `properties` e la creazione del relativo oggetto `JSON` è corretto.

Di seguito viene mostrato un esempio che serve per testare se la `username` non è nulla.

```
1  @Test
2  public void testGetUsername() throws IOException
3  {
4      extractDB=new ExtractDB();
5      String username=extractDB.getUsername();
6      assertNotNull(username);
7  }
```

Inoltre, per ogni classe testata è stato prodotto anche un report con i risultati dei test, come si può vedere nell'immagine 3.4

JUNIT TEST CLASSE EXTRACTDBUNITTEST								
ID	NOME METODO	DESCRIZIONE	INPUT	OUTPUT ATTESI	POST CONDIZIONI ATTESE	OUTPUT OTTENUTI	POST CONDIZIONI OTTENUTE	ESITO (FAIL, PASS)
1	testGetUsername	Test del metodo getUsername della classe ExtractDb, per ottenere l'username dalle properties.	null.	Ritorno della Username diversa dal valore null.	null.	Otteniamo l'username diversa dal valore null.	null.	PASS
2	testGetUsernameFallido	Test del metodo getUsername quando restituisce null, della classe ExtractDb, per ottenere l'username dalle properties.	null.	Ritorno della Username col valore null.	null.	Otteniamo l'username col valore null.	null.	PASS
3	testGetPassword	Test del metodo getPassword della classe ExtractDb, per ottenere la password dalle properties.	null.	Ritorno della Password diversa dal valore null.	null.	Otteniamo la password diversa dal valore null.	null.	PASS

Figura 3.4: Esempio di report junit.

3.7 Docker

Docker è una soluzione innovativa per creare, distribuire ed eseguire applicazioni software in modo rapido e sicuro. Si basa su una tecnologia di virtualizzazione leggera che permette di isolare le applicazioni in container indipendenti e portatili. Questi container contengono tutto ciò che serve per far funzionare le applicazioni, dalle dipendenze alle configurazioni, e possono essere spostati facilmente tra diverse piattaforme e infrastrutture, anche nel cloud. Il container Docker è il luogo dove si sviluppano le applicazioni, si assemblano con tutti gli elementi necessari per il loro funzionamento e si sottopongono a eventuali test o verifiche di qualità. La creazione del container può avvenire su quasi tutti i sistemi operativi e le infrastrutture, compreso l'ambiente cloud [12]. Si è scelto di usare Docker perchè offre diversi vantaggi: permette di distribuire più rapidamente l'applicazione perchè con un solo comando si può eseguire il file Docker sul server e avviare i servizi che servono; permette di individuare quale servizio va in errore e, essendo che isola in container separati e indipendenti ogni servizio, l'applicazione continuerà a funzionare; permette il trasfe-

rimento da un ambiente all'altro; consente di aggiungere o rimuovere un servizio facilmente. Per configurare Docker bisogna mettere un file nel microservizio che indica su che porta è esposto il microservizio.

```
1 FROM eclipse-temurin:17-jdk-alpine
2 EXPOSE 9078
3 ADD target/*.jar etl.jar
4 ENTRYPOINT ["java", "-jar", "/etl.jar"]
```

Poi nella cartella principale va messo un file che contiene come far partire i vari microservizi e da che microservizi dipendono. Per esempio per il microservizio trattato, si ha:

```
1 etl:
2   # --- CONTAINER NAME ---
3   container_name: etl-microservice
4   build:
5     context: etlMicroservice/
6     dockerfile: Dockerfile
7   # --- PORT MAPPING ---
8   ports:
9     - 9078:9078
10  # --- ENVIRONMENT VARIABLE ---
11  environment:
12    - eureka.client.serviceUrl.defaultZone=http://discovery:9071/eureka
13  depends_on:
14    - discovery
15    - config
16  restart: always
```

Infine per tutto il progetto è stata usata la metodologia Agile Scrum.

3.8 Metodologia Agile Scrum

Scrum è un approccio adattivo per gestire progetti complessi che richiedono una varietà di requisiti, con benefici come la selezione flessibile dei requisiti per gli sprint e la mancanza di regole rigide da seguire. Il flusso di lavoro di scrum si basa su una stretta interazione tra lo scrum team, il Master e il Product Owner durante cicli ripetuti del software in evoluzione. Il processo di scrum coinvolge uno scrum master,

il Product Owner e il team di scrum. Il ruolo principale dello scrum master è quello di rimuovere gli ostacoli. Il team di scrum è multidisciplinare e include sviluppatori, tester e altri specialisti di vari settori necessari allo sviluppo, che portano a un prodotto finale versatile e innovativo che soddisfa il cliente. Uno sprint è il blocco più piccolo di Scrum ed è la fase in cui un piccolo team lavora su un obiettivo prefissato. La durata dello sprint varia da una a tre settimane. L'obiettivo dello sprint è scelto dallo sprint backlog, che è una raccolta di tutti i requisiti dello sprint corrente su cui lavorare. Questi requisiti sono richiesti dal cliente e sono chiamati storie dell'utente: vengono divisi in backlog di sprint, seguiti dalla pianificazione di sprint che contiene la metodologia per portare a termine uno sprint. Il team si riunisce quotidianamente per una breve riunione chiamata daily scrum, in cui ogni membro condivide lo stato del proprio lavoro, i problemi incontrati e i piani per il giorno successivo. Lo scopo di uno sprint è produrre un prodotto che sia pronto per essere consegnato al cliente. Al termine di uno sprint, il team presenta il risultato del proprio lavoro al cliente o al proprietario del prodotto in una riunione chiamata sprint review, in cui si raccolgono i feedback e si valuta la soddisfazione degli obiettivi. Il product owner non può cambiare gli obiettivi di sprint una volta che sono stati definiti, ma può aggiornare il progetto con nuove funzionalità che emergono durante lo sviluppo. Il piano e il processo di rilascio vengono migliorati grazie alla retrospettiva di ogni sprint e alla verifica di ogni rilascio. Il progetto si basa su una serie di sprint pianificati, che hanno una durata prefissata, per produrre il risultato finale pronto per essere consegnato e testato secondo le aspettative del product owner [15]. Questa è stata la metodologia adottata per lo sviluppo del progetto.

Prima di ogni sprint, lo scrum master e il product owner hanno definito le attività da svolgere nello sprint, poi il team si è incontrato e ha pianificato i vari task tenendo conto dei tempi personali, basati sull'esperienza (ogni sprint è durato 15/20 giorni). Durante lo sprint, ogni mattina si è tenuta una riunione per condividere cosa era stato fatto il giorno precedente, se c'erano problemi da risolvere e cosa si sarebbe fatto quel giorno. Il team aveva a disposizione una lavagna o board dove segnare i task, suddivisi in task da fare, task in corso e task completati, in modo che sia gli altri membri del team che lo scrum master potessero essere aggiornati in qualsiasi momento semplicemente guardando la board, come si può vedere nell'immagine 3.5.

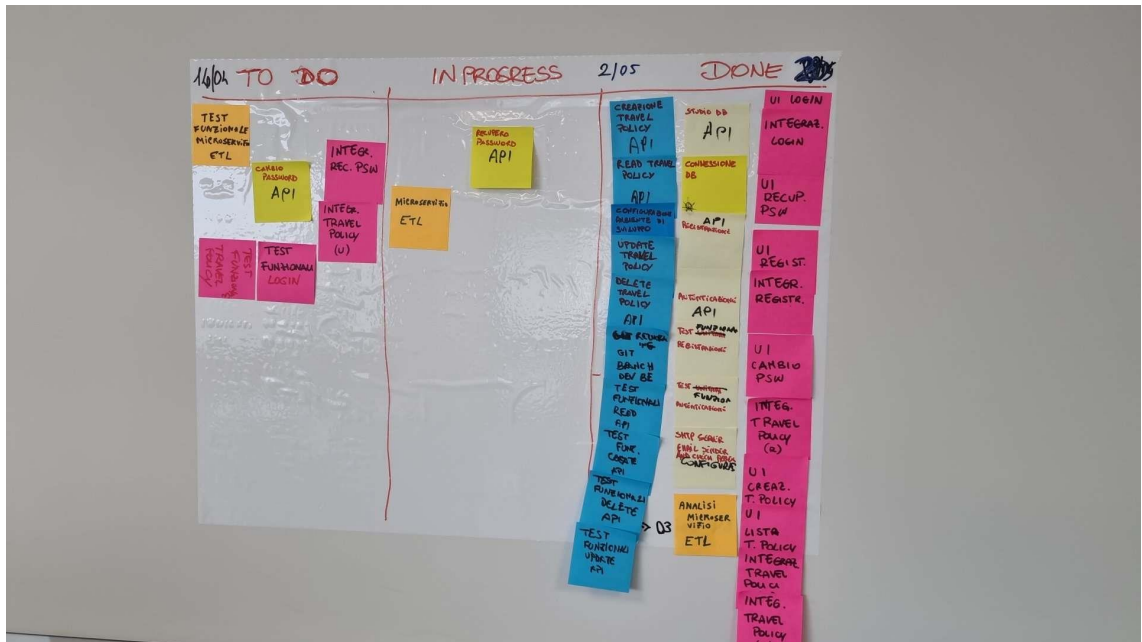


Figura 3.5: Esempio di board scrum.

Al termine dello sprint, il team si è riunito con lo scrum master e il product owner per mostrare tramite una demo le funzionalità implementate e ricevere feedback e suggerimenti. Dopo la demo, il team ha discusso delle cose che erano andate bene e quelle che erano andate male in quello sprint e come si sarebbero potute migliorare. Infine, il team ha fatto anche la retrospettiva, cioè ha analizzato i task che non erano stati completati e i task emersi dai feedback della demo e li ha inseriti nella pianificazione del nuovo sprint.

CAPITOLO 4

Analisi dei risultati

Dopo che è stato realizzato il microservizio ed integrato con l'applicazione btrip, è stata realizzata una valutazione attraverso intervista a due dipendenti dell'azienda. Di seguito sono riportati i risultati ottenuti dalle interviste.

4.1 Come veniva svolta l'operazione ETL

Dopo aver analizzato le risposte dei dipendenti, è stato rilevato che l'operazione ETL non era stata implementata prima. L'azienda ha deciso di introdurla quando ha incontrato il problema durante lo sviluppo del progetto e ha pensato di risolverlo con un microservizio per l'ETL. Questa soluzione, basata sulle esperienze precedenti, è stata valutata come la migliore nell'analisi dei requisiti.

4.2 Valutazione utilità ed efficacia del microservizio ETL

Dalle risposte dei dipendenti, è stato accertato che il microservizio realizzato ha dimostrato di essere una soluzione efficace per gestire i dati provenienti dalle diverse

fonti, sia interne che esterne all'azienda. Grazie al microservizio realizzato è possibile aggiornare, estrarre e trasformare i dati, facilitando lo sviluppo dell'applicativo. La tecnologia ETL è un'ottima tecnologia che viene, però, ancora poco sfruttata (anche se, rispetto agli anni precedenti, varie aziende ne stanno facendo uso).

4.3 Come si può migliorare il microservizio

Il microservizio potrebbe essere migliorato acquisendo dati da diverse fonti e formati: per fare questo si potrebbe parametrizzare il microservizio in modo che possa riconoscere e gestire i vari tipi di dati in ingresso. Inoltre, si potrebbero utilizzare tool meno complessi e più all'avanguardia in quanto alcune volte l'utilizzo dei tool desta un po' di complessità nel codice e, generando le classi, diventa complesso strutturare classi già predefinite; per di più ci possono essere delle variabili che possono comportare problematiche nel codice.

4.4 Utilizzo del microservizio ETL a livello aziendale

Il microservizio può essere utilizzato per l'aggiornamento continuo dei dati dei dipendenti di un'azienda, dei progetti di un'azienda e dei clienti. Un altro utilizzo potrebbe essere quello di utilizzare il microservizio quando si necessita di fare delle chiamate ETL continue per aggiornare i dati. Il microservizio potrebbe diventare un componente shared in varie applicazioni aziendali, rivolta all'integrazione dei dati per diversi scopi (ad esempio amministrativi o risorse umane); questo anche perché, per la maggior parte dei software, è una necessità comune integrare dati provenienti da fonti di dati differenti.

4.5 Vantaggi del microservizio ETL

Il microservizio fornisce i seguenti vantaggi: agilità, semplicità di distribuzione, libertà tecnologica, codice riutilizzabile, resilienza, scalabilità, flessibilità. Il microservizio ETL permette con un'unica chiamata API di aggiornare i dati dei dipendenti, i clienti e i progetti presenti nel database aziendale e trasformarli e trasferirli al

database del software aziendale. Quindi, il vantaggio principale è la semplicità nell'aggiornare i dati e, essendo un microservizio, ci fornisce la portabilità in altri contesti.

4.6 Risultati ottenuti

Dalle risposte dei due dipendenti, si è avuto un riscontro positivo nello sviluppo dell'applicazione sotto l'aspetto dell'utilità sia del microservizio che della tecnologia ETL. Inoltre, questo microservizio può essere migliorato dall'azienda e può anche essere usato per altri progetti dell'azienda semplificando il lavoro.

CAPITOLO 5

Conclusioni

Questa tesi presenta una ricerca sulla tecnologia ETL accennando anche all'intelligenza artificiale e al machine learning e un'analisi comparativa di diversi tool disponibili online. Con alcuni di questi tool sono stati realizzati e illustrati degli esempi pratici. Successivamente, è stata fatta una panoramica sui microservizi ed è stata spiegata la scelta di usare il tool Talend per creare un microservizio ETL basato sulle classi Java generate dal tool stesso. Il microservizio è stato poi valutato tramite delle interviste a due dipendenti dell'azienda che lo hanno utilizzato. In conclusione, per gli sviluppi futuri, si potrebbe migliorare il microservizio ETL rendendolo in grado di gestire dati provenienti da fonti e formati diversi, riuscendo anche ad integrarlo in diversi progetti; inoltre, si potrebbe anche fare in modo che il microservizio venga eseguito quando i dati subiscono delle modifiche dei dati oppure si potrebbe impostare un intervallo di tempo per quando si deve effettuare l'aggiornamento. Inoltre, si potrebbe integrare il microservizio con l'intelligenza artificiale e il machine learning.

Bibliografia

- [1] J. Goldfedder, *Choosing an ETL Tool*. Berkeley, CA: Apress, 2020, pp. 75–101. [Online]. Available: https://doi.org/10.1007/978-1-4842-5653-4_5 (Citato a pagina 6)
- [2] J. Sreemathy, S. Priyadharshini, K. Radha, K. Sangeerna, and G. Nivetha, “Data validation in etl using talend,” in *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, 2019, pp. 1183–1186. (Citato alle pagine 6 e 8)
- [3] A. Simitsis, S. Skiadopoulos, and P. Vassiliadis, “The history, present, and future of etl technology,” 2023. (Citato alle pagine 8 e 29)
- [4] A. AWS, “Cos’è l’etl (estrazione, trasformazione, caricamento)?” [Online]. Available: <https://aws.amazon.com/it/what-is/etl/> (Citato a pagina 9)
- [5] J. Condemi, “Etl extract, transform and load: cos’è e come può migliorare efficientemente la gestione e l’interpretazione dei dati,” 2023. [Online]. Available: <https://www.bigdata4innovation.it/business-intelligence/etl-extract-transform-and-load-significato-funzione-e-tool/> (Citato a pagina 10)
- [6] R. Martoglia, A. Giannetti, and F. Barbanti, “Progetto, sviluppo e orchestrazione di pipeline etl tramite apache airflow in esecuzione su servizi cloud.” (Citato a pagina 10)

- [7] G. Bakos, *KNIME essentials*. Packt Publishing Ltd, 2013. (Citato a pagina 11)
- [8] V. Krishnan, "Research data analysis with power bi," 2017. (Citato a pagina 11)
- [9] P. Janus, "Sql server integration services," *Pro PerformancePoint Server 2007: Building Business Intelligence Solutions*, pp. 53–82, 2008. (Citato a pagina 12)
- [10] Informatica, "Cos'è l'etl (estrazione del carico di trasformazione)?" [Online]. Available: <https://www.informatica.com/tw/resources/articles/what-is-etl.html> (Citato a pagina 29)
- [11] Z. Xiao, I. Wijegunaratne, and X. Qiang, "Reflections on soa and microservices," in *2016 4th International Conference on Enterprise Systems (ES)*, 2016, pp. 60–67. (Citato a pagina 31)
- [12] D. Jaramillo, D. V. Nguyen, and R. Smart, "Leveraging microservices architecture by using docker technology," in *SoutheastCon 2016*, 2016, pp. 1–5. (Citato alle pagine 33 e 42)
- [13] B. Christudas, *Spring Cloud*. Berkeley, CA: Apress, 2019, pp. 183–244. [Online]. Available: https://doi.org/10.1007/978-1-4842-4501-9_8 (Citato a pagina 36)
- [14] P. Louridas, "Junit: unit testing and coiling in tandem," *IEEE Software*, vol. 22, no. 4, pp. 12–15, 2005. (Citato a pagina 41)
- [15] A. Srivastava, S. Bhardwaj, and S. Saraswat, "Scrum model for agile methodology," in *2017 International Conference on Computing, Communication and Automation (ICCCA)*, 2017, pp. 864–869. (Citato a pagina 44)

Ringraziamenti

Vorrei ringraziare prima di tutto la mia famiglia, mio padre, mia madre, per avermi sempre sostenuto sia nei momenti belli che brutti e per tutti i sacrifici fatti; non mi hanno fatto mai arrendere, mi hanno permesso di realizzare questo sogno, dandomi sempre il loro supporto e vicinanza.

Ringrazio mia sorella, per avermi aiutato durante il mio percorso di studi, avermi sempre motivato e supportato in qualsiasi modo (anche per avermi aiutato con l'italiano nella stesura della tesi).

Ringrazio i miei zii e i miei cugini, per avermi sempre sostenuto, motivato e supportato durante tutto il mio percorso universitario.

Ringrazio i miei amici, Simona e Salvatore, per la compagnia durante gli anni di studio e le serate passate assieme, per avermi aiutato durante il mio percorso di studi e aiutato nelle scelte. Con voi ho condiviso un'importante periodo della mia vita tra felicità e difficoltà. Vi ringrazio anche per la comprensione avuta durante l'ultimo periodo.

Ringrazio Daniele, con cui ho condiviso il primo anno di studio, e tante serate. Per avermi sostenuto e motivato durante il mio percorso di studi, nonostante i chilometri di distanza.