Documentazione Data Pipeline
Pluvio/Termo

Indice

Descrizione del Problema

I seguenti paragrafi illustrano lo scenario di riferimento e le specifiche richieste dal cliente volte all'implementazione di una data pipeline.

Scenario

Nel territorio siciliano sono state installate 185 stazioni meteo che generano dati sulle precipitazioni (in millimetri) e sulla temperatura (in gradi centigradi). Il numero di stazioni può variare in base all'attivazione di nuove stazioni o alla dismissione delle esistenti. Ogni stazione è identificata in modo univoco dal nome della località in cui è stata installata. Alla fine di ogni giornata, ciascuna stazione genera due file in formato CSV, uno per la temperatura e uno per le precipitazioni. Ogni file contiene tutte le misurazioni del parametro corrispondente (ad esempio temperatura o precipitazioni) effettuate durante la giornata, con una periodicità minima di 5 minuti (che potrebbe variare in intervalli di 5, 10, 15, 20, 30 minuti, ecc.). La tipologia del file (parametro misurato) e il giorno di riferimento possono essere identificati sia dal nome del file stesso che all'interno del suo contenuto.

Specifiche

La pipeline da progettare e realizzare deve poter raccogliere i file, sia sulle precipitazioni che sulle temperature, archiviarli in un raw data storage, quindi estrarne i dati ed associarli a ciascuna specifica stazione meteo, in separate tabelle per pluvio, piuttosto che per temperatura.

I file ricevuti per essere considerati validi dovranno superare i seguenti controlli:

- Correttezza formato nome file
- Correttezza formato file (prime tre righe di intestazione, sono corrette? Il formato dei dati su ciascuna colonna è corretto? Es. prima colonna contiene un timestamp, seconda colonna contiene valori
- Corrispondenza dato nel file con dato nel nome del file (file pluvio contiene dati pluviometrici? Controllare sia intestazione riga che range valori presenti)

Durante la verifica della validità dei file bisogna tenere in considerazione che i valori negativi sono codici di errore, che in un primo passaggio di verifica di formattazione del file possono essere accettati come validi

Dopo aver superato i controlli, i file devono essere archiviati inalterati e caricati su un DB RAW data. Una volta disponibili i dati Raw, si effettuano due passaggi di correzione ed arricchimento:

Correzione:

- Si identificano i dati di errore (valore negativo) e quelli fuori scala e per ciascuno di essi si applica un algoritmo di correzione che sostituisce il valore mancante con l'ultimo valore valido ricevuto

Arricchimento:

- Sulla base di una mera tabella di lookup, si associa ogni stazione meteo ad una zona, intesa come rappresentanza di un gruppo di stazioni.

Successivamente si salvano tali dati su un DB dati validi

In seguito al salvataggio dei dati corretti e arricchiti segue un ultimo stadio di processing, che consiste nel generare dati aggregati come segue:

- Temperatura
    - Calcolare la temperatura media per stazione meteo
        - oraria (per ora piena, es. tra le 13:00 e le 13:55)
        - giornaliera (es. tra la 00:00 e le 23:55)
        - giornaliera dalle 9 (tra le 9:00 e le 8:55)
    - Calcolare la temperatura media di ciascun gruppo
- Precipitazioni
    - calcolare i valori cumulati (sommando) per stazione meteo
        - orario
        - giornaliero
        - giornaliero alle 9
    - Calcolare il valore di precipitazione cumulata per ciascun gruppo

Una volta terminata l'esecuzione della pipeline i dati aggregati generati dovranno essere visualizzati attraverso un BI tool ed inoltre dovrà essere inviata una mail ad un indirizzo prestabilito per informare l'utente della disponibilità dei dati aggiornati.

Analisi

Per poter implementare una pipeline in modo efficace, è utile porsi delle domande che guidino la fase di progettazione. Utilizzando tali domande è possibile tradurre le specifiche in uno scenario o, ancora meglio, verificare che le specifiche fornite siano sufficientemente dettagliate. Di seguito, viene riportato un elenco delle domande principali:

- **Obiettivi della pipeline**
  - Qual è l'obiettivo principale della pipeline che stiamo progettando?
  - Ci sono vincoli di tempo e budget? Se sì, quali?
  - Abbiamo bisogno di dati in tempo reale (ad esempio, dati in streaming) o è sufficiente un'elaborazione batch?
- **Informazioni sulle sorgenti dei dati**
  - Quali sono le sorgenti dati che devono essere inserite nella pipeline?
  - Qual è il volume dei dati che verrà processato dalla pipeline? Sarà costante nel tempo ?
  - Quali sono i formati e le strutture dei dati coinvolti (ad esempio, CSV, JSON, database)?
  - Ci sono requisiti di integrità dei dati che devono essere affrontati?
- **Trasformazioni**
  - Quali trasformazioni sono necessarie?
  - È necessario applicare regole aziendali specifiche o controlli di convalida?
  - Sono previste attività di arricchimento o di feature engineering?
- **Sistemi di Storage e Integrazione con altre piattaforme**
  - Dove saranno conservati i dati elaborati?
  - Che tipo di sistema di archiviazione verrà utilizzato?
  - Esistono sistemi o piattaforme di dati esistenti con cui la pipeline deve integrarsi?
- **Sicurezza e privacy dei dati**
  - Ci sono considerazioni sulla sicurezza o sulla privacy che devono essere affrontate?
  - Quali misure devono essere adottate per garantire la riservatezza e l'integrità dei dati?
  - Esistono requisiti di conformità (ad es. GDPR, HIPAA) che devono essere rispettati?

Durante il processo di analisi, risulta essenziale considerare l'opportunità di adottare un orchestratore. A tal fine, l'identificazione delle risposte alle seguenti domande può fornire una guida preziosa per prendere una decisione.

- **Orchestratore**
  - La pipeline coinvolge più componenti o flussi di lavoro interconnessi che devono essere coordinati?
  - C’è bisogno di definire l'ordine di esecuzione e le dipendenze tra i vari task o processi della pipeline?
  - È necessaria una gestione automatica degli errori, dei tentativi e del ripristino in caso di guasti o incongruenze dei dati?
  - È necessario il monitoraggio in tempo reale dell'esecuzione della pipeline?
  - La pipeline prevede attività di lunga durata o di elaborazione batch che devono essere gestite?
  - La pipeline si affida a servizi o API esterni che richiedono coordinamento e gestione?

Analisi della Pipeline

Sulla base delle domande poste e delle specifiche descritte precedentemente, è possibile definire la seguente pipeline.

La pipeline da progettare dovrà gestire uno scenario Batch, in quanto i dati, che sono il risultato dell'aggregazione di tutte le stazioni, saranno disponibili alla fine di ogni giornata tramite dei file CSV. Considerando il volume dei dati giornalieri, (185 stazioni che effettuano misurazioni con una frequenza minima di 5 minuti) la pipeline rientra in uno scenario Small Data con processamento Batch come detto in precedenza.

I dati ricevuti dovranno essere validati al fine di essere certi dell'esatta tipologia di file che si sta trattando; dopodiché, i dati dovranno essere trasformati e arricchiti come da specifiche ed infine dovranno essere generati dei dati aggregati utili alla comprensione dei dati stessi. Al termine di ogni processo di trasformazione o processamento è previsto che i dati vengano conservati in un database.

Sul tema della sicurezza e privacy dei dati non sono emerse grosse criticità dalle specifiche fornite, motivo per cui non è stata predisposta alcuna misura a riguardo. Inoltre, nel caso oggetto di studio non è prevista l'integrazione della pipeline con sistemi o piattaforme di dati esistenti e, nonostante non sia necessaria l'adozione di un orchestratore per la seguente pipeline, è stato comunque deciso di utilizzarne uno.

Dalla descrizione effettuata e dalle specifiche fornite è adesso possibile rappresentare quello che sarà il design della pipeline

Cos'è il Data Flow Diagram (DFD)

Il Data Flow Diagram (DFD) è una notazione grafica molto usata per i sistemi informativi e per la descrizione del flusso di dati in quanto permette di descrivere un sistema per livelli di astrazione decrescenti con una notazione di specifica molto "intuitiva".

Attraverso i Data Flow Diagram si definiscono soprattutto come fluiscono (e vengono elaborate) le informazioni all'interno del sistema; quindi, l'oggetto principale è il flusso delle informazioni o, per meglio dire, dei dati. Motivo per il quale diventa fondamentale capire dove sono immagazzinati i dati, da che fonte provengono, su quale fonte arrivano, quali componenti del sistema li elaborano.

Le componenti di questo tipo di diagramma sono:

- **Funzioni o processi**, rappresentate da cerchi;
- **Flussi di dati**, rappresentati da frecce;
- **Archivi di dati**, rappresentati da rettangoli aperti;
- **Agenti esterni o Input/Output di dati**, rappresentati da rettangoli.

Tuttavia, la rappresentazione delle componenti varia a seconda della convenzione utilizzata, in allegato le due convenzioni maggiormente utilizzate

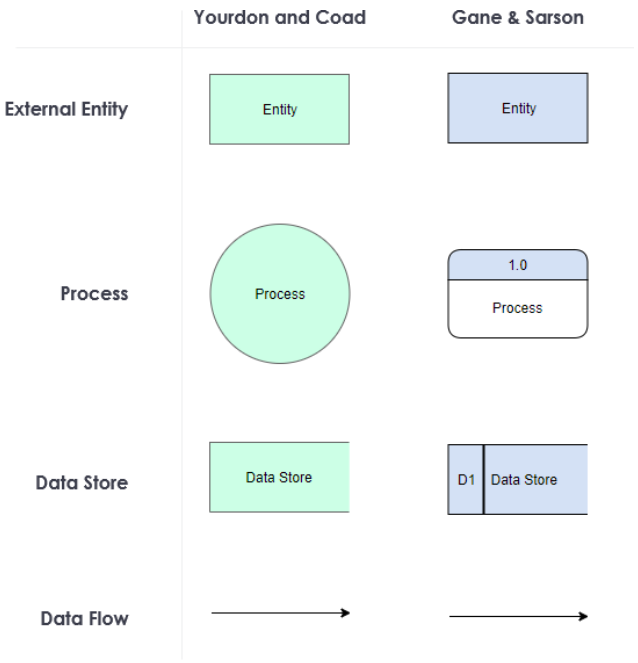


Figura 1 Convezioni Utilizzate per rappresentare DFD

Data Flow Diagram Pipeline Pluvio-Termo

Per rappresentare una versione semplificata della pipeline verrà utilizzato il Data Flow Diagram adottando la notazione di Yourdon e Coad

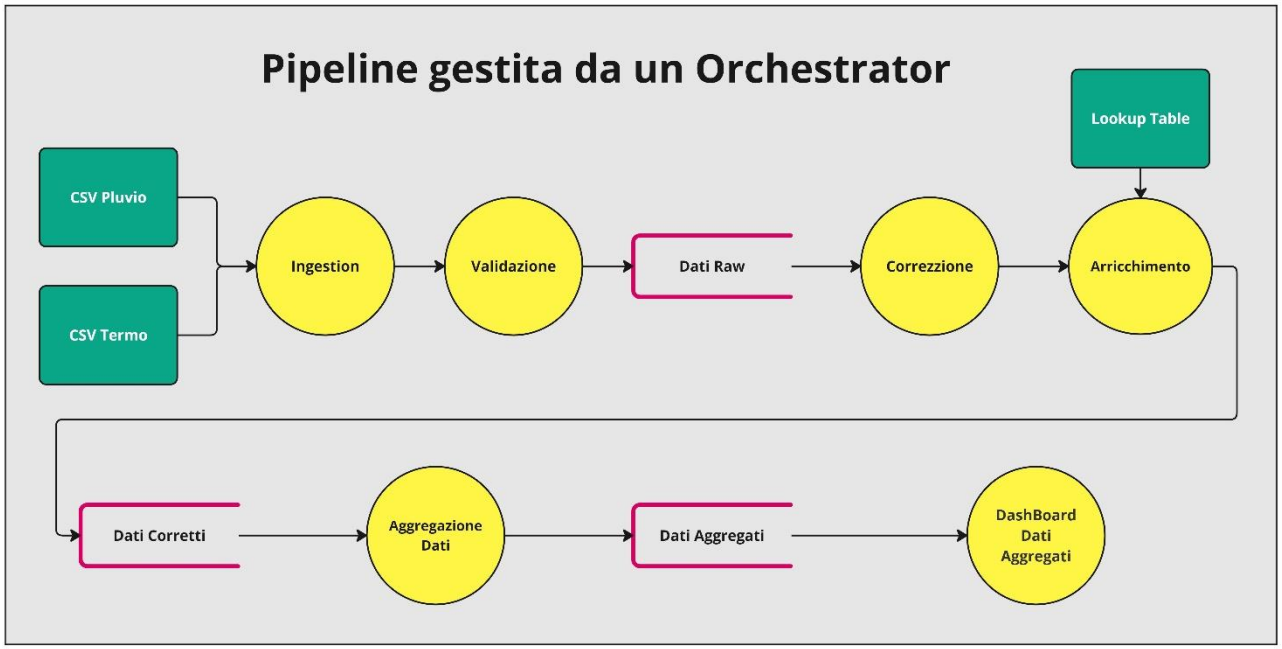
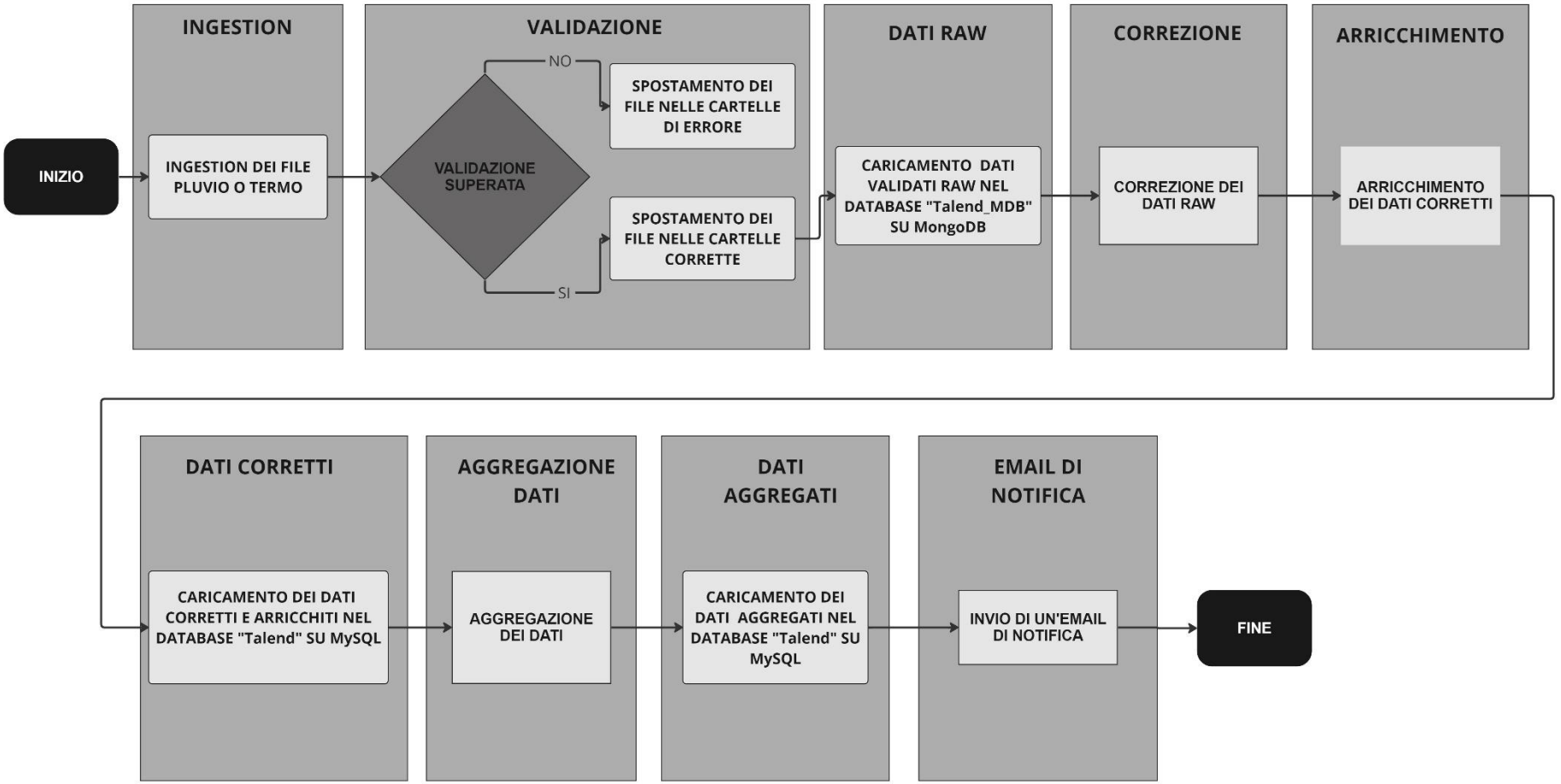


Figura 2 DFD della Pipeline Pluvio-Termo

Diagramma di Flusso della Pipeline Pluvio-Termo

Per avere una visione più dettagliata della pipeline verrà utilizzato il diagramma di flusso di seguito riportato.



Progettazione

Dopo aver effettuato l'analisi della pipeline, è possibile passare alla fase di progettazione, descrivendo ogni componente presente nel Data Flow Diagram.

Ingestion

Avendo a disposizione i dati di un intero anno, questo processo ha il compito di simulare l'ingestion giornaliera dei dati. Ad ogni avvio della pipeline, il processo sposterà in ordine cronologico un file "Pluvio" e un file "Termo" dalla loro cartella di partenza "Dati_ETL" alla cartella di destinazione "Dati Giornalieri". Considerando la semplicità di questo processo, sarà implementato utilizzando uno script Python.

Validazione

Il seguente processo dovrà validare i dati che sono stati aggiunti, dal processo precedente "Ingestion", nella cartella "Dati Giornalieri". La validazione consiste nella verifica della:

- Correttezza del formato del nome del file
- Correttezza del formato interno del file (prime tre righe di intestazione, sono corrette? Il formato dei dati su ciascuna colonna è corretto? Es. prima colonna contiene un timestamp, seconda colonna contiene valori
- Corrispondenza del dato contenuto nel file con il nome del file (file pluvio contiene dati pluviometrici? Controllare sia intestazione riga che range valori presenti)

Quindi:

- se corretti, la struttura dei dati, prima di essere posti nella cartella di destinazione, dovrà essere modificata, effettuando un Pivot da Colonne a Righe per ogni Stazione, ottenendo così la seguente struttura per i file Pluvio e Termo:

Data	Rainfall	Stazione
04/05/2022	5	Scicli

Data	Temperature	Stazione
04/05/2022	26	Scicli

  In seguito, andranno spostati nella cartella "Dati Giornalieri Output" e posti in una sottocartella "Pluvio" o "Termo" in base al contenuto del file;
- viceversa, andranno comunque spostati nella cartella "Dati Giornalieri Output" ma posti in una sottocartella diversa a seconda della tipologia errore:
  - Nome del file
  - Header
  - Range valori

Per implementare questo processo, sintetizzato con il seguente diagramma di flusso, si è scelto di utilizzare uno Script Python

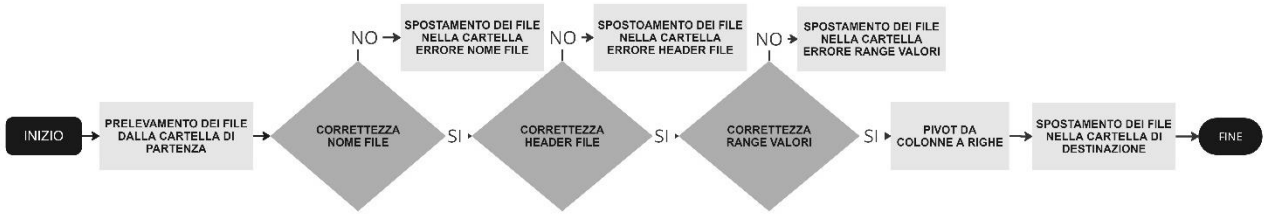


Figura 3 Diagramma di Flusso del processo di Validazione

Dati Raw

Il componente “Dati Raw” presente nel Data Flow Diagram rappresenta un sistema di archiviazione dati e quello che si è scelto di utilizzare in questa fase è il database NoSQL MongoDB. Il modo in cui i file Pluvio e Termo verranno caricati sul database sarà differente, poiché, per i file Pluvio verrà utilizzato un servizio web in locale fruibile tramite le Rest Api, invece per i file Termo, si è scelto di utilizzare il software Talend Open Studio for Big Data.

Caricamento Dati Raw Pluvio su MongoDB con Rest API

Per caricare i dati Raw Pluvio su MongoDB, si è scelto di utilizzare un servizio web in locale fruibile tramite le Rest API, che:

- prende in input, tramite il verbo GET del metodo http, la posizione del file da caricare
- carica i dati nella collection chiamata “Pluvio_Raw” del database “Talend_MDB” su MongoDB;
- elimina il file presente nella cartella di input.

Inoltre, il servizio mette a disposizione un’API endpoint che dà la possibilità di interrogare il processo riguardo lo stato di avanzamento, tramite il comando GET “status”.

Caricamento Dati Raw Termo su MongoDB con Talend Open Studio for Big Data.

Per caricare i dati Raw Termo su MongoDB, il software Talend Open Studio for Big Data, preleverà i dati dalla sottocartella “Termo” presente nella cartella “Dati Giornalieri Output”, li caricherà nella collection chiamata “Termo_Raw” del database “Talend_MDB” su MongoDB, ed in seguito eliminerà il file presente nella cartella di partenza.

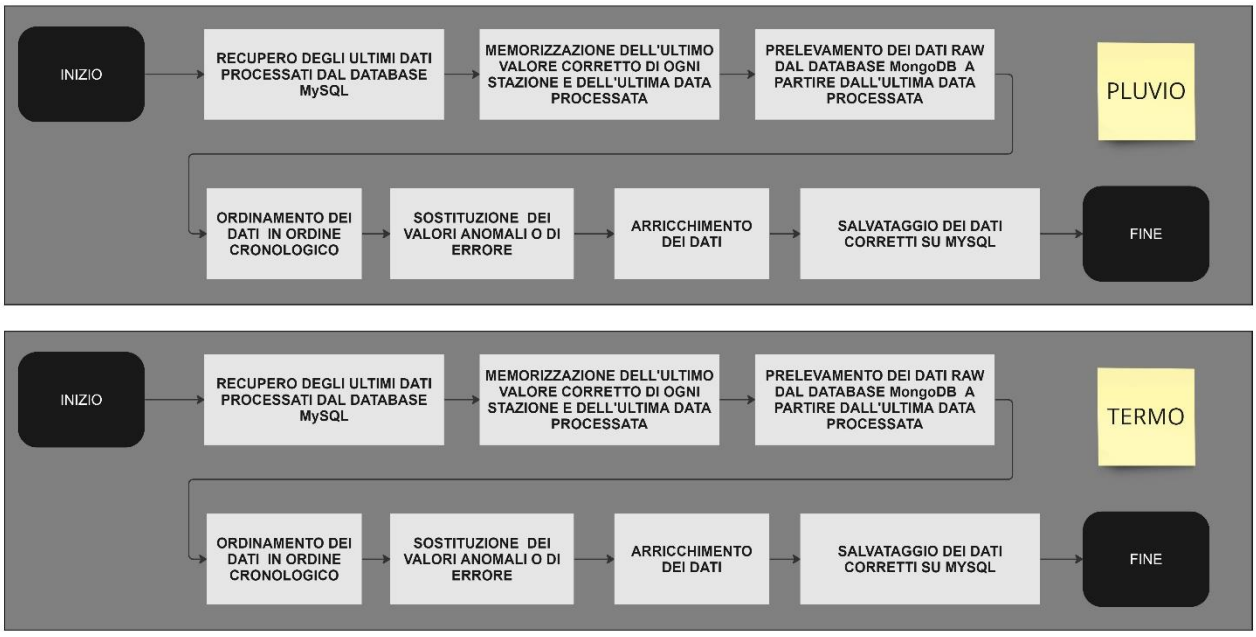
Correzione, Arricchimento e Caricamento Dati Corretti

Correzione e Arricchimento nel Data Flow Diagram sono rappresentati come due processi separati ma per semplificare il tutto, si è scelto di implementarli come un unico processo utilizzando il software Talend Open Studio for Big Data. In questa fase, i dati presenti nel database “Talend_MDB” su MongoDB dovranno essere:

- corretti: andando a sostituire i valori fuori scala e quelli di errore con l’ultimo valore valido ricevuto
- arricchiti: aggiungendo tramite una tabella di lookup l’informazione della zona relativa alla località dove risiede la stazione

Infine, i dati processati andranno caricati nel database “Talend” su MySQL nelle tabelle “Pluvio” o “Termo” in base alla tipologia dei dati. Inoltre, si è scelto di dividere il processo in due distinti processi sulla base della tipologia di file da processare (Pluvio o Termo).

Di seguito troviamo il diagramma di flusso che sintetizza il flusso di lavoro Talend.



Aggregazione Dati e Caricamento su MySQL

Il suddetto processo riveste un ruolo fondamentale all'interno della pipeline, poiché è responsabile della produzione dei dati aggregati di rilevanza per l'utente finale. Il processo avrà il compito di effettuare delle query al database "Talend" su MySQL producendo i seguenti dati aggregati:

- **Temperatura**
  - Calcolare la temperatura media per stazione meteo
    - oraria (per ora piena, es. tra le 13:00 e le 13:55)
    - giornaliera (es. tra la 00:00 e le 23:55)
    - giornaliera dalle 9 (tra le 9:00 e le 8:55)
  - Calcolare la temperatura media di ciascun gruppo
- **Precipitazioni**
  - calcolare i valori cumulati (sommando) per stazione meteo
    - orario
    - giornaliero
    - giornaliero alle 9
  - Calcolare il valore di precipitazione cumulata per ciascun gruppo

Dopo aver effettuato l'aggregazione, i dati verranno salvati nel database "Talend" su MySQL in differenti tabelle relative alle diverse aggregazioni. Anche in questo caso il processo verrà implementato utilizzando Talend Open Studio for Big Data e, diviso in due distinti processi sulla base della tipologia di file da processare (Pluvio o Termo). In conclusione verrà inviata una email ad un indirizzo prestabilito per informare l'utente finale della disponibilità dei dati aggiornati

DashBoard Dati Aggregati

Per visualizzare i dati aggregati prodotti si è scelto di utilizzare uno dei più famosi BI tool, Tableau. Tableau è una piattaforma di business intelligence che ci consente di effettuare un'analisi end-to-end dei dati, visualizzando i dati con l'ausilio delle Dashboards

Orchestratore

Come detto in fase di analisi, i processi della pipeline dovranno essere orchestrati, l'orchestratore selezionato è Apache Airflow.

Apache Airflow

Apache Airflow è una soluzione Open source per la definizione, schedulazione, esecuzione e monitoraggio di workflow di integrazione dati. Il core dell'applicazione consente la scrittura di componenti software che vengono orchestrati ed eseguiti secondo uno schema a grafo (DAG). La piattaforma è dotata di molte integrazioni ready-to-use per una più immediata ed estesa copertura di tutte le esigenze di un processo aziendale. Il prodotto è mantenuto ed evoluto da Apache Software Foundation.

Come funziona Apache Airflow?

Le pipeline sono descritte con l'uso di elementi fondamentali di Apache:

DAG

La pietra angolare della tecnologia è descritta dai grafi aciclici diretti (DAG). Questo modello è un grafo che non presenta cicli, ma percorsi paralleli provenienti dallo stesso batch. In parole povere, il DAG è un'entità che combina le attività a seconda della pipeline di dati, dove la dipendenza tra le applicazioni si manifesta chiaramente.

Operatore

Un operatore è un elemento separato della catena di task (pipeline). Utilizzando questi elementi, gli sviluppatori descrivono quale compito deve essere eseguito. Apache Airflow ha un elenco di operatori predefiniti che includono:

- PythonOperator esegue il codice Python
- BashOperator esegue gli script/comandi di bash
- PostgresOperator chiama le query SQL in PostgreSQL
- EmailOperator invia le e-mail

Sensore

Il sensore è una variante di un operatore che trova applicazione nelle pipeline event-driven. Esempi:

- PythonSensor attende che la funzione restituisca Vero
- S3Sensor verifica la disponibilità dell'oggetto in base alla chiave nel bucket S3.

Scheduler

Monitora tutti i DAG, gestisce i flussi di lavoro e invia i lavori a Executor.

WebServer

Il webserver svolge il ruolo di interfaccia utente di Apache Airflow. Aiuta a monitorare lo stato e l'avanzamento dei compiti e a registrare i dati dai depositi remoti.

Database

Tutte le informazioni pertinenti sono memorizzate lì (attività, periodi di pianificazione, statistiche di ogni sprint, ecc.)

Executor

L'Executor esegue i compiti e li invia ai lavoratori.

Infine, dimostriamo come funziona Apache con un semplice esempio. Innanzitutto, Apache rivede tutti i DAG in background. Le attività urgenti che devono essere completate ricevono il marchio SCHEDULED nel database. Lo Scheduler recupera i task dal database e li distribuisce agli Executor. Successivamente, i compiti ricevono lo stato QUEUED e, una volta che i lavoratori iniziano a eseguirli, viene assegnato lo stato RUNNING al lavoro. Quando l'attività è completata, il lavoratore la indica come conclusa/fallita, a seconda del successo del risultato finale, e lo Scheduler aggiorna lo stato nel database.

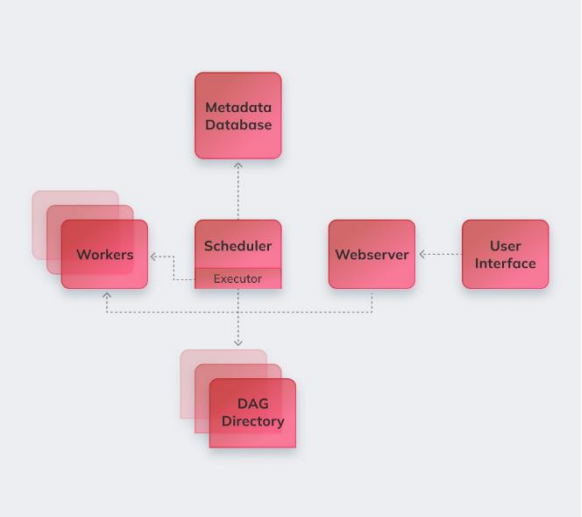
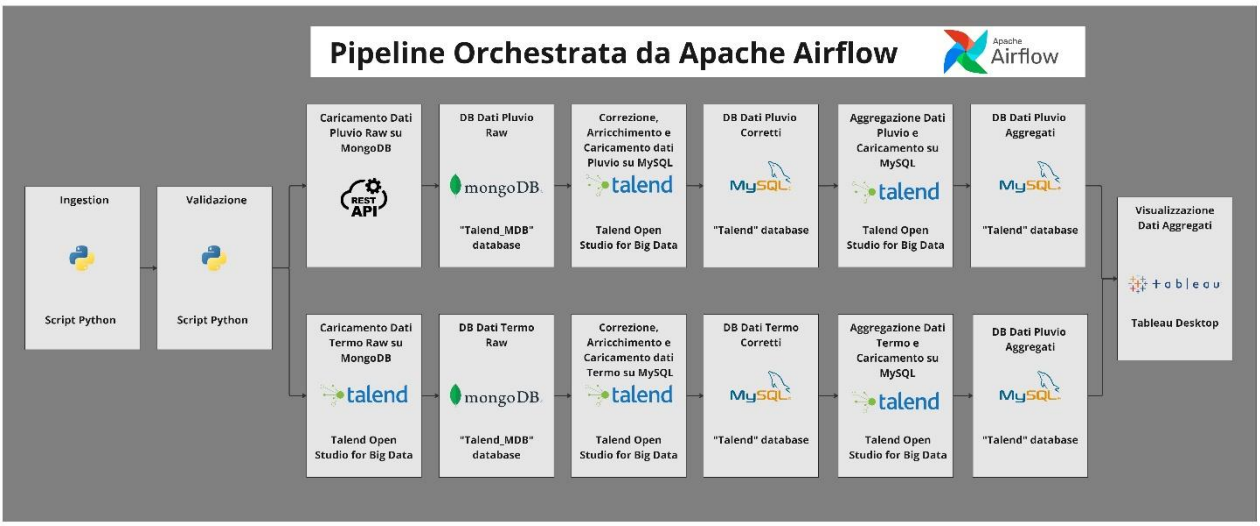


Diagramma Architetturale

Il diagramma architetturale mostra le rappresentazioni visive dei componenti di un sistema software. In un sistema software, il termine architettura si riferisce alle varie funzioni, alle loro implementazioni e alle loro interazioni reciproche. Di seguito, il diagramma architetturale della pipeline progettata.

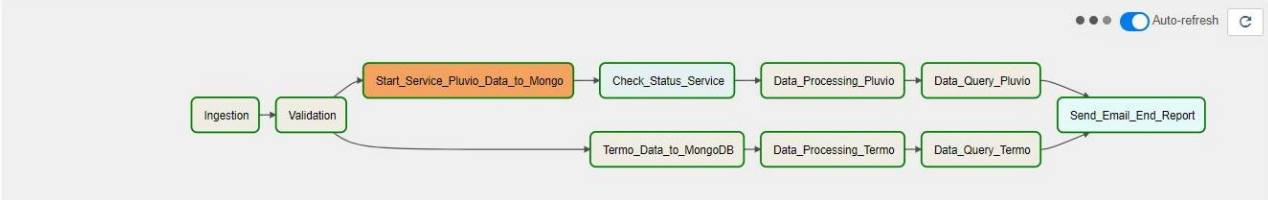


Implementazione

Questo parte della documentazione ha il compito di mostrare come effettivamente la pipeline è stata implementata. Di seguito si riporta la DAG prodotta con Airflow che ha lo scopo di fornire una guida durante la descrizione dell'implementazione, dato che Airflow si occuperà della gestione ed esecuzione dei processi.

I vari processi su Airflow prendono il nome di task e, in base alla tipologia del task da eseguire, Airflow dispone di diversi tipi di esecutori chiamati Operator. Per ogni task verrà descritta l'implementazione nativa del processo e, come questo è stato integrato con Airflow.

Ecco il link GitHub ai codici sorgente utilizzati per l'implementazione.



Come si evince dal Grafo, i tasks successivi al task Validation se pur simili tra loro, sono stati divisi in due rami sulla base del dato da processare ("Pluvio" o "Termo"), al fine di rendere l'implementazione più chiara e semplice.

Airflow

Airflow permette di monitorare la pipeline in ogni istante e catturare i log di ogni task; inoltre, prima di assegnare la label "failed" alla pipeline al verificarsi di un errore di un task, Airflow consente di rieseguirlo ed è stato programmato per farlo una sola volta dopo 30 minuti.

La versione utilizzata è la 2.6.1 installata in locale utilizzando Docker. Di seguito le linee guida principali dell'installazione:

- Installare Docker Community Edition (CE) sulla workstation
- Installare Docker Compose
- Recuperare il file docker-compose.yaml dal sito ufficiale di Airflow per poter configurare l'ambiente e scaricare il software
- Inizializzare l'ambiente
- Inizializzare il database
- Lanciare Airflow

Per maggiori informazioni ecco il link alla guida ufficiale di Airflow.

Task Ingestion

Questo task ha l'obiettivo di simulare l'Ingestion giornalieri dei file csv Pluvio e Termo spostando i dati dalla cartella "datiETL" contenente tutti i dati giornalieri raccolti, alla cartella "Dati_Giornaieri".

Implementazione del Processo

Il processo implementato con uno script Python "ingestion.py", versione 3.11.3, richiede in input:

- Il Path della directory di partenza ("DatiETL") dei file
- Il Path della directory di destinazione (Dati_Giornalieri") dei file

La directory di partenza contiene due sottocartelle: "Pluvio" e "Termo", al cui interno sono presenti i file raccolti in un anno. Lo script leggerà i file presenti all'interno delle due cartelle, li ordinerà per data e sposterà una copia di un solo file "Pluvio" ed un solo file "Termo", cronologicamente meno recente, alla directory di

destinazione. Dopodiché, i file che sono stati copiati verranno rinominati come ". uploaded" al fine di evitare di copiarli nuovamente.

Integrazione con Airflow

Il seguente script verrà eseguito da Airflow mediante l'operatore BashOperator, che lo eseguirà da bash chiamando l'interprete Python e fornendo il path dello script e le variabili richieste.

```
Ingestion = BashOperator(
    task_id='Ingestion',
    bash_command="python "+python_ingestion+" "+input_path_ingestion+" "+output_path_ingestion,
    cwd=dag.folder)
```

Task Validation

Questo task ha l'obiettivo di validare i file csv Pluvio e Termo presenti nella cartella "Dati_Giornalieri", in seguito alla validazione i file verranno spostati nelle opportune sottocartelle di errore o di output dentro la cartella "Dati_Giornalieri_Output".

Implementazione del Processo

Il processo implementato con uno script Python "preprocessing.py", versione 3.11.3, richiede in input:

- Il Path della directory di partenza ("Dati_Giornalieri") dei file
- Il Path della directory di destinazione ("Dati_Giornalieri_Output") dei file

Lo script esegue la stessa procedura validazione alle due tipologie di file, validando prima i file Pluvio poi quelli Termo come segue:

- Innanzitutto, viene verificato che la struttura del nome del file sia la seguente tramite le Regular Expressions(regex):
    - DatiPluvio_2015-01-07_00-00.csv
    - DatiTermo_2015-05-07_00-00.csv

  se non presenti errori, il processo continua la validazione del file corrente, altrimenti il file viene spostato nella cartella "Errore_Nome_File".
- Dopodichè, dopo aver convertito il file in un pandas DataFrame, viene verificato che le tre righe di intestazione siano uguali a quelle di default relative alle tipologie del file (Pluvio o Termo), nel caso sotto sono state riportare le righe di intestazione di default dei file Pluvio.

	ACIREALE	AGIRA	AGRIGENTO	AIDONE	ALCAMO	ALIA	ALIMENA	ANTILLO	ARAGONA	ARANCIO DIGA
Location Ids	ACIREALE	AGIRA	AGRIGENTO	AIDONE	ALCAMO	ALIA	ALIMENA	ANTILLO	ARAGONA	ARANCIO DIGA
Time	Rainfall	Rainfall	Rainfall	Rainfall	Rainfall	Rainfall	Rainfall	Rainfall	Rainfall	Rainfall

  se queste non risultano uguali il file viene spostato nella cartella "Errore_Header", altrimenti la procedura di verifica continua.

- L'ultimo step di verifica consiste nel calcolare il range dei valori per ogni stazione all'interno del file escludendo i valori di errore, se il range risulta essere tra 0 e 200 per i file Pluvio o tra -40 e 60 per i file Termo, il file risulta validato correttamente, altrimenti verrà spostato nella cartella "Errore_Range_Values".

La struttura dei file validati corretti, sarà modificata effettuando un pivot da Colonne a Righe delle stazioni, passando quindi dal seguente schema:

Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9	Column10	Column11
	ACIREALE	AGIRA	AGRIGENTO	AIDONE	ALCAMO	ALIA	ALIMENA	ANTILLO	ARAGONA	ARANCIO DIGA
Location Ids	ACIREALE	AGIRA	AGRIGENTO	AIDONE	ALCAMO	ALIA	ALIMENA	ANTILLO	ARAGONA	ARANCIO DIGA
Time	Rainfall	Rainfall	Rainfall	Rainfall	Rainfall	Rainfall	Rainfall	Rainfall	Rainfall	Rainfall
2015-01-04 00:00:00	0	0	0	0	0	0	0	0	-999.8	0
2015-01-04 00:05:00	-999.0	-999.0	-999.0	-999.0	-999.0	-999.0	-999.0	-999.0	-999.0	-999.0

Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9
	ACATE_Mogli	ACIREALE	AGIRA	AGIRA_Mangiagrilli	AGRIGENTO	AGRIGENTO_Mandrascava	AGRIGENTO_Scibica	AIDONE
Location Ids	ACATE_Mogli	ACIREALE	AGIRA	AGIRA_Mangiagrilli	AGRIGENTO	AGRIGENTO_Mandrascava	AGRIGENTO_Scibica	AIDONE
Time	Temperature	Temperature	Temperature	Temperature	Temperature	Temperature	Temperature	Temperatu
2015-05-04 00:00:00	16.4	19.9	18.4	18.3	20.7	16.6	15.9	17.5
2015-05-04 00:05:00	16.4	-999.0	-999.0	17.4	-999.0	16.4	15.9	-999.0

Al seguente:

Date	Rainfall	Station
2015-01-04 00:00:00	0	ACIREALE
2015-01-04 00:00:00	0	AGIRA

Date	Temperature	Station
2015-01-04 00:00:00	16.4	ACIREALE
2015-01-04 00:00:00	19.9	AGIRA

Infine, lo script salverà i file modificati nella directory di destinazione “Dati_Giornalieri_Output” ed eliminerà i file nella directory di partenza “Dati_Giornalieri”

Integrazione con Airflow

Lo script verrà eseguito da Airflow utilizzando l’operatore BashOperator, che lo eseguirà da bash chiamando l’interprete Python e fornendo il path dello script e le variabili richieste.

```
Validation = BashOperator(
        task_id='Validation',
        bash_command="python "+python_validation+" "+input_path_validation+" "+output_path_validation,
        cwd=dag.folder
)
```

Task Start_Service_Pluvio_Data_to_Mongo e Check_Status_Service

Questi due task sono legati tra loro poiché, il primo si occuperà di avviare il processo di caricamento dati Pluvio Raw su MongoDB tramite le Rest API fornite, mentre il secondo, controllerà sempre attraverso le Rest Api lo stato del processo di caricamento ad intervalli regolari.

Descrizione del Processo

Il seguente processo utilizza un servizio web in locale fruibile tramite Rest API, all'indirizzo URL base http://localhost:9088/ , per caricare i dati Pluvio Raw nella collection "Pluvio_Raw" del database "Talend_MDB" su MongoDB, ogni documento della collection possiede le seguenti keys :

- "Date"
- "Rainfall"
- "Station"

Le Rest API fornite dispongono di due endpoint accessibili tramite il metodo GET del protocollo HTTP:

- "**http://localhost:9088/activate/" folder_path_file_pluvio"** consente di avviare il processo di caricamento dei dati, richiedendo in input il path della cartella dove risiedono i file pluvio. Inoltre, dopo aver terminato il processo il servizio eliminerà i file presenti nella cartella di partenza. Questo endpoint verrà chiamato dal task **Start_Service_Pluvio_Data_to_Mongo**
- "**http://localhost:9088/status/** consente di monitorare lo stato del processo, infatti, la chiamata http di questo endpoint ritornerà uno dei seguenti stati:
  - "Activated"
  - "Running"
  - "Finished"

  Questo endpoint verrà chiamato dal **task Check_Status_Service** ogni 15 secondi per controllare lo stato del caricamento dei dati, non appena lo stato risulterà "finished" il task risulterà eseguito con successo e potranno essere eseguiti i successivi task.

Integrazione Task Start_Service_Pluvio_Data_to_Mongo con Airflow

L'endpoint "**http://localhost:9088/activate/**" verrà chiamato da Airflow utilizzando l'operatore SimpleHttpOperator che consente di effettuare chiamate HTTP.

```
command_Api ="activate/C:--Users--pepee--Desktop--Tirocinio--Airflow--dags--JobTalend--Dati--Dati_Giornalieri_Output--Pluvio"
request_Api = SimpleHttpOperator(
    task_id='Start_Service_Pluvio_Data_to_Mongo',
    http_conn_id='web_api',
    method='GET',
    endpoint=command_Api,
    response_check=lambda response: True if check(response.status_code) is True else False,
    )
```

Integrazione Task Check_Status_Service con Airflow

L'endpoint "**http://localhost:9088//status/** verrà chiamato da Airflow utilizzando il sensore HttpSensor. Questo componente blocca il flusso di lavoro di Airflow, effettuando una chiamata http ad intervalli regolari (in questo caso ogni 15 secondi) finché una condizione non si verifichi. La condizione scelta è che la risposta della chiamata http contenga la parola "finished" che indica la fine del caricamento dei dati su MongoDB

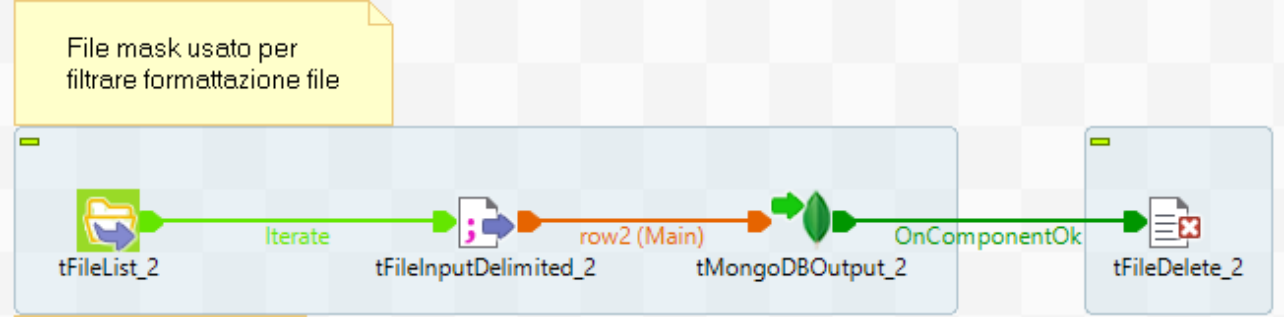
```
check_status_service = HttpSensor(

    task_id="Check_Status_Service",
    http_conn_id="web_api",
    method="GET",
    endpoint="status/12",
    response_check=lambda response: "finished" in response.text,
    poke_interval=15

    )
```

Task Termo_Data_to_Mongo

Il seguente Task carica i dati Termo Raw presenti nella sottocartella “Termo” dentro la cartella “Dati_Giornalieri_Output” nella collection “Termo_Raw” del database “Talend_MDB” su MongoDB.

Descrizione del Processo

A differenza degli altri, questo processo è stato implementato utilizzando il software Talend Open Studio for Big Data versione 8.0.1. Il seguente schema mostra in maniera esplicita il funzionamento del processo, nel dettaglio:



- Il componente tFileList fornirà la lista di path dei file Termo Raw presenti nella sottocartella “Termo” dentro la cartella “Dati_Giornalieri_Output”;
- I path verranno passati al componente tFileInputDelimited, che aprirà i file uno per volta e passerà il contenuto riga per riga al componente successivo
- Il componente tMongoDB leggerà le righe in input e le caricherà nella collection “Termo_Raw” del database “Talend_MDB” presente su MongoDB con le seguenti keys per ogni document:
    - “Date”
    - “Temperature”
    - “Station”

Alla fine del caricamento di ogni file su MongoDB il file presente nella cartella di partenza verrà eliminato.

Integrazione Termo_Data_to_Mongo con Airflow

Per poter integrare il processo con Airflow è stato necessario effettuare la build del progetto Talend, convertendo il processo in un eseguibile bash. L’eseguibile bash al suo interno lancerà il processo eseguendo un codice java.

L’eseguibile verrà lanciato da Airflow utilizzando l’operatore BashOperator, fornendo il path dell’eseguibile ed il path della cartella di partenza dove risiedono i dati Termo Raw.

```
bash_command_termo=dag.folder+"/JobTalend/Talend/Termo_MDB/Caricamento_MDB_Termo/Caricamento_MDB_Termo_run.sh "
context_param_termo ="--context_param termo="+output_path_validation+"/Termo/"
Dati_Termo_to_MongoDB = BashOperator(
        task_id='Termo_Data_to_MongoDB',
        bash_command=bash_command_termo+context_param_termo,
        cwd=dag.folder
)
```

Task Data_Processing_Pluvio

Il task Data_Processing_Pluvio preleva i dati Pluvio Raw dalla collection “Pluvio_Raw” del database “Talend_MDB” su MongoDB, li corregge, li arricchisce ed in seguito li salva nella tabella “Pluvio” del database “Talend” su MySQL.

Implementazione del Processo

Il seguente processo è stato implementato utilizzando Talend Open Studio for Big Data versione 8.0.1. Il processo inizia con una query alla tabella “Pluvio” del database “Talend” su MySQL chiedendo la data degli ultimi dati processati, questa informazione verrà salvata in una variabile locale dal componete tJavaRow_1.

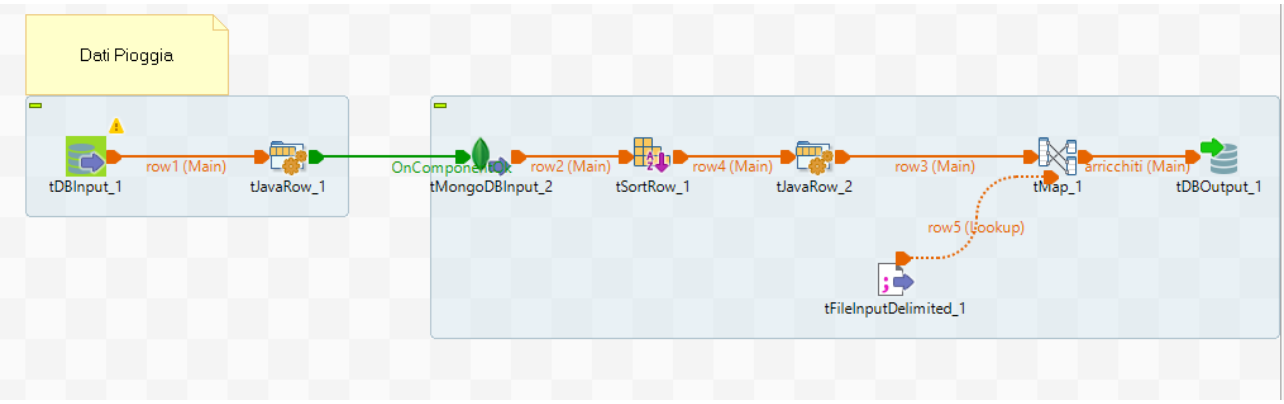
Successivamente, verrà effettuata una query alla collection “Pluvio_Raw” del database “Talend_MDB” su MongoDB, chiedendo tutti i dati Pluvio_Raw riferiti al periodo successivo alla data dell’ultimo processamento, i dati richiesti saranno ordinati per data, dal più recente al meno recente dal componente tSortRow_1.

Di seguito, i dati verranno trasmessi riga per riga al componete successivo, tJavaRow_2, che sostituirà i dati di errore o i valori fuori scala con l’ultimo valore valido salvato in memoria.

Dopodichè, i dati verranno arricchiti tramite un’operazione join, l’arricchimento consiste nell’aggiungere un’ulteriore colonna ai dati che contenga l’informazione della zona a cui appartiene la località dove risiede la stazione, l’informazione della zona è fornita tramite una tabella di lookup formata da due colonne :

Station	Zone

Di conseguenza l’operazione join sarà effettuata tra i dati Pluvio e la tabella di lookup.



Infine, i dati corretti e arricchiti verranno salvati nella tabella “Pluvio” del database “Talend” su MySQL con il seguente schema:

Date (Primary Key)	Station (Primary key)	Rainfall	Zone

Integrazione Data_Processing_Pluvio con Airflow

Per poter integrare il processo con Airflow è stato necessario effettuare la build del progetto Talend, convertendo il processo in un eseguibile bash. L’eseguibile bash al suo interno lancerà il processo eseguendo un codice java.

L’eseguibile verrà lanciato da Airflow utilizzando l’operatore BashOperator, fornendo il path dell’eseguibile ed il path della tabella di lookup.

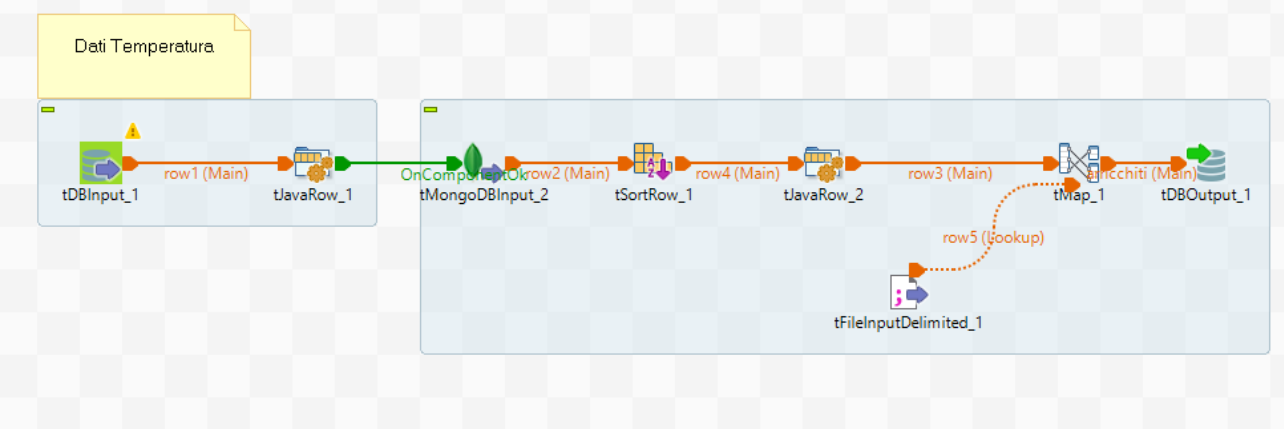
```
bash_command_pro_pluvio=dag.folder+"/JobTalend/Talend/Data_Pro_Pluvio/Data_Processing_Pluvio/Data_processing_Pluvio_run.sh "
context_param_pro_pluvio ="--context_param arr_pluvio="+output_path_validation+"/Lookup_Pluvio/lookup_pluvio.csv"
Data_Processing_Pluvio = BashOperator(
        task_id='Data_Processing_Pluvio',
        bash_command=bash_command_pro_pluvio+context_param_pro_pluvio,
        cwd=dag.folder
)
```

Task Data_Processing_Termo

Il task Data_Processing_Termo svolge le stesse operazioni del task Data_Processing_Pluvio le uniche differenze consistono nei database di input e output, poiché, i dati Termo Raw verranno prelevati dalla collection 'Termo_Raw" del database "Talend_MDB" su MongoDB, saranno corretti, arricchiti e salvati con la seguente struttura nella tabella "Termo" del database "Talend" su MySQL:

Date (Primary Key)	Station (Primary key)	Temperatura	Zone

Implementazione del Processo



Integrazione Data_Processing_Termo con Airflow

Per poter integrare il processo con Airflow è stato necessario effettuare la build del job Talend, convertendo il processo in un eseguibile bash. L'eseguibile bash al suo interno lancerà il processo eseguendo un codice java.

L'eseguibile verrà lanciato da Airflow utilizzando l'operatore BashOperator, fornendo il path dell'eseguibile ed il path della tabella di lookup.

```
bash_command_pro_termo=dag.folder+"/JobTalend/Talend/Data_Pro_Termo/Data_Processing_Termo/Data_processing_Termo_run.sh "
context_param_pro_termo ="--context_param arr_termo="+output_path_validation+"/Lookup_Termo/lookup_termo.csv"
Data_Processing_Termo = BashOperator(
        task_id='Data_Processing_Termo',
        bash_command=bash_command_pro_termo+context_param_pro_termo,
        cwd=dag.folder
)
```

Task Data_Query_Pluvio e Data_Query_Termo

Questi due task hanno il compito di aggregare i dati Pluvio e Termo effettuando le queries alle tabelle “Pluvio” e “Termo” del database “Talend” su MySQL

Implementazione dei Processi

Le aggregazioni dei dati Pluvio e quelle dei dati Termo sono state effettuate utilizzando due job Talend separati. Ecco le queries utilizzate per aggregare i dati Pluvio:

- Aggregazione oraria dalle xx:00 alle xx:55: l’output della query sarà salvato in una tabella “Pluvio_sum_hourly” del database “Talend” su MySQL

```
select Date,SUM(Rainfall),Station from pluvio
where minute(Date) <= 55
group by year(Date),month(Date),day(Date),hour(Date),Station
```

- Aggregazione giornaliera dalle 00:00 alle 23.55: l’output della query sarà salvato in una tabella “Pluvio_sum_daily” del database “Talend” su MySQL

```
select Date,SUM(Rainfall),Station from pluvio
where TIME(Date) between '00:00:00' AND'23:55:00'
group by  year(Date),month(Date),day(Date),Station
```

- Aggregazione giornaliera dalle 09:00 alle 08.55: l’output della query sarà salvato in una tabella “Pluvio_sum_daily_9” del database “Talend” su MySQL

```
select date_add(newdate, interval +9 hour) as Date,Rainfall,Station from
    (
    select newdate,SUM(Rainfall) as Rainfall,Station  from
        (
        select date_add(Date, interval -9 hour) as newdate,Rainfall,Station
        from pluvio
        ) as T
    where TIME(newdate) between '00:00:00' AND '23:55:00'
    group by  year(newDate),month(newDate),day(newdate),Station
    ) as T1;
```

. Ecco le queries utilizzate per aggregare i dati Termo:

- Aggregazione oraria dalle xx:00 alle xx:55: l'output della query sarà salvato in una tabella "Termo_avg_hourly" del database "Talend" su MySQL

```
select Date,AVG(Temperature),Station from Termo
where minute(Date) <= 55
group by year(Date),month(Date),day(Date),hour(Date),Station
```

- Aggregazione giornaliera dalle 00:00 alle 23.55: l'output della query sarà salvato in una tabella "Termo_avg_daily" del database "Talend" su MySQL

```
select Date,AVG(Temperature),Station from Termo
where TIME(Date) between '00:00:00' AND'23:55:00'
group by year(Date),month(Date),day(Date),Station
```

- Aggregazione giornaliera dalle 09:00 alle 08.55: l'output della query sarà salvato in una tabella "Termo_avg_daily_9" del database "Talend" su MySQL

```
select date_add(newdate, interval +9 hour) as Date,Temperature,Station from
    (
    select newdate,AVG(Temperature) as Temperature,Station  from
        (
        select date_add(Date, interval -9 hour) as newdate,Temperature,Station
        from Termo
        ) as T
    where TIME(newdate) between '00:00:00' AND '23:55:00'
    group by year(newdate),month(newdate),day(newdate),Station
    ) as T1;
```

Integrazione Data_Query_Pluvio e Data_Query_Termo con Airflow

Per poter integrare i due processi con Airflow è stato necessario effettuare la build dei jobs Talend, convertendo i processi in due eseguibili bash.

Gli eseguibili verranno lanciato da Airflow utilizzando l'operatore BashOperator.

```
bash_command_query_pluvio=dag.folder+"/JobTalend/Talend/Query_Pluvio/Query_Pluvio/Query_Pluvio_run.sh "
Data_Query_Pluvio = BashOperator(
        task_id='Data_Query_Pluvio',
        bash_command=bash_command_query_pluvio,
        cwd=dag.folder
)
```

```
bash_command_query_termo=dag.folder+"/JobTalend/Talend/Query_Termo/Query_Termo/Query_Termo_run.sh "
Data_Query_Termo = BashOperator(
        task_id='Data_Query_Termo',
        bash_command=bash_command_query_termo,
        cwd=dag.folder
)
```

Task Send_Email_End_Report

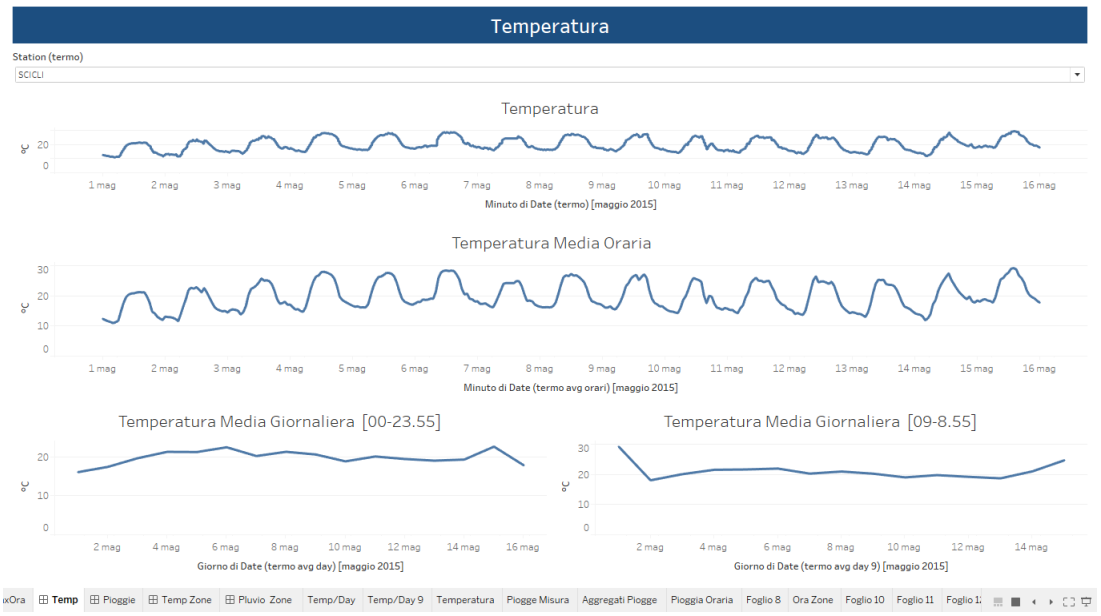
Questo ultimo task invierà una mail ad un indirizzo prestabilito per avvertire l'utente finale della disponibilità dei dati aggiornati. In questo caso il processo è stato implementato direttamente su Airflow tramite l'utilizzo dell'operatore EmailOperator.

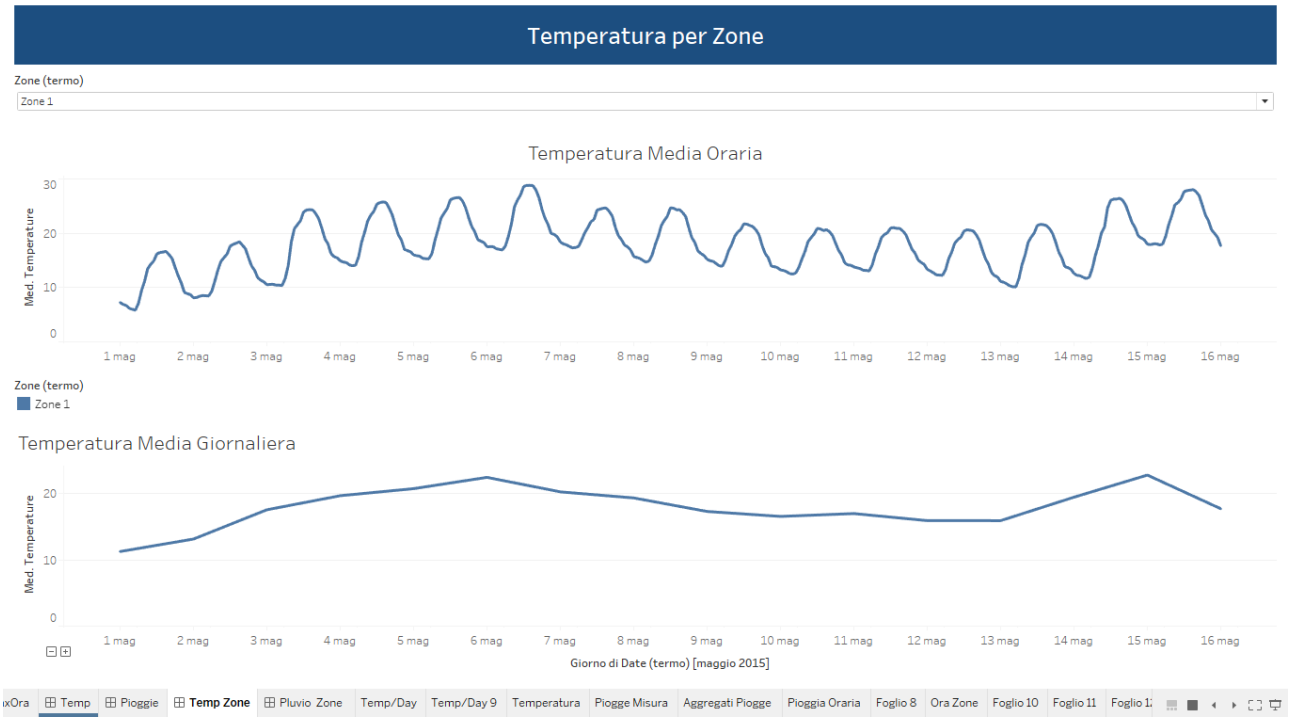
```
email = EmailOperator(
    task_id='Send_Email_End_Report',
    to='pulino918@gmail.com',
    subject='Airflow Alert',
    html_content=""" <h1>Pipeline Completata</h1> """
    )
```

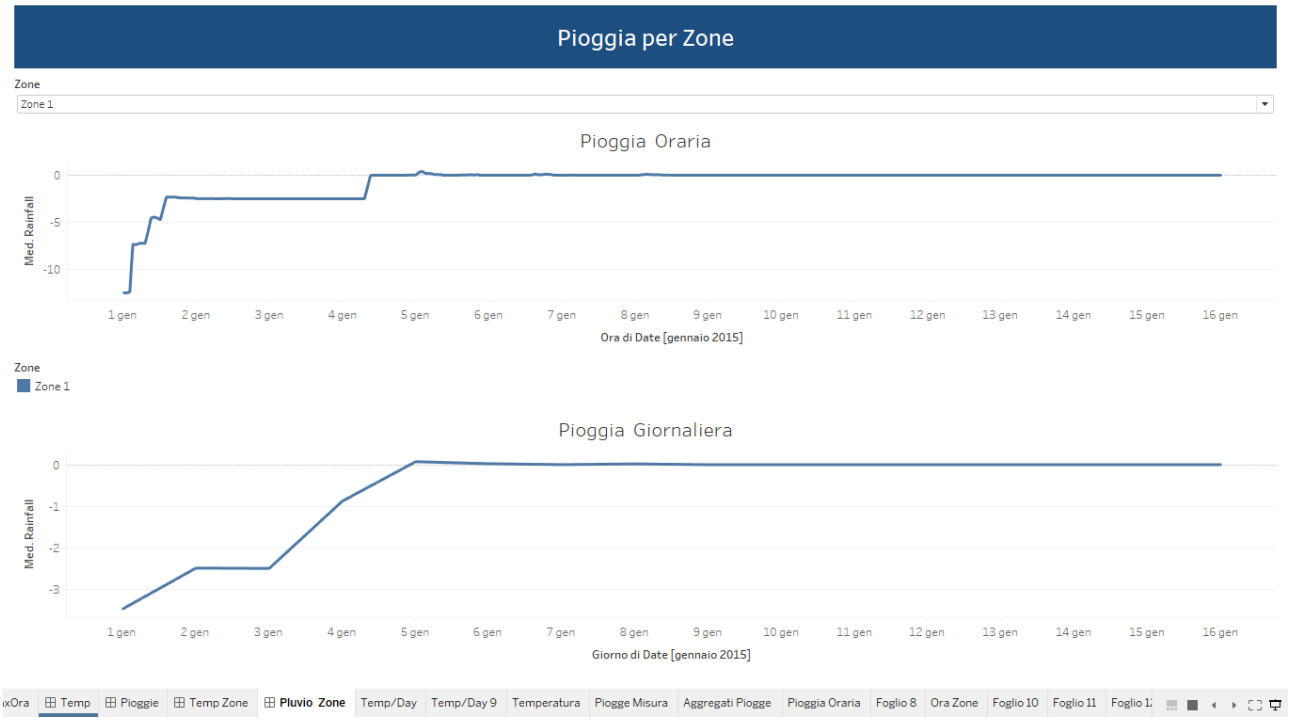
Dashboards Tableau

Per poter visualizzare i dati sono state create delle dashboards utilizzando Tableau Desktop.

Le Dashboards prelevano i dati direttamente dal database "Talend" presente su MySQL in modalità live; quindi, ogni volta che nuovi dati verranno aggiunti alle varie tabelle sul database MySQL questi verranno automaticamente inseriti nella visualizzazione. Inoltre, la visualizzazione dei dati potrà essere filtrata per Stazione o per Zona.




Temperatura per Zone
Zone (termo)
Zone 1
Temperatura Media Oraria
Med. Temperature
30
20
10
0
1 mag
2 mag
3 mag
4 mag
5 mag
6 mag
7 mag
8 mag
9 mag
10 mag
11 mag
12 mag
13 mag
14 mag
15 mag
16 mag
Zone (termo)
Zone 1
Temperatura Media Giornaliera
Med. Temperature
20
10
0
1 mag
2 mag
3 mag
4 mag
5 mag
6 mag
7 mag
8 mag
9 mag
10 mag
11 mag
12 mag
13 mag
14 mag
15 mag
16 mag
Giorno di Date (termo) [maggio 2015]
Ora
Temp
Pioggie
Temp Zone
Pluvio Zone
Temp/Day
Temp/Day 9
Temperatura
Piogge Misura
Aggregati Piogge
Pioggia Oraria
Foglio 8
Ora Zone
Foglio 10
Foglio 11


Pioggia per Zone
Zone
Zone 1
Pioggia Oraria
Med. Rainfall
0
-5
-10
1 gen
2 gen
3 gen
4 gen
5 gen
6 gen
7 gen
8 gen
9 gen
10 gen
11 gen
12 gen
13 gen
14 gen
15 gen
16 gen
Ora di Date [gennaio 2015]
Zone
Zone 1
Pioggia Giornaliera
Med. Rainfall
0
-1
-2
-3
1 gen
2 gen
3 gen
4 gen
5 gen
6 gen
7 gen
8 gen
9 gen
10 gen
11 gen
12 gen
13 gen
14 gen
15 gen
16 gen
Giorno di Date [gennaio 2015]
Ora
Temp
Pioggie
Temp Zone
Pluvio Zone
Temp/Day
Temp/Day 9
Temperatura
Piogge Misura
Aggregati Piogge
Pioggia Oraria
Foglio 8
Ora Zone
Foglio 10
Foglio 11