Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di laurea in Informatica

Data ingestion e anomaly detection: il ciclo di vita del dato

Tutor Interno
Prof. Lamberto Ballan
Università di Padova

Tutor Esterno
Dott. Michele Giusto

Studente Candidato
Giovanni Sorice

Anno Accademico 2018 - 2019

Alla mia famiglia.

Sommario

Il presente documento, descrive l'esperienza di stage svolta dal laureando Giovanni Sorice all'interno dell'azienda Data Reply S.r.l di Milano. Lo stage è stato svolto nei mesi di giugno e luglio 2018 ed è durato complessivamente 320 ore.
Presenterò quindi il lavoro svolto durante il progetto di stage che aveva al centro della sua attenzione la gestione dei dati, dal loro rilevamento al loro utilizzo come informazione e quindi bene prezioso per ogni azienda.
Il progetto, ha portato allo sviluppo di due prodotti che lavorano in sinergia per poter apportare il massimo valore aggiunto all'azienda cliente. I due prodotti si dividono prevalentemente per scopo e processamento dei dati, infatti il primo si occupa di acquisire, conservare e trasformare il dato in informazione, mentre il secondo punta a classificare i dati e a cercarne eventuali anomalie.

Ringraziamenti

Desidero innanzitutto ringraziare il prof. Lamberto Ballan per la disponibilità dimostrata in questi mesi di scrittura della presente tesi e il gran numero di consigli che mi ha dato.

Desidero ringraziare le persone che mi hanno aiutato durante il mio periodo di stage all'interno di Data Reply, tra cui il Dott. Michele Giusto e il Dott. Alessandro Celi per gli utili consigli.

Ringrazio i miei genitori che mi hanno permesso di raggiungere questo obiettivo e mi hanno dato forza nei momenti difficili.

Ringrazio, di cuore, mio fratello Domenico per il sostegno, gli insegnamenti e per avermi dato un modello a cui ispirarmi. Ringrazio Giulia per avermi aiutato nella creazione delle immagini.

Infine, ringrazio i miei amici per aver condiviso con me questa esperienza.

Castelfranco Veneto, Settembre 2019 Giovanni Sorice

Indice

Elenco delle figure

Elenco delle tabelle

1 Introduzione

In questo capitolo verrà introdotta l'azienda ospitante e le tecnologie da essa utilizzate.

1.1 L'azienda

Data Reply S.r.l fa parte delle aziende del gruppo Reply S.p.a e si occupa del mondo *big data*, *data science* e *artificial intelligence*. L'azienda è giovane sia nella sua creazione, dato che è stata fondata nel 2013, sia dal punto di vista della composizione del personale. Questo le permette di avere flessibilità e freschezza mentale (in termini di idee) ma allo stesso tempo *know how* e conoscenza del settore messa in campo dagli elementi con maggiore esperienza.

Il mercato di Data Reply è quello della consulenza, un mondo spesso ostico e molto complesso ma che ha visto un'importante crescita degli investimenti in Italia negli ultimi anni.

Figura 1.1: Logo Data Reply S.r.l.

1.1.1 Le tecnologie da esplorare

Le principali tecnologie utilizzate dall'azienda riguardano i settori *big data*, *Data Science* e *Artificial Intelligence*. Troviamo un importante utilizzo delle suite di prodotti cloud, come Amazon Web Services (AWS) e Google Cloud, e di prodotti da poter utilizzare all'interno di macchine proprie, come Cloudera Distribution Hadoop (CHD). Infine anche l'utilizzo di *framework* per il calcolo distribuito, come Apache Spark, sono di fondamentale importanza.

1.2 Processi del progetto

1.2.1 Metodologia di sviluppo

All'interno di Data Reply vengono utilizzati diverse metodologie di sviluppo, che vanno dalle metodologie agile, nella quale vediamo tra le più utilizzate Scrum e Kanban, a metodologie a cascata. Essendo un'azienda di consulenza, spesso si trova a lavorare anche con altre aziende allo stesso progetto e per questo motivo non è raro trovare metodologie più rigide ai cambiamenti come le metodologie a cascata.

1.2.2 Strumenti di supporto

Gli strumenti di supporto utilizzati, come le metodologie di sviluppo, sono decisi in base al progetto, al cliente e in accordo con le altre aziende che lavoreranno insieme al team Data Reply. Di consueto, l'azienda propone l'utilizzo di Git come strumento di versionamento, GitLab per la gestione dei repository di lavoro e GitLab Mattermost per le comunicazioni informali tra i componenti del progetto e dell'azienda.

1.2.3 Propensione alla modernizzazione

L'azienda cerca sempre di rinnovarsi e rimanere al passo con le ultime tecnologie in modo da poter proporre ai propri clienti il giusto compromesso tra novità e affidabilità. Spesso infatti, vi sono progetti mirati allo studio ed utilizzo di tecnologie e prodotti da poco sul mercato, così da garantire un vantaggio competitivo sulle altre aziende. Questo si può anche notare dai raduni svolti dall'azienda madre Reply nella quale Data Reply cerca sempre di portare progetti innovativi e che vadano a toccare le ultime tendenze del momento.

1.3 La gestione dei dati

I *big data* hanno rivoluzionato le possibilità di utilizzo dei dati e con esso anche il modo in cui le aziende utilizzano i dati dei clienti e da loro prodotti. Per questo motivo, vi è sempre più attenzione sul come un dato viene trattato per essere immagazzinato e sfruttato al meglio.
Nasce così il bisogno di acquisire dati in tempo reale o in *batch*, conservarli e renderli accessibili in formato originale o anche in modi in cui le informazioni risaltino ancora di più all'occhio. Questo processo viene chiamato ***Data Ingestion*** (o anche più semplicemente ingestion). Questo, dunque, è solo una delle fasi che sono necessarie per la gestione dei dati. Le altre possono essere riassunte in:

- ***Data Processing***, nella quale troviamo la lavorazione del dato grezzo, in modo che sia pronto ad essere analizzato secondo gli standard aziendali e la capacità di automatizzare ed ingegnerizzare l'estrazione delle informazioni, in modo da rendere periodico e ripetibile il lavoro svolto sui dati;
- ***Data Analysis***, in cui viene svolta la ricerca e la creazione di modelli per l'estrazione di informazioni;
- ***Data Integration***, vi è la fruizione delle informazioni ricavate attraverso applicativi che interrogano i dati analizzati in modo da utilizzarli per scopi specifici (e.g. report aziendali).

1.4 Cos'è l'anomaly detection?

L'*anomaly detection* è una tecnica statistica che permette di identificare "irregolarità" nei dati della serie temporale per un determinato valore di dimensione o metrica. Utilizzata sempre di più negli ultimi anni grazie alle nuove scoperte nel settore dei *big data*, può rendere la vita di molti lavoratori più semplice notificando in modo repentino le situazioni che vanno al di fuori del normale.

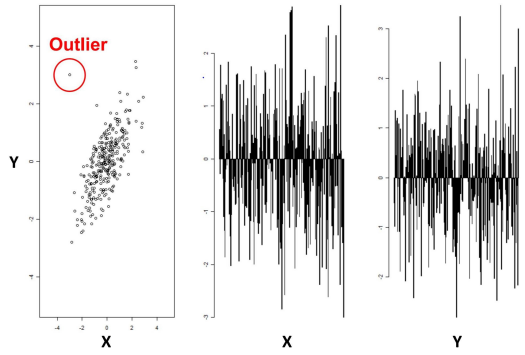

Figura 1.2: Esempio di anomalia [**?**]

Nell'immagine possiamo vedere un chiaro caso di anomalia, il punto cerchiato in rosso si trova al di fuori della normale distribuzione dei punti. Il rilevamento delle anomalie risulta utile in molti settori, tra cui quello delle frodi bancarie e il monitoraggio dello stato del sistema.

1.5 Organizzazione del documento

In questa sezione vengono riassunti i contenuti di ogni capitolo e le convenzioni tipografiche adottate.

1.5.1 Contenuti

Il secondo capitolo descrive l'evoluzione dello stage, dalla proposta alla pianificazione del lavoro.

Il terzo capitolo approfondisce l'analisi e la progettazione del progetto, descrivendo metodologia di lavoro, i requisiti trovati e i risultati ottenuti.

Il quarto capitolo approfondisce le tecnologie e gli strumenti affrontate spiegando l'integrazione fra tutte le parti.

Il quinto capitolo vengono tratte le conclusioni e argomentate le considerazioni personali.

1.5.2 Convenzioni tipografiche

Riguardo la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

- gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
- i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*.

2

Evoluzione dello stage

In questo capitolo verrà introdotto il progetto, spiegandone vincoli, obiettivi, pianificazione e motivi della scelta.

2.1 La proposta di stage

La proposta di stage mi è stata fatta nei mesi successivi alla visita all'azienda svolta attraverso l'Università di Padova. Fin da subito il complesso e lo spirito aziendale mi sono sembrati adatti ad ospitare degli studenti volenterosi di sperimentare nuove tecnologie.
Durante i colloqui con il Dott. Michele Giusto, mi è stato esposto un progetto incentrato sui *big data* e il rilevamento delle anomalie che corrispondeva al mio bisogno di affrontare nuove sfide e approfondire argomenti non trattati durante i corsi universitari.
Negli ultimi anni infatti, tra le tecnologie che hanno riscosso più successo e fama, troviamo i *big data* e tecniche di intelligenza artificiale con particolare attenzione al *machine learning*. Per questo motivo e per il mio già precedente interessamento verso queste tecnologie, ho deciso di dare forma agli obiettivi di questo progetto.

2.2 Il progetto

Il progetto si prefissava l'obiettivo di analizzare e costruire l'intero ciclo di vita del dato, secondo metodi, strumenti e tecniche innovative.

Per essere maggiormente concreti, il progetto prevedeva l'implementazione di un sistema di gestione dei dati aziendali del cliente, che quindi comportasse l'estrapolazione di informazioni dal dato grezzo, con successiva classificazione degli utenti che interagivano con il sistema stesso, in modo da intercettare eventuali comportamenti sospetti.
Per svolgere questo progetto, prima del mio arrivo e dell'effettiva proposta, l'azienda ospitante ha svolto uno studio di fattibilità in modo da sottolineare eventuali carenze di informazioni, criticità e punti a favore del progetto.
Durante il mio periodo di permanenza, il progetto è stato portato al termine seguendo le indicazioni iniziali e con alcuni aggiustamenti durante il percorso. Inoltre, sono state rispettate le fondamentali indicazioni del tutor aziendale che mi ha affiancato durante lo stage.
Terminato il mio stage il prodotto risultava conforme alle aspettative e facilmente manutenibile grazie alla documentazione redatta durante il periodo di studio del problema, sviluppo e test.

2.2.1 Obiettivi

Durante i colloqui di esposizione del progetto di stage, sono sorti alcuni obiettivi da soddisfare al termine dello stesso.
Essi si dividono in:

- Tutti gli obiettivi che sono fondamentali per la corretta riuscita del progetto, anche detti, obiettivi obbligatori:
  - Il sistema deve essere in grado di aggiornare periodicamente la concezione di normalità, per adattarsi al contesto dinamico;
  - Il sistema deve essere in grado di ricevere i flussi *batch* ed archiviarli dove desiderato;
  - Il sistema deve essere in grado di processare i flussi *batch* producendo i *dataset* di output richiesti;
  - I processi devono essere programmati per avviarsi periodicamente.

- Obiettivi desiderabili, cioè, gli obiettivi che porterebbero ad una maggiore completezza del progetto di stage ma che non sono fondamentali per la sua buona riuscita:
  - Il sistema deve riuscire ad aggiornarsi continuamente senza interrompere l'erogazione del servizio;

  - Il sistema deve essere in grado di scrivere i risultati su una struttura interrogabile via SQL;
  - Il sistema deve poter scalare le risorse in base al volume di dati.

- Tutti gli obiettivi che sono marginali al progetto ma che porterebbero comunque valore aggiunto allo stesso, anche detti, obiettivi facoltativi:

  - Il sistema deve poter lavorare su dati archiviati su Elasticsearch;
  - Il sistema deve poter essere in grado di scrivere i propri output su *database NoSQL*.

2.2.2 Vincoli

Durante i colloqui di esposizione del progetto di stage, sono sorti alcuni vincoli da soddisfare per svolgerlo al meglio.

Vincoli temporali

La durata dello stage doveva essere al minimo di 300 ore e al massimo di 320, in modo da rispettare i crediti stabiliti all'interno del piano di studi, svolte tutte presso la sede in via Robert Koch 1/4 a Milano negli orari 9.00-13.00, 14.00-18.00.

Vincoli tecnologici

Utilizzo della suite Google Cloud per lo sviluppo dell'*ingestion* dei dati e di Spark MLlib per la parte di riconoscimento delle anomalie.

2.3 Motivi della scelta

La scelta di abbracciare il progetto di Data Reply è dovuto principalmente a 4 motivi:

- Proposta di un progetto interessante con esperti del settore;
- Azienda dinamica e giovane con grandi ambizioni;

- Argomenti e tecnologie molto ricercate nell'ultimo periodo che mi permetteranno di inserirmi al meglio nel mondo del lavoro;
- Possibilità di svolgere lo stage presso la sede di Milano, città molto dinamica e innovativa.

Questi motivi mi hanno convinto a scegliere questa proposta per mettere alla prova le mie abilità.

2.4 Pianificazione del lavoro

Durante i colloqui svolti con il tutor aziendale, è stato redatto il piano di lavoro. Ciò ha portato la suddivisione dello stage in 8 parti, ognuna della durata di una settimana.

Prima Settimana (40 ore)

- Incontro con persone coinvolte nel progetto per discutere i requisiti e le richieste relativamente al sistema da sviluppare;
- Verifica credenziali e strumenti di lavoro assegnati;
- Presa visione dell'infrastruttura esistente;
- Formazione sulle tecnologie adottate.

Seconda Settimana (40 ore)

- Comprensione strumenti di storicizzazione;
- Analisi funzionamento *filesystem* distribuito;
- Produzione script Python per produzione dati simulati.

Terza Settimana (40 ore)

- Caricamento dati su *filesystem* distribuito;
- Securizzazione accesso ai dati;
- Studio teorico approccio statistico al problema.

Quarta Settimana (40 ore)

- Comprensione strumenti di processamento;
- Analisi funzionamento *RDD*;
- Implementazione processo *batch* con Python e Spark.

Quinta Settimana (40 ore)

- Analisi funzionamento *Dataframe*;
- Implementazione processo di elaborazione Spark;
- Ingegnerizzazione soluzione *batch*.

Sesta Settimana (40 ore)

- Test prestazionale processo Spark;
- Tuning prestazionale;
- Studio applicazione real-time.

Settima Settimana (40 ore)

- Storicizzazione dati su *storage* persistente;
- Studio funzionamento Kafka;
- Implementazione processo real-time con Spark Streaming.

Ottava Settimana - Conclusione (40 ore)

- Costruzione *layer* accesso al dato persistente;
- Ingegnerizzazione soluzione Spark Streaming;
- Integrazione *storage* persistente.

3

Analisi e progettazione

In questo capitolo analizzeremo le metodologie di lavoro, i problemi riscontrati e i risultati ottenuti.

3.1 Metodo di lavoro

Dopo un'attenta analisi delle esigenze di questo progetto, la metodologia che abbiamo deciso di utilizzare è Agile Kanban. Questa metodologia, ci ha permesso di avere un'alta flessibilità sui task da svolgere, una visione ampia del progetto e il focus su una singola attività alla volta. La metodologia Kanban è stata scelta anche per affrontare l'esiguo numero del team, composto da sole 2 persone, con effettivamente solo una che lavorava a pieno regime su di esso. Questo perché rendeva un'ottima visione dell'insieme delle attività da svolgere, in progresso e già svolte, oltre a non necessitare di ruoli predefiniti come nell'Agile scrum

Metodologia kanban Fa parte della famiglia delle metodologie agile, alla base vi è l'utilizzo di una lavagna nella quale vi è specificato uno stato per ogni colonna presente. Quello sopra riportato è uno degli esempi più semplici di lavagna Kanban, nella quale ogni task deve assumere tre stati per considerarsi completato "ToDo", "Doing" e "Done". Solitamente ogni componente del team ha la responsabilità di portare a completamente un solo task alla volta, così da evitare sovrapposizioni e confusione.

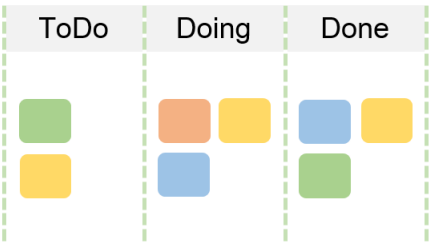

Figura 3.1: Esempio lavagna Kanban.

Questa metodologia ha sicuramente giovato al progetto dato l'alto livello di flessibilità garantito e necessario al rispetto dalle richieste periodiche del cliente.

Trello Dato che i componenti del team difficilmente sarebbero riusciti a condividere lo stesso spazio di lavoro, inteso come luogo fisico, abbiamo deciso di utilizzare una lavagna digitale fornita da Trello.
Trello rende disponibile gratuitamente online un ottima implementazione dei principi esposti all'interno della metodologia Kanban, gli elementi chiave risultano essere 3: la lavagna, le liste (o anche colonne della lavagna) e le card (cioè i singoli task).

Figura 3.2: Logo di Trello [**?**].

3.2 Problemi affrontati

Affrontare sfide, problemi e quesiti di carattere tecnologico, architetturale e implementativo è stata sicuramente la parte più formativa del periodo di stage, dato che in queste situazioni lo studio autonomo e il confronto con esperti del settore come il tutor aziendale sono stati di primaria importanza.
I problemi affrontati durante questo progetto sono molteplici e vanno dall'analisi e scelta delle tecnologie da utilizzare all'integrazione tra le stesse.
Per circoscrivere le problematiche abbiamo deciso di sviluppare il progetto in due sotto-progetti autonomi. Il primo tratta dell'*ingestion* dei dati e il secondo dell'utilizzo dei dati con il fine

di categorizzare gli utenti. Da questi quindi, derivano diverse tipologie di problemi, tra cui la scelta e lo studio degli strumenti e delle best practices.
Il punto cruciale del sotto-progetto riguardante l'*ingestion* è stato lo studio delle tecnologie riguardanti Google Cloud Platform e con esso le conseguenti scelte obbligate, come lo studio di Apache Beam, l'utilizzo di sole librerie compatibili con Google Cloud ed i suoi strumenti. Per quanto riguarda la classificazione degli utenti, si è scelto di intraprendere una strada che potesse portare a dei vantaggi di utilizzo e portabilità, cioè si è deciso di utilizzare un container Docker per lo sviluppo ed utilizzo degli algoritmi di classificazione. Questo è stato fatto per agevolare l'installazione e configurazione in locale dell'intero pacchetto di strumenti da utilizzare, tra cui Spark, Kafka e Hadoop. Ovviamente, l'utilizzo di Docker ha comportato la scelta del miglior compromesso per la creazione di un container adatto. Dato l'ampio pacchetto di strumenti da studiare in autonomia, abbiamo deciso quindi di utilizzare un container Cloudera [**?**] già contenete la maggior parte degli strumenti necessari per sviluppare un processo di classificazione degli utenti. Inoltre, per far si che i dati venissero continuamente aggiornati, come effettivamente succede nella fase di *ingestion*, abbiamo deciso di utilizzare Kafka Straming per mantenere un flusso continuo di verifica delle anomalie, così da mantenere in costante aggiornamento il cliente.

3.3 Requisiti

3.3.1 Struttura

Ogni requisito è descritto dalla seguente struttura:

- Tipo;
- Importanza;
- Stato implementazione;
- Fonti.

Inoltre, a ciascun requisito corrisponde un codice identificativo cosi composto:

R {importanza}.{tipo}.{identificativo}

- R specifica che si tratta di un requisito;
- importanza identifica la rilevanza del requisito e può assumere 3 valori:

  - 0: indica che il requisito è obbligatorio e il suo soddisfacimento dovrà necessariamente avvenire;
  - 1: indica che il requisito è desiderabile, cioè il suo soddisfacimento può portare maggiore completezza al sistema ma non è fondamentale per lo stesso;
  - 2: indica che il requisito è opzionale, e quindi la decisione di implementarlo o meno verrà presa dopo le dovute considerazioni.

- tipo distingue se si tratta di un requisito funzionale (F), di qualità (Q), di prestazione (P) o di vincolo (V);
- identificativo è un numero progressivo che identifica i sottocasi.

3.3.2 Tabella dei requisiti

Per assolvere gli obiettivi del progetto e per rispondere in modo concreto ai problemi sorti, durante lo stage sono stati definiti i seguenti requisiti.

Id Requisito	**Descrizione**	**Fonte**
R0F1	Il sistema deve essere in grado di trasferire i file con estensione *evtx* e *XML* dal server SFTP a Google Cloud Storage.	Interno
R1F1.1	Il sistema deve essere in grado di verificare la corretta estensione dei file.	Interno
R1F1.2	Il sistema deve essere in grado di scartare i file con una non corretta estensione.	Interno
R1F1.3	Il sistema deve essere in grado di validare i file con estensione *evtx*.	Interno
R1F1.4	Il sistema deve essere in grado di convertire i file *evtx* in *XML*.	Interno

Id Requisito	Descrizione	Fonte
R0F2	Il sistema deve essere in grado di aggiornarsi periodicamente.	Interno
R1F2.1	Il sistema deve essere in grado verificare la presenza di nuovi file all'interno del server SFTP.	Interno
R1F2.2	Il sistema deve essere in grado di avviare il processo di *ingestion* dei file ad ogni nuovo arrivo.	Interno
R1F2.2.1	Il sistema deve essere in grado di notificare l'arrivo di nuovi file all'interno del Bucket Google Cloud Storage.	Interno
R1F2.2.2	Il sistema deve essere in grado di avviare il processo di *ingestion* una volta ricevuta la notifica dal Bucket Google Cloud Storage.	Interno
R0F3	Il sistema deve essere in grado di processare i flussi *batch* producendo i *dataset* di output richiesti.	Interno
R0F3.1	Il sistema deve essere in grado di archiviare i file di output all'interno del Bucket Google Cloud Storage.	Interno
R1F3.2	Il sistema deve essere in grado di archiviare i file di output all'interno di Google Cloud BigQuery.	Interno
R0F4	I processi devono essere schedulati per avviarsi periodicamente.	Interno
R1F5	Il sistema deve riuscire ad aggiornarsi continuamente senza interrompere l'erogazione del servizio.	Interno
R1F6	Il sistema deve essere in grado di scrivere i risultati su una struttura interrogabile via SQL.	Interno

Id Requisito	Descrizione	Fonte
R1F7	Il sistema deve essere in grado di classificare i dati all'interno dei file provenienti dal processo di *ingestion*.	Interno
R1F7.1	Il sistema deve essere in grado di classificare i dati all'interno dei file provenienti dal processo di *ingestion* per singolo giorno.	Interno
R1F7.2	Il sistema deve essere in grado di classificare i dati all'interno dei file provenienti dal processo di *ingestion* per singolo file su base multi-giorno.	Interno
R1F8	Il sistema deve essere in grado di inviare una email al manager responsabile in caso di anomalie nei dati classificati.	Interno
R1F9	Il sistema deve essere in grado di rendere visibile dei report all'interno di Google Data Studio.	Interno
R2F10	Il sistema deve poter lavorare su dati archiviati su Elasticsearch.	Interno
R2F11	Il sistema deve poter essere in grado di scrivere i propri output su DB NoSQL.	Interno
R0Q1	La progettazione e il codice devono seguire le norme riportate all'interno degli standard di Data Reply S.r.l.	Interno
R0Q2	Tutti i documenti e il codice prodotto devono rispettare le metriche riportate all'interno degli standard di Data Reply S.r.l.	Interno
R0Q3	Deve essere prodotto un manuale sviluppatore.	Interno
R0V1	Il sistema deve utilizzare strumenti appartenenti a Google Cloud Platform o ad esso compatibili.	Interno

Id Requisito	Descrizione	Fonte
R1V1	Il sistema deve utilizzare Spark MLlib per il sistema di classificazione.	Interno
R2V1	Il sistema di classificazione deve essere sviluppato all'interno di un container Docker.	Interno
R1P1	Il sistema deve poter scalare le risorse in base al volume di dati.	Interno

Tabella 3.1: Requisiti di Vincolo

3.4 Risultati

Alla fine del periodo di stage, i risultati ottenuti sono stati considerati più che positivi dal tutor aziendale Dott. Michele Giusto, dato che sono stati soddisfatti tutti i requisiti obbligatori e desiderabili. I requisiti facoltativi, non sono stati implementati per lasciare maggiore spazio e importanza ai requisiti obbligatori.
Inoltre, il sotto-progetto riguardante la fase di *ingestion* ha avuto ottimi risultati, tale per cui si è dimostrata pronta per essere utilizzata in fasi di "produzione". Per quanto riguarda la parte di classificazione, è stata messa a punto per scheletro e funzionalità senza però essere pronta per gestire effettivamente l'aggiornamento real time dei dati, soprattutto perché da parte del clienti vi è stato un rallentamento nel passaggio di dati ed informazioni per la scelta del miglior modello.

4

Sviluppo

In questa sezione, verranno introdotte le tecnologie che sono state utilizzate nel progetto di stage e verrà descritto come sono state integrate. All'interno delle sezione di integrazione capiremo come è stato possibile costruire un sistema che si occupi di selezionare i file corretti da leggere, importare il contenuto manipolato all'interno di un applicativo interrogabile dall'esterno, avviare delle analisi sui dati e creare dei report sugli stessi.

4.1 Integrazione

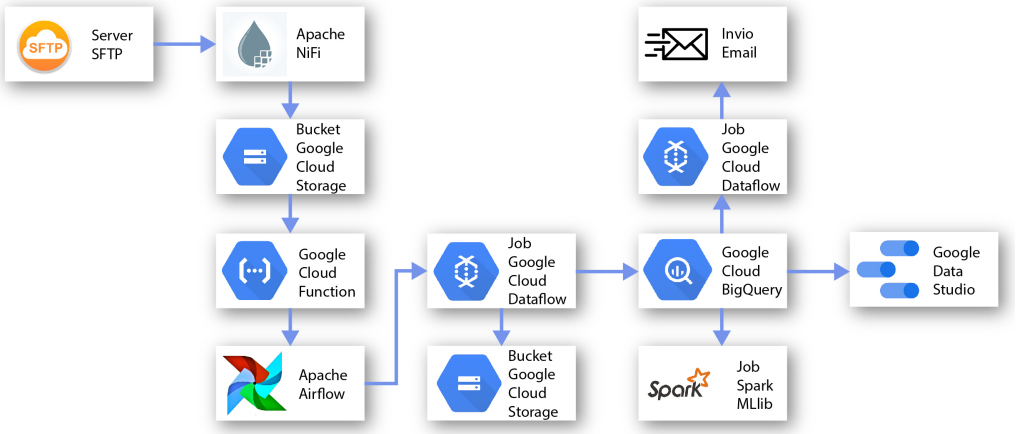

Figura 4.1: *Workflow* progetto.

L'integrazione delle tecnologie elencate in §4.2.1, ha portato a stabilire un vero e proprio ciclo di vita del dato. La gestione del dato inizia con il caricamento dei file all'interno di un server SFTP da parte del cliente. Tramite Apache NiFi, avviene una prima pre-elaborazione. Infatti, vengono accettati solo file con estensione *XML* o *evtx*, nel caso di file differenti questi vengono scartati. La pre-elaborazione la troviamo nella validazione e conversione dei file *evtx* in *XML* così da uniformare e generalizzare il più possibile le procedure di *ingestion* dei dati. Attraverso Apache Nifi, i file vengono spostati all'interno di Google Cloud Storage, il quale, al trasferimento di ogni file, scatena l'avvio delle Google Cloud Function. Come descritto all'interno di §4.2.1, Google Cloud Function offre la possibilità di essere avviata tramite eventi, in questo caso troviamo l'utilizzo dell'origine Cloud Storage di tipo *finalize*. Tramite la Function, viene avviato il processo di *ingestion* sul file appena trasferito tramite l'attivazione del file DAGs all'interno di AirFlow.
La fase di *Data Processing* è composta da alcuni *job* Google Cloud Dataflow scritti in Python e caratterizzati dall'utilizzo di Apache Beam per la creazione delle pipeline dei dati. Il processamento dei dati è mirato al rendere i dati maggiormente leggibili e consultabili decodificandoli e aggregandoli. Dopodiché, vengono scritti all'interno di apposite tabelle Google Cloud BigQuery.
Per quanto riguarda la *Data Analysis*, il cliente ha richiesto 2 tipologie distinte di analisi dei dati. La prima riguardava il confronto dei dati giornalieri con i dati storicizzati, per capire se vi fossero possibili anomalie riscontrate (come ad esempio la forte diminuzione dell'utilizzo di determinati strumenti), e nel caso di riscontri, immediatamente notificarlo al responsabile del settore tramite email. La secondo riguardava la classificazione degli utenti che utilizzano il sistema del cliente per trovare eventuali cambi di categorie non autorizzate (che corrispondono ad anomalie). Per concludere il processo da dato ad informazione, abbiamo costruito un sistema che, tramite Kafka e quindi secondo lo schema produttore-consumatore in *streaming*, invia dei dati ad un algoritmo di *machine learning* sviluppato tramite Apache Spark (più in particolare tramite la libreria Apache Spark MLlib) che raggruppa gli utenti su base settimanale, conserva i valori dei raggruppamenti e verifica che ogni utente non abbia cambi di cluster in modo anomalo.

Al termine di questo processo, abbiamo il definitivo passaggio da semplice dato ad effettiva informazione per cliente tramite i report e le *dashboard* riassuntive di Google Data Studio. Questa fase viene di norma chiamata *Data Integration*.

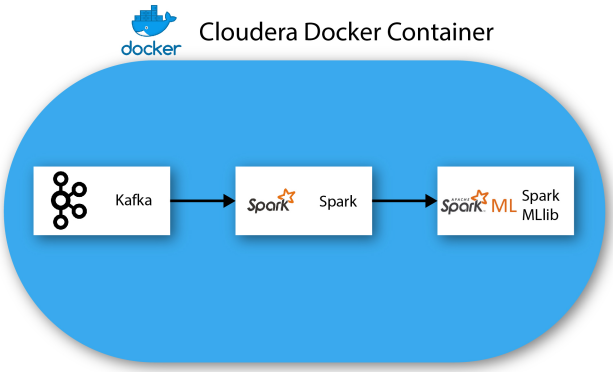

Figura 4.2: *Workflow* integrazione Kafka e Spark all'interno di Docker Container.

4.2 Tecnologie

4.2.1 Google Cloud

Il principale servizio utilizzato durante il progetto è stato Google Cloud, in esso troviamo una moltitudine di strumenti utili allo sviluppo, all'esecuzione e all'interazione di progetti riguardanti *big data* e Intelligenza Artificiale. Inoltre, importante risulta la possibilità di consultare, manipolare e presentare i dati. Il progetto Google Cloud (anche detto Google Cloud Platform o GCP), viene lanciato nel 2008 con uno dei suoi servizi di punta ancora oggi App engine. Nell'arco degli anni sia il bacino di utenti, sia i servizi offerti sono via via aumentati. Oggigiorno i suoi clienti [?] sono molti, tra cui grandi aziende, e le funzionalità di spicco sono molteplici, i campi di maggiore interesse riguardano strumenti di Computazione, Immagazzinamento & *Database*, *big data*, *Cloud AI* e *IoT*. Per quanto riguarda l'esperienza personale, ho trovato Google Cloud un buon servizio, con dei problemi da risolvere sulla completezza della documentazioni e a volte con grafiche da migliorare, ma comunque un ottimo compromesso tra qualità e prezzo. Il resto della sezione introduce il prodotti e i servizi GCP utilizzati durante lo stage.

Google Cloud Storage

Lanciato nel 2010, Google Cloud Storage è il servizio di hosting dei file della piattaforma. All'interno ogni utente può scegliere in che modalità salvare i dati (zona del mondo, carico di lavoro e frequenza d'accesso), così da trovare la soluzione maggiormente adatta ai propri bisogni. I suoi punti di forza sono:

Figura 4.3: Logo Google Cloud Platform.

- Interoperabilità: possibilità di interfacciarsi con strumenti e servizi di aziende terze (come Amazon S3);
- Consistenza: tutte le operazioni sono svolte secondo il principio di Atomicità;
- Controllo di accesso: possibilità di definire l'accesso ad un singolo oggetto o *bucket* a seconda della categoria di appartenenza degli utenti;
- *Resumable Upload*: in caso di fallimento di un *upload*, questo può essere continuato/ricominciato.

Per quanto riguarda l'esperienza d'utilizzo, si è rivelato un ottimo servizio di *storage*, l'abilitazione di un'unica API rende disponibile l'intero pacchetto di funzionalità e ciò l'ho trovato molto comodo.

Google Cloud BigQuery

Servizio lanciato insieme a Google Cloud Storage nel 2010, Google Cloud BigQuery , è lo strumento che permette di interagire e analizzare grandi moli da dati, definito come un *data warehouse*. All'interno di GCP, vi è un'apposita interfaccia utente per utilizzarlo. Si basa su tabelle codificate in formato *JSON* e sulla quale vengono processate query in standard SQL [?] tramite avanzati algoritmi di MapReduce.
I principali punti di forza sono:

- La piattaforma rende disponibile la creazione e distruzione di tabelle basati su schemi JSON, inoltre rende disponibile l'import e l'export di dati in formati come *CSV* e J*SON*;
- La semplicità del linguaggio SQL con scalabilità e prestazioni che solo i servizi cloud riescono a dare;
- Un'ottima integrazione con servizi e *API* esterni, per la raccolta e l'utilizzo dei dati;
- Possibilità di garantire l'accesso ad uno o più categorie di utenti;
- Possibilità di interagire con strumenti di *machine learning* nativi di Google e non (come TensorFlow).

L'opinione del team di sviluppo riguardo lo strumento è molto positiva sopratutto grazie alla scalabilità automatica che rende semplice e veloce l'utilizzo delle query su grandi moli di dati, un possibile punto di debolezza può essere trovato nell'utilizzo in sola lettura dei dati esistenti, quindi non vi è la possibilità di modifica dei record se non un nuovo inserimento.

Google Cloud Dataflow

Rilasciato al pubblico nel 2015, Google Cloud Dataflow è lo strumento che permette di processare pipeline per l'integrazione, la preparazione e l'analisi di grandi quantità di dati. Il modello di pipeline consigliato è quello di Apache Beam che vedremo in seguito.
Tra i maggiori punti di forza troviamo:

- Creazione di pipeline secondo modelli già esistenti o personalizzati dall'utente;
- Composizione di pipeline, e quindi *job* specifici, direttamente da codice;
- Possibilità di utilizzo di dati da molteplici fonti, non solo da strumenti Google;
- Ottima scalabilità interna in base agli effettivi bisogni del lavoro in esecuzione;
- Buona granularità della specificazione dei parametri di esecuzione[**?**] della pipeline;
- Visualizzazione grafica della pipeline.

Per quanto concerne l'utilizzo di questo prodotto, il team si esprime con un giudizio più che positivo dato anche la grande quantità di documentazione disponibile.

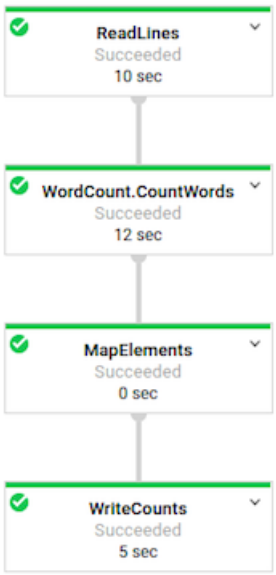

Figura 4.4: Esempio di pipeline Dataflow

Google Cloud Function

Rilasciato in beta nel 2017, Google Cloud Functions è il perfetto intermediario tra gli eventi che vengono scaturiti nel cloud e l'invocazione di API. Le origini degli eventi supportati sono:

- Cloud Pub/Sub;
- Cloud Storage;
- HTTP;
- Stackdriver Logging;
- Firebase.

Inoltre, per ogni tipologia di evento vi sono ulteriori specificazioni che non verranno trattate tutte in modo estensivo in questo elaborato.
Per quanto concerne l'utilizzo svolto durante il progetto, il team ritiene questo strumento valido e con ottime potenzialità e margini di miglioramento. Uno svantaggio molto importante da sottolineare è la scarsa documentazioni presente nelle fonti ufficiali.

4.2.2 Apache NiFi

> An easy to use, powerful, and reliable system to process and distribute data.
>
> NiFi Creator

La descrizione che viene data nella documentazione di NiFi [**?**] riassume alla perfezione le caratteristiche di questo strumento e ciò che ci permette di fare.
Innanzitutto, le sue funzioni sono proprio quelle di automatizzare lo "scorrere" del flusso dati all'interno del sistema in cui viene utilizzato. Le funzionalità sono molteplici e ben rodate, vanno dal semplice spostamento dei file, al filtraggio, composizione e conversione in diversi formati. La documentazione è molto esauriente, spiegando anche più del necessario tutti i passaggi da seguire per utilizzare un determinato componente. La semplicità e l'intuitività nel costruire il flusso di dati, rendono l'esperienza utente decisamente piacevole, anche grazie all'interfaccia grafica semplice e intuitiva. Altro punto a favore è l'affidabilità del prodotto e la grande quantità di componenti disponibili orientati alla sicurezza, così da rendere il passaggio di informazioni il più sicuro possibile.

Figura 4.5: Logo Apache NiFi

Google Data Studio

Lanciato nel 2016, Google Data Studio è lo strumento di creazione e visualizzazione dei dati attraverso report e *dashboard*. L'idea che sta alla base di questo *tool* è la condivisione dei dati che sono presenti all'interno di GCP attraverso tabelle e grafici.
I maggiori vantaggi nell'utilizzarlo sono:

- Accesso ai dati personali di GCP;
- Possibilità di integrare i dati fra di loro con query standard SQL;
- Condivisione a gruppi di utenti.

Durante l'esperienza lo strumento si è rivelato molto acerbo nelle sue funzionalità e nell'utilizzo, oltre a dover attendere troppo tempo per il caricamento di dati e grafici. L'idea alla base è ottima, dato che vi è la possibilità di utilizzare i dati all'interno di GCP anche in continuo aggiornamento, d'altro canto la documentazione e l'interfaccia utente non rendono semplice quanto potrebbero la creazione di un report aziendale da presentare al cliente.

4.2.3 Apache Beam

Apache Beam è un implementazione del modello Dataflow [**?**], esso infatti fissa come definire ed eseguire pipeline sui dati di tipo *ETL* (*extract*, *transform*, and *load*), *batch* e *streaming* continui. Questo progetto nasce dopo il rilascio di un software development kit (SDK) contenente un implementazione del modello Dataflow da parte di Google nel 2014, nel 2016 Google decise di concedere il core del proprio SDK per costituire il progetto OpenSource Apache Beam.
I concetti alla base di questa implementazione, e quindi del modello Dataflow, possono essere riassunti nel cercare un modello il quale riesca a definire ed eseguire pipeline di tipo *batch* e *streaming* allo stesso tempo, senza dover differenziare le pipeline come nel modello *Lambda architecture* [**?**].
Un tipico utilizzo di una pipeline può essere il seguente:

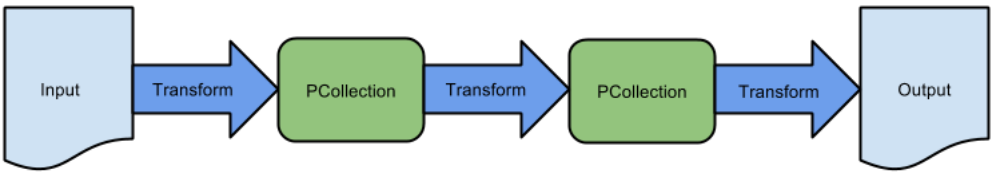

Figura 4.6: Esempio utilizzo Apache Beam.

- Creazione della *Pipeline*;
- Creazione di una *PCollection* da dei dati di input;
- Applicazione di *PTrasforms* sulla *PCollection*;
- Scrittura dei dati all'interno di un file.

Le azioni di maggiore interesse stanno nelle PTrasforms, infatti rende possibile processare ogni singolo elemento delle PCollection, effettuare raggruppamenti per uno o più elementi, unire e partizionare le *PCollection*. La documentazione a disposizione è decisamente idonea e precisa rispetto alla potenzialità di utilizzo. Inoltre, essendo un progetto OpenSource vi è la possibilità di collaborare al continuo sviluppo, e quindi al successo, che questa tecnologia sta avendo.

4.2.4 Apache AirFlow

Apache AirFlow è una piattaforma per creare, schedulare e monitorare workflows. Airflow, inoltre, cerca la soluzione più performante per l'esecuzione dei task, creando un Grafo Aciclico Diretto (meglio conosciuto come "DAGs" cioè "Directed Acyclic Graphs") in modo da riconoscere le dipendenze tra i compiti stessi ed eseguire i task con meno dipendenze. La scelta di utilizzare questa tecnologia è stata presa sia per la già precedente esperienza aziendale con essa, sia perché completamente compatibile e già installata all'interno di Google Cloud Composer. Data la sua notevole compatibilità ci ha permesso di soddisfare un requisito del cliente riguardante l'*ingestion* in tempo reale dei dati inviati.

4.2.5 Docker

Nato nel 2013, Docker è una piattaforma software che permette di creare *build*, testare e distribuire applicazioni con la massima rapidità. Docker raccoglie il software in unità standardizzate chiamate container che offrono tutto il necessario per la loro corretta esecuzione. L'utilizzo di Docker in questo progetto per la parte del sistema di classificazione è coerente con i principali pregi e funzionalità offerte da questo strumento. Tra le principali funzionalità troviamo:

- Integrazione con i maggiori infrastrutture;
- Portabilità;
- Controllo di versione e riutilizzo dei componenti;
- Facilità nella condivisione.

4.2.6 Kafka

Apache Kafka è una piattaforma *open source* di *stream processing*, mira a creare una piattaforma a bassa latenza ed alta velocità per la gestione di dati in tempo reale. Apache Kafka è quindi un broker di messaggistica, dato che una volta definiti dei *topic*, la domanda di dati del *consumer* e l'offerta di dati del *producer* è mediata da un Kafka server. Questo offre alcuni vantaggi:

- Memorizzazione dei messaggi in modo da prevenire la perdita dei dati;
- Alta flessibilità di variazione nel flusso, quindi controllo e prevenzione di eventuali colli di bottiglia;
- Tollerante agli errori;
- Buona scalabilità.

Nel progetto, funge da intermediario tra il database e il sistema di classificazione, per mantenere un flusso continuo di dati e scovare eventuali operazione sospette.

4.2.7 Apache Spark

Sviluppato all'interno dell'Università della California e poi donato ad Apache, Spark è un *framework* per il calcolo distribuito. All'interno di Apache Spark vi troviamo una moltitudine di funzionalità e librerie, tra cui Apache Spark MlLib.

Apache Spark MlLib

All'interno di questa libreria, troviamo alcuni tra i più famosi e preformanti algoritmi di *machine learning*. Questa libreria è stata utilizzata per la sua semplicità di utilizzo e per le sue prestazioni ottime nei confronti delle altre librerie.
K-Means [?] è l'algoritmo utilizzato nel progetto, scelto per iniziare una fase di perlustrazione della libreria e mantenuto dato lo stato ancora non avanzato del progetto nella parte di analisi ed utilizzo delle informazioni.

4.3 Strumenti

Gli strumenti di lavoro utilizzati sono stati scelti in base agli standard aziendali e per mantenere la massima compatibilità con il pacchetto GCP.

Python è stato il linguaggio di riferimento per il progetto, la decisione è stata presa tenendo in considerazione i linguaggi supportati da GCP e pensando all'utilizzo nella parte di riconoscimento delle anomalie. Infatti, la numerosa presenza di librerie di supporto e la presenza di molta documentazione a supporto per il medesimo linguaggio ha spostato l'ago della bilancia su Python al posto di linguaggi comunque molto diffusi e conosciuti come Java. Inoltre, essendo un linguaggio non studiato durante i corsi di laurea, ho ritenuto fosse un'ottima occasione per iniziare ad utilizzarlo.
PyCharm è stata conseguenza diretta dell'uso di Python, dato che è un'ottimo *IDE* che offre assistenza durante la codifica. Oltre a garantire una buona compatibilità con GCP, fondamentale per il ruolo che ricopre.
Aderendo alle consuetudini aziendali, il sistema di controllo di versione adottato è Git, attraverso la piattaforma GitLab. Si è inoltre deciso di utilizzare GitLab Mattermost per la comunicazione all'interno del team.

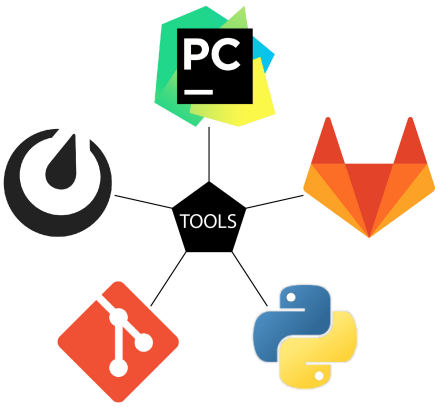

Figura 4.7: Loghi strumenti utilizzati.

5 Conclusione

All'interno di questo capitolo vengono esposte le considerazioni oggettive e soggettive riguardante l'esperienza fatta.

5.1 Consuntivo finale

Il progetto ha avuto la durata inizialmente prevista, anche se sono state apportate alcune modifiche alla pianificazione iniziale. Per questo motivo, riporto la pianificazione aggiornata all'ultima settimana di stage.

- Comprensione e analisi delle componenti del progetto: 56 ore;
- Realizzazione di pipeline *batch* al fine di popolare *data lake* su cui svolgere processamenti periodici : 140 ore;
- Test e validazione pipeline *batch* : 20 ore;
- Studio e applicazione del concetto di *Data Ingestion*: 10 ore;
- Analisi, creazione e codifica *job* di classificazione all'interno del Docker container:40 ore;
- Creazione e codifica *job* rilevamento delle anomalie tra dati giornalieri: 15 ore;
- Studio e applicazione del concetto di *Data Integration*: 10 ore;

- Integrazione finale tra le componenti principali della gestione dei dati: 19 ore;
- Documentazione progetto: 10 ore;

5.2 Considerazione sugli obiettivi

Di seguito vengono elencati in forma tabellare gli obiettivi con il loro relativo stato.

Obiettivi obbligatori	**Stato**
Il sistema deve essere in grado di aggiornare periodicamente la concezione di normalità, per adattarsi al contesto dinamico.	Completato
Il sistema deve essere in grado di ricevere i flussi *batch* ed archiviarli dove desiderato.	Completato
Il sistema deve essere in grado di processare i flussi *batch* producendo i *dataset* di output richiesti.	Completato
I processi devono essere programmati per avviarsi periodicamente.	Completato

Tabella 5.1: Obiettivi obbligatori

Obiettivi desiderabili	**Stato**
Il sistema deve riuscire ad aggiornarsi continuamente senza interrompere l'erogazione del servizio.	Completato
Il sistema deve essere in grado di scrivere i risultati su una struttura interrogabile via SQL.	Completato
Il sistema deve poter scalare le risorse in base al volume di dati.	Completato

Tabella 5.2: Obiettivi desiderabili

Obiettivi facoltativi	Stato
Il sistema deve poter lavorare su dati archiviati su Elasticsearch.	Non implementato
Il sistema deve poter essere in grado di scrivere i propri output su DB NoSQL.	Non implementato

Tabella 5.3: Obiettivi facoltativi

Gli obiettivi obbligatori sono stati completati, come anche gli obiettivi desiderabili. Gli obiettivi facoltativi sono risultati superflui alle richieste del cliente e per questo non sviluppati.

5.3 Sviluppi futuri

Il prodotto sviluppato presenta una buona capacità di gestione del dato, senza però sfruttare al massimo le capacità di classificazione, dato l'utilizzo di un semplice algoritmo di *clustering*. Per questo motivo, consiglio di cambiare l'algoritmo di classificazione, scegliendone uno maggiormente performante e prestazionale. Inoltre, vi è da completare il passaggio di dati da classificare e quindi completare l'implementazione del produttore/consumatore Kafka.
Data la modularità delle classi create per Google Dataflow, vi è la possibilità di aggiungere facilmente la fase di *data ingestion* per altre tipologie di file o di schema *XML* diversi da quelli trattati attualmente.

5.4 Bilancio formativo

I *big data* e l'utilizzo di algoritmi di *machine learning* per il riconoscimento delle anomalie, sono argomenti che mi hanno sempre incuriosito e che volevo approfondire maggiormente. All'interno dello stage ho avuto la possibilità di confrontarmi con svariate problematiche che mi hanno reso conscio del mio attuale bagaglio di conoscenze e che mi hanno aiutato a maturare il mio pensiero sul mondo della consulenza. Le competenze da me acquisite durante questo periodo di stage, possono essere riassunte con il seguente elenco:

- Concetti di gestione dei dati;
- Nozioni di *big data*;

- Nozioni di Riconoscimento delle Anomalie;
- Utilizzo strumenti e servizi GCP (usati nel progetto);
- Creazione flusso di gestione dei dati;
- Utilizzo Apache NiFi;
- Utilizzo Apache AirFlow;
- Creazione DAGs;
- Nozioni ed utilizzo di Apache Beam;
- Nozioni di Apache Spark;
- Utilizzo di Apache Spark MLlib;
- Condivisione delle informazioni tramite Google Data Studio;
- Utilizzo di Docker;
- Utilizzo di Kafka.

Il livello delle competenze acquisite non è paragonabile a quello di un professionista del settore, ma credo di avere raggiunto un livello discreto nella maggior parte degli strumenti utilizzati, un livello che mi permetterà di approfondire e di utilizzare tutto ciò che riguarda quegli strumenti e tecnologie. Ciò che ho imparato e visto durante questa esperienza mi ha sicuramente aiutato a scegliere il percorso da seguire nei prossimi anni.

5.5 Considerazioni personali

Lo stage è stato sicuramente più che positivo, sia sotto il profilo formativo che esperienziale. Questo non vuol dire che non ci siano state note negative o piccoli problemi durante questa esperienza, infatti trattandosi di un'esperienza all'interno di un'azienda di consulenza, la comunicazione con il cliente era spesso di fondamentale importanza, per questo credo che una comunicazione più fitta e veloce potesse produrre maggiori risultati. Nel complesso comunque, sono più che soddisfatto degli obiettivi raggiunti e del lavoro svolto all'interno del team di Data Reply.

Glossario

agile scrum Scrum è un framework agile per la gestione del ciclo di sviluppo del software, iterativo ed incrementale.. 11

app engine Piattaforma come servizio (PaaS) di cloud computing per lo sviluppo e l'hosting di applicazioni web gestite dai Google Data Center.. 21

atomicità Caratteristica nella quale le transazioni sono viste come un'unica operazione di esecuzione indivisibile dal punto di vista logico.. 22

best practices Si intendono le esperienze, le procedure o le azioni più significative, o comunque quelle che hanno permesso di ottenere i migliori risultati, relativamente a svariati contesti e obiettivi preposti.. 13

bucket Identifica l'oggetto contenitore all'interno di Google Cloud Storage, vengono distinte dalle semplici cartelle, dato che vi si possono specificare parametri di controllo dell'accesso.. 22

Cloudera Distribution Hadoop (CHD) Distribuzione dell'ecosistena Apache Hadoop gestita dalla società Cloudera Inc.. 2

container Si tratta di un insieme di uno o più processi isolati dal resto del sistema. Poiché tutti i file necessari per eseguire tali processi vengono forniti da un'immagine distinta, i container Linux sono portabili e coerenti in tutti gli ambienti, dallo sviluppo ai test e infine alla produzione. 13

DAGs All'interno di AirFlow si tratta di una collezione di tutte le attività organizzate in modo da mostrare tutte le relazioni e dipendenze.. 27

data lake sistema o repository di dati conservati nel loro formato originale o prima delle elaborazioni su di essi.. 31

data warehouse collezione o aggregazione di dati strutturati.. 22

mapReduce Si tratta di un framework software brevettato e introdotto da Google per supportare la computazione parallela e distribuita su grandi quantità di dati in cluster di computer. Un programma MapReduce è costituito da una procedura di map dei dati che si occupa di filtrare ed ordinare i dati e una procedura di reduce che si occupa di ragguppare i dati in base ad un insieme di chiavi.. 22

openSource Viene riferito a un tipo di software o al suo modello di sviluppo o distribuzione. La sua caratteristica principale è la pubblicazione del codice sorgente. 26

software development kit (SDK) Insieme di strumenti per lo sviluppo e la documentazione di software.. 26

workflows Indica il flusso di lavoro, o meglio, indica la gestione dei processi lavorativi attraverso la creazione di modelli e il management informatico dell'insieme dei task da svolgere per raggiungere specifici obiettivi di business. 27

Bibliografia