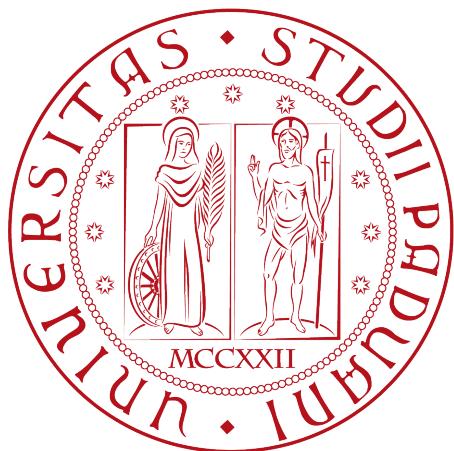


Università degli Studi di Padova
DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"
CORSO DI LAUREA IN INFORMATICA



**Introduzione al big data computing ed una
sua applicazione**

Tesi di laurea triennale

Relatore

Prof.Tullio Vardanega

Laureando

Stefano Panozzo

ANNO ACCADEMICO 2017-2018

Stefano Panizzo: *Introduzione al big data computing ed una sua applicazione*, Tesi di laurea triennale, © Dicembre 2018.

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage del laureando Stefano Panozzo presso l'azienda IT Euro Consulting di Padova. Lo stage è stato svolto alla conclusione del percorso di studi della Laurea Triennale ed è durato in totale 320 ore. Gli obiettivi da raggiungere erano molteplici.

La prima richiesta dell'azienda era analizzare la struttura del [cluster](#) in cui risiedevano i dati da utilizzare in seguito. Successivamente l'azienda richiedeva l'analisi e la trasformazione del dataset di interesse per estrarre ed ottenere nuove informazioni utili per creare un modello che prevedesse il target desiderato. Infine, la progettazione e lo sviluppo di una [web app](#) per la rappresentazione dei risultati ottenuti in precedenza. I primi due capitoli del presente documento hanno lo scopo di presentare il contesto aziendale in cui è stato sostenuto lo stage e di spiegare come il progetto di stage si renda utile all'interno della strategia aziendale. Il terzo capitolo documenta lo svolgimento dello stage descrivendo le attività che sono state portate a termine, i punti salienti del progetto stesso e le principali scelte progettuali. Il quarto ed ultimo capitolo presenta infine una valutazione dello svolgimento dello stage rispetto agli obiettivi aziendali e alle conoscenze acquisite dallo studente.

Indice

1 Il contesto aziendale	1
1.1 Il profilo aziendale	1
1.2 Tecnologie utilizzate	2
1.2.1 Sviluppo software	3
1.2.2 Big Data	3
1.3 Processi aziendali	4
1.3.1 Metodologia di sviluppo	4
1.3.2 Controllo di versione	5
1.3.3 Ambiente di sviluppo	7
1.4 Propensione dell'azienda per l'innovazione	7
2 Lo stage nella strategia aziendale	9
2.1 Aspettative personali	9
2.2 Aspettative aziendali	10
2.3 Presentazione del progetto	12
2.3.1 Analisi ed elaborazione del dataset e dei dati di interesse	12
2.3.2 Realizzazione di una web app per la presentazione dei risultati	14
2.4 Vincoli	14
2.4.1 Vincoli metodologici	14
2.4.2 Vincoli temporali	14
2.4.3 Vincoli tecnologici	15
2.5 Aspettative personali sul progetto di stage	16
3 Resoconto dello stage	17
3.1 Descrizione del progetto	17
3.1.1 Il problema	17
3.2 Studio di Hadoop e dei suoi tools	18
3.2.1 Apache Hadoop	18
3.2.2 Apache Hive, Cloudera Impala e Apache Spark	24
3.3 Elaborazione del dataset di interesse	27
3.3.1 Identificazione tipo e calcolo dati statistici	27
3.3.2 Calcolo della matrice di correlazione	28
3.4 Progettazione e sviluppo della web app	30
3.4.1 Architettura software	31
3.5 Risultati ottenuti	35
Glossario	37

Elenco delle figure

1.1	DevOps lifecycle (Fonte: https://goo.gl/rh31h6)	2
1.2	Esempio di tool Hadoop (Fonte: Cloudera - Data Analyst Training)	4
1.3	Framework to build an MVP (Fonte: https://goo.gl/s8LmEH)	5
1.4	Funzionamento ed operazioni in Git (Fonte: https://goo.gl/m1PVZy)	6
1.5	Attività dell’azienda in ambito big data (Fonte: https://goo.gl/tyLXjo)	7
2.1	RESTful Web Service (Fonte: https://goo.gl/zvrBJz)	12
2.2	Rappresentazione di un cluster Hadoop (Fonte: https://goo.gl/DgYJ4J)	13
2.3	Interfaccia di Cloudera Manager	13
2.4	Diagramma di Gantt dello stage	15
3.1	Illustrazione del problema proposto (Fonte: https://goo.gl/AW22at)	18
3.2	Attori principali del sistema Hadoop (Fonte: Cloudera - Developer Training for Spark and Hadoop)	19
3.3	Divisione di un file in blocchi su HDFS (Fonte: Cloudera - Administrator Training for Apache Hadoop)	19
3.4	Caricamento dei blocchi nei nodi (Fonte: Cloudera - Administrator Training for Apache Hadoop)	20
3.5	Rappresentazione dei nodi in Hadoop (Fonte: Cloudera - Administrator Training for Apache Hadoop)	20
3.6	Esempio di scrittura di un file in HDFS (Fonte: Cloudera - Administrator Training for Apache Hadoop)	21
3.7	Esempio di lettura di un file in HDFS (Fonte: Cloudera - Administrator Training for Apache Hadoop)	21
3.8	Struttura di un cluster che utilizza YARN (Fonte: Cloudera - Administrator Training for Apache Hadoop)	22
3.9	Sequenza avvio applicazione di YARN	23
3.10	Esecuzione di una query (Fonte: Cloudera - Data Analyst Training)	24
3.11	Rappresentazione delle connessioni ad Hive Metastore (Fonte: Cloudera - Administrator Training for Apache Hadoop)	24
3.12	Rappresentazione dei daemons Impala (Fonte: Cloudera - Administrator Training for Apache Hadoop)	25
3.13	Differenza tra l’esecuzione di query in Hive ed Impala (Fonte: - Administrator Training for Apache Hadoop)	25
3.14	Esempio di operazioni in Spark	26
3.15	Esempio del CSV contenente le statistiche delle variabili ottenute	28
3.16	Esempio del CSV contenente la matrice di correlazione tra variabili	29
3.17	Esempio di architettura three-tier generica (Fonte: https://goo.gl/nuFksu)	32

3.18 Esempio di architettura three-tier in Java EE (Fonte: https://goo.gl/S6RYgT)	32
3.19 Flusso di dati in un servlet	33
3.20 Esempio di interazione fra JSP, Servlet ed Enterprise JavaBeans (Fonte: https://goo.gl/bWRWty)	34

Elenco delle tabelle

2.1 Obiettivi dello stage	11
3.1 Requisiti per la web app	31
3.2 Riepilogo soddisfacimento requisiti per la web app	36
3.3 Bilancio soddisfacimento requisiti per la web app	36

Capitolo 1

Il contesto aziendale

1.1 Il profilo aziendale

IT Euro Consulting¹ è un'azienda di medie dimensioni con sede legale a Padova, nata nel 2007 e facente parte del gruppo SCAI, presente su tutto il territorio italiano. Dalla sua nascita si è sempre occupata prevalentemente di consulenza, *System Integration* ed *Application Management*, in ambito ICT, operando in tutti i principali settori di mercato: bancario ed assicurativo, industria, pubblica amministrazione e servizi. Nel corso degli anni l'azienda ha consolidato le proprie conoscenze, offrendo svariati servizi, soprattutto nei seguenti ambiti:

- * **Big Data:** supporto alle aziende nel loro processo di crescita e cambiamento, tramite moderne soluzioni di *Business Intelligence* e la possibilità di prevedere scenari ed eventi futuri e prendere le più opportune decisioni operative o di business grazie all'analisi della gran mole di dati che ogni giorno vengono creati. Vengono quindi offerti servizi di *big data engineer*, *big data scientist*, *big data architect* e *big data administrator*;
- * **Internet of Things:** soluzioni *end-to-end*, basate su tecnologie leader di mercato che consentono di indirizzare in modo efficace la realizzazione di sistemi IoT accelerando la realizzazione di componenti web e mobile per la raccolta, la visualizzazione e l'analisi dei dati;
- * **Reference Architecture:** intesa come *best practice* e struttura di base per un insieme di domini applicativi all'interno di un'organizzazione, la quale agevola il continuo allineamento dei processi e delle strategie con le giuste soluzioni tecnologiche. Vengono quindi offerti servizi di *assessment*, design e consulenza;
- * **DevOps:** automazione delle attività manuali nelle diverse fasi del *Software Development Lifecycle*. Il modello DevOps non si concentra esclusivamente sull'introduzione di nuovi tool, ma è inteso come una combinazione di cultura e processi unita agli strumenti di automazione. Vengono quindi offerti servizi di *assessment* e consulenza;

¹<https://www.itecons.it>.

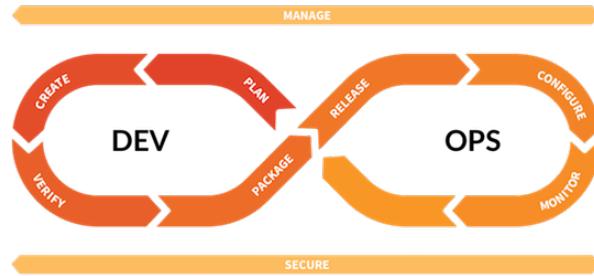


Figura 1.1: DevOps lifecycle (Fonte: <https://goo.gl/rh31h6>)

- * **System Integration:** servizi di consulenza o interventi progettuali per aiutare le aziende a gestire al meglio le proprie strutture tecnologiche complesse e soluzioni applicative per semplificare la coesione fra i vari sottosistemi che compongono la struttura;
- * **Application Management:** servizi di manutenzione correttiva, adattativa ed evolutiva di soluzioni applicative durante il loro intero ciclo di vita;
- * **Customer Relationship Management:** con l'obiettivo di ottenere una visione completa per perseguire uno scenario di **Single Customer View**, abilitante al dialogo *one-to-one* tra l'organizzazione ed il proprio cliente indipendentemente dalle canalità attraverso le quali avviene l'interazione;
- * **System & Data Administration:** servizio consultivo svolto avvalendosi di un insieme di strategie, processi e regole che consentono di gestire i sistemi e trattare i dati fondamentali per lo sviluppo aziendale. Vengono quindi offerti servizi di *database administration*, *database security*, *data governance*, *data analysis* e *scheduling management*.

I clienti principali di IT Euro Consulting sono aziende nazionali ed internazionali che lavorano nei seguenti settori:

- * *Banking*
- * *Insurance*
- * *Telecommunications*
- * *Media & Technology*
- * *Public Administration*
- * *Utilities & Energy*
- * *Manufacturing*

1.2 Tecnologie utilizzate

Le tecnologie utilizzate dall'azienda per la realizzazione dei propri prodotti si possono raggruppare in due macro-sezioni, ovvero riguardanti lo sviluppo software e le attività inerenti ai *big data*. Per quanto concerne la prima, si può suddividere ulteriormente in due aree: *back-end* e *front-end*.

1.2.1 Sviluppo software

Back-end: buona parte del *back-end* dei prodotti dell'azienda è scritta in linguaggio Java, in particolare utilizzando le specifiche fornite dalla versione *Enterprise Edition*². Questa scelta è dovuta al grande supporto offerto da questo linguaggio in fatto di controllo degli accessi e sicurezza di applicativi delicati come quelli in ambito bancario e assicurativo;

Front-end: per quanto riguarda il *front-end*, è utilizzato prevalentemente il framework TypeScript Angular³, in quanto offre una grande elasticità d'impiego e buone prestazioni.

1.2.2 Big Data

Hadoop⁴: framework utilizzato per la gestione del **cluster** e supporta applicazioni distribuite con elevato accesso ai dati, strutturati tramite il *filesystem* chiamato **HDFS**. Permette alle applicazioni di lavorare con migliaia di nodi e petabyte di dati;

Hive⁵: utilizzato per effettuare le *query* e l'analisi preliminare dei dati in *dataset* di grandi dimensioni;

Impala⁶: simile ad Hive, ma fornisce prestazioni leggermente migliori a discapito di una minor affidabilità e peggior gestione degli errori;

Spark⁷: framework per il calcolo distribuito di dati strutturati in un **cluster**. Supporta applicazioni scritte in molteplici linguaggi, quelli utilizzati in azienda sono principalmente Scala e Python;

R⁸: linguaggio di programmazione e ambiente di sviluppo specifico per l'analisi statistica dei dati, utilizzato per la stima di modelli predittivi partendo dai dati ricavati utilizzando i precedenti strumenti;

Python⁹: grazie alla sua elasticità ed ai suoi svariati utilizzi, Python è utilizzato anche per scopi analoghi al precedente strumento. Essendo la sintassi di questo linguaggio molto semplice, è ultimamente preferito a R.

²<https://www.oracle.com/technetwork/java/javaee>.

³<https://angular.io/>.

⁴<https://hadoop.apache.org/>.

⁵<https://hive.apache.org/>.

⁶<https://impala.apache.org/>.

⁷<https://spark.apache.org/>.

⁸<https://www.r-project.org/>.

⁹<https://www.python.org/>.

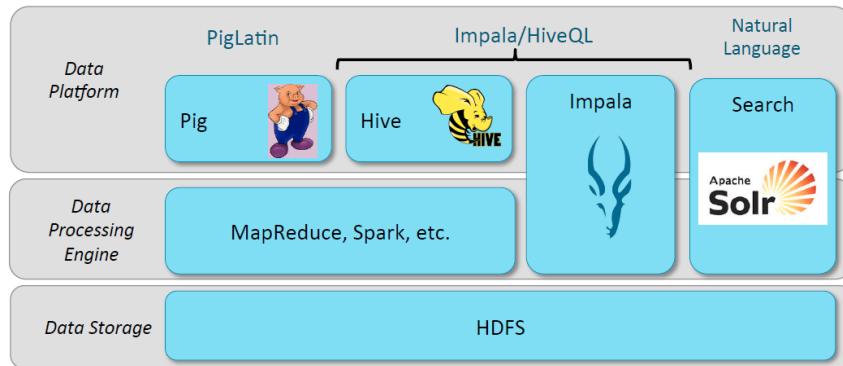


Figura 1.2: Esempio di tool Hadoop (Fonte: Cloudera - Data Analyst Training)

1.3 Processi aziendali

L’azienda svolge il suo operato in base alla tipologia di lavoro da effettuare. Oltre ad eseguire progetti per un possibile cliente, sono attivi progetti per lo sviluppo di nuovo prodotto, dopo aver effettuato le ricerche di mercato d’interesse, da consegnare poi al reparto marketing per trovare compratori interessati; spesso, inoltre, vengono attivati dei progetti in seguito all’interesse relativo ad alcune gare d’appalto, principalmente per il settore privato ma in passato anche per quello pubblico.

1.3.1 Metodologia di sviluppo

Sviluppo su commissione

Per quanto riguarda i progetti relativi allo sviluppo di un nuovo prodotto a seguito di una richiesta da parte del cliente, sotto la supervisione di un *project manager* per coordinare i lavori ed interfacciarsi con il *management* dell’azienda, vengono svolte le seguenti attività:

1. **Analisi:** svolta in contemporanea dagli analisti e dal reparto marketing per il contatto con il cliente. Solitamente si cerca di partire da prodotti già sviluppati in azienda per poi personalizzarli in base alle richieste del cliente, così da poter offrire soluzioni già di base consolidate e testate in molteplici situazioni. Questo è preferibile soprattutto se le applicazioni sottostanti hanno bisogno di una maggior sicurezza, com’è il caso in ambito finanziario, anche per poter effettuare una manutenzione rapida in caso di malfunzionamenti.
Il reparto marketing mantiene la maggior parte delle relazioni con il cliente finale all’inizio del rapporto, anche se un colloquio diretto di tecnici specializzati è ovviamente necessario dopo le prime interazioni per poter adottare soluzioni più specifiche e tecnicamente più complesse in base alle necessità del cliente;
2. **Implementazione:** dopo aver identificato i requisiti assieme al cliente, il team incaricato si occupa dell’implementazione del prodotto concordato. Solitamente, per ogni progetto, sono assegnate un certo numero di persone e risorse per occuparsi del *back-end* e del *front-end* in base alla complessità ed alle tempistiche del progetto; in base alla tipologia ed alla necessità, questi saranno affiancati anche dal team che si occupa di *big data* all’inizio dei lavori per le analisi sui

dati necessari. Ogni team è supervisionato da un *team leader* che mantiene il controllo sull'andamento dei lavori e le relazioni con gli altri team di sviluppo ed il *project manager*;

3. **Rilascio:** dopo un'attenta attività di *testing* interna all'azienda e di collaudo con il cliente, viene rilasciata una versione stabile del prodotto;
4. **Manutenzione:** con il passare del tempo, il prodotto viene mantenuto e aggiornato secondo nuove specifiche del cliente o in seguito a problemi riscontrati, sia sul singolo prodotto sia in caso di problemi in prodotti che condividono la stessa base di partenza e quindi possibili degli stessi errori che potrebbero compromettere la stabilità e la sicurezza.

Sviluppo nuovo prodotto

Nel caso il prodotto non sia precedentemente commissionato da un cliente, le attività che vengono seguite sono leggermente differenti. Il reparto di ricerca e sviluppo, il team *big data* ed il reparto marketing collaborano alla ricerca di un prodotto appetibile per un eventuale cliente a cui verrà in genere presentato solamente un *Minimum Viable Product*: in questo modo è possibile presentare all'interessato un prototipo del prodotto con alcune funzionalità essenziali e significative, senza perdite inutili di tempo e risorse per l'azienda. Le attività di implementazione, rilascio e manutenzione che sono eseguite in seguito all'ottenimento di un cliente interessato sono invece pari a quelle dello *sviluppo su commissione*.

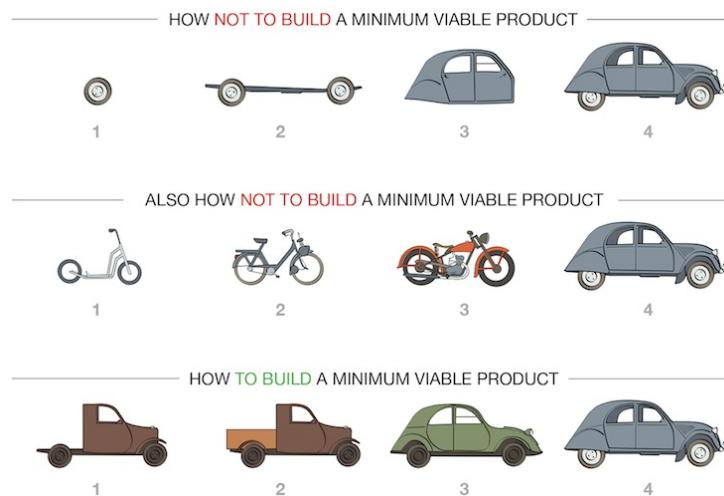


Figura 1.3: Framework to build an MVP (Fonte: <https://goo.gl/s8LmEH>)

1.3.2 Controllo di versione

Come strumento di versionamento del codice, l'azienda utilizza *Git*¹⁰ ed in particolare, per poter gestire tutto il codice derivante dai vari progetti, viene utilizzata la versione

¹⁰<https://git-scm.com/>.

enterprise di [GitLab¹¹](#), disponibile gratuitamente sotto licenza *open source*. Alcuni dei vantaggi che ha spinto l'azienda ad utilizzare Git sono:

- * **Ridondanza**: ogni sviluppatore possiede una copia dell'intera *repository*. Il rischio di perdita dei sorgenti del progetto è quindi inversamente proporzionale al numero di sviluppatori che ne possiedono in locale l'ultima versione; in caso di perdita del progetto di uno sviluppatore quindi si andrà incontro solamente alla perdita delle ultime modifiche personali fatte;
- * **Disponibilità**: anche in assenza di connessione alla *repository* principale, è possibile continuare ad effettuare *commit* ed a salvare le modifiche fatte nel tempo. Una volta ripristinata la connessione, la *repository* locale può essere sincronizzata con quella remota rendendo le modifiche disponibili a tutti;
- * **Branch e Merge**: permette con molta facilità la creazione di *branch*, consentendo di creare quindi delle ramificazioni in cui sviluppare funzionalità non stabili, evitando di intaccare il ramo principale dove di norma risiede una versione stabile e testata del prodotto. Nel momento in cui si vogliono salvare le modifiche anche nel ramo principale, [Git](#) permette di effettuare l'operazione di *merge* del nuovo ramo testato con il ramo principale e considerare quindi le ultime modifiche come stabili;
- * **Fork e Pull request**: queste due operazioni permettono di clonare una *repository* (tramite *fork*) e, successivamente, proporre l'inclusione delle modifiche apportate all'interno del clone nella *repository* originale.

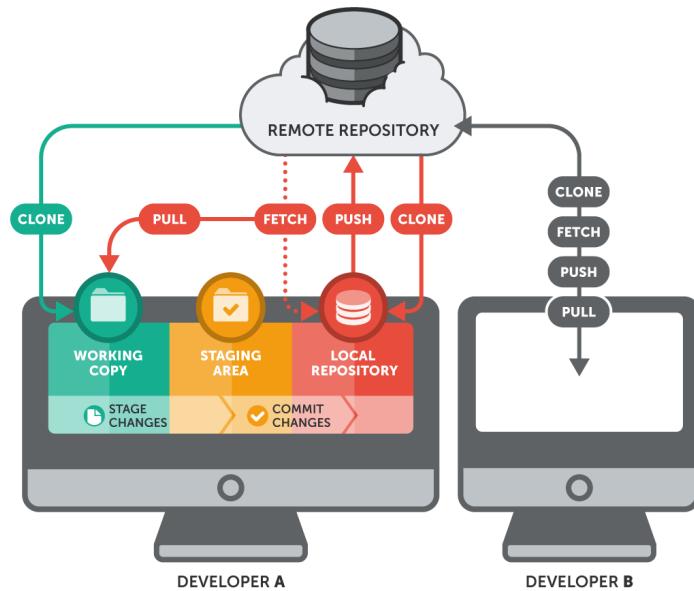


Figura 1.4: Funzionamento ed operazioni in Git (Fonte: <https://goo.gl/m1PVZy>)

¹¹<https://about.gitlab.com/>.

1.3.3 Ambiente di sviluppo

Gli strumenti utilizzati per lo sviluppo, divisi in categorie, sono i seguenti:

- * **IntelliJ IDEA**¹²: è l'**IDE** utilizzato in prevalenza dagli sviluppatori in quanto supporta vari linguaggi di programmazione e vari tool utili;
- * **Visual Studio Code**¹³: come il precedente, supporta vari linguaggi di programmazione e molteplici estensioni volte a migliorare lo sviluppo. A differenza del precedente, però, non supporta nativamente Java ed è quindi meno utilizzato;
- * **Bash**: per effettuare l'analisi dei dati preliminari da parte del team *big data* viene utilizzata la *shell* di Linux per eseguire i tool interessati (ad esempio Hive, Impala, Spark o per visualizzare il *filesystem HDFS*).

1.4 Propensione dell'azienda per l'innovazione

L'azienda è alla continua ricerca di nuove tecnologie e prodotti innovativi che possano soddisfare sempre più le esigenze del cliente. Nel primo caso l'azienda cerca di cogliere il meglio delle nuove tecnologie per poterne trarre il maggior beneficio possibile attraverso progetti sperimentali e, per la prima volta quest'anno, stage universitari. Essendo l'innovazione un importante fattore di crescita, IT Euro Consulting coltiva questo aspetto cercando personale che abbia attitudine al cambiamento e contemporaneamente esprima le proprie soluzioni ai problemi incontrati dando libero sfogo alla propria creatività. Nel secondo caso, invece, l'azienda si prefigge l'obiettivo di creare nuovi prodotti al passo con le esigenze di potenziali clienti, utilizzando le ultime tecnologie disponibili.

Un'ulteriore prova della propensione all'innovazione dell'azienda è la presenza di un team specifico ed in continua espansione per il segmento *big data*, cosa usuale all'estero ma ben più rara in Italia in aziende di medie dimensioni, nelle quali l'importanza dei dati e della potenzialità che essi possiedono non è ancora entrata pienamente nell'ideale di business delle aziende.



Figura 1.5: Attività dell'azienda in ambito big data (Fonte: <https://goo.gl/tyLXjo>)

¹²<https://www.jetbrains.com/idea/>.

¹³<https://code.visualstudio.com/>.

Capitolo 2

Lo stage nella strategia aziendale

2.1 Aspettative personali

La mia esperienza lavorativa si è sempre limitata a piccoli lavori occasionali, un periodo in cui ho svolto un progetto esterno per un'azienda ed in seguito uno stage nella stessa azienda, entrambi durante la scuola secondaria di secondo grado.

Già da queste piccole esperienze, avevo capito quanto la pratica sul campo fosse una valida opportunità per applicare attivamente quanto studiato durante la carriera da studente e, inoltre, apprendere nuove nozioni. Dall'attività di stage, quindi, le mie speranze erano quelle di poter apprendere nuove nozioni e consolidare quelle già in mio possesso tramite un approccio che fosse più pratico di quello che normalmente viene utilizzato all'interno dell'ambiente universitario.

I miei obiettivi iniziali per un progetto di stage erano quindi i seguenti:

- * Entrare in contatto con nuove tecnologie;
- * Vedere e capire il funzionamento di una realtà aziendale in ambito tecnologico ed ICT;
- * Poder lavorare con il supporto di personale qualificato;
- * Vedere in pratica come le nozioni apprese in aula o da studio personale vengono realmente applicate nel mondo lavorativo.

Per far sì che le mie opzioni di scelta fossero quanto più numerose possibili, ho deciso di partecipare all'iniziativa denominata Stage-IT¹ organizzata dall'Università di Padova. Prima di andare all'evento, ho stilato una lista di possibili aziende a cui presentarmi, facendo soprattutto attenzione ai progetti proposti. Durante l'evento ho quindi colto l'occasione per discutere dell'attività di stage con molteplici aziende. Molte di loro operavano in campi che a mio parere non risultavano essere particolarmente interessanti, ma qualche azienda ha colto la mia curiosità e mi ha permesso di avere con loro un colloquio di presentazione reciproca. Alcune di queste erano più interessate a stage per inserimento lavorativo e di durata di circa 6 mesi, che ad uno finalizzato per la tesi, e quindi ho dovuto scartarle a priori nonostante il mio interesse.

¹<http://informatica.math.unipd.it/laurea/stageit.html>.

Fortunatamente, l'azienda IT Euro Consulting proponeva uno stage adeguato per lo sviluppo della tesi e gli argomenti coinvolti erano pienamente di mio interesse: durante il colloquio conoscitivo comunque non mi è stato subito presentato un progetto di stage, ma piuttosto mi è stata esposta la struttura aziendale, la focalizzazione dello stage nell'ambito *big data* e la propensione dell'azienda per l'innovazione e l'interesse a conoscere nuovi punti di vista come quello di uno studente universitario. Tutto ciò mi ha attratto fin da subito e quindi, dopo un paio di incontri successivi avvenuti in azienda, si è proceduto alla presentazione del progetto di stage vero e proprio.

Considerando i miei obiettivi e la lista di aziende con le quali ho avuto un contatto a Stage-IT, ho deciso di svolgere il progetto di IT Euro Consulting, in quanto di maggiore interesse rispetto ai progetti offerti da altre aziende.

2.2 Aspettative aziendali

Quest'anno IT Euro Consulting ha deciso, per la prima volta, di attivare un progetto di stage universitario che non avesse come fine ultimo l'assunzione del tirocinante. I motivi di questa scelta sono molteplici e condivisibili.

In primo luogo è necessario distinguere i percorsi che l'azienda ha deciso di far intraprendere ai diversi tirocinanti in base allo scopo dello stage.

Per gli studenti, appena laureati o laureandi, il cui fine è l'assunzione al termine del tirocinio è previsto un periodo in azienda di circa 6 mesi, durante il quale, dopo il primo periodo formativo, si viene inseriti in progetti già avviati e quindi affiancati dal team a cui è assegnato il lavoro. Negli ultimi 3 anni infatti i reparti sviluppo e *big data* si sono molto ampliati e sono state assunte dall'azienda molteplici persone in seguito ad uno stage o tirocinio. Oggigiorno infatti la maggior parte del personale appartenente ai reparti sviluppo e *big data* è qualificata con un titolo di laurea in Informatica o Ingegneria.

Nell'altro caso, ovvero lo stage curricolare, come di mio interesse, la durata è di circa 300-350 ore e si distingue in una fase di apprendimento ed in una fase di progetto, sempre affiancati da uno o più tutor interni. Durante il progetto, si concede al tirocinante maggiore libertà sulla scelta delle tecnologie da utilizzare e su decisioni progettuali. Queste poi vengono discusse con il tutor in modo da correggere eventuali scelte errate frutto dell'inesperienza del tirocinante.

I vantaggi di questi due percorsi sono in parte comuni: in primo luogo si ha l'inserimento in organico di nuove risorse provenienti dal mondo universitario. Assumere anche solo provvisoriamente una figura per uno stage proveniente dal mondo universitario giova all'azienda in quanto il personale ha la possibilità di confrontarsi ed aggiornarsi con costui sui nuovi insegnamenti e corsi universitari. Questa vicinanza è però anche utile per lo studente, in quanto ha la possibilità di vedere concretamente come quanto appreso in aula sia implementato effettivamente nel mondo reale e di capire come funzioni realmente un'azienda, se non ha già avuto esperienze simili durante la sua carriera. Il secondo motivo è la possibilità per l'azienda di comprendere il livello di preparazione medio degli studenti universitari ed essere attivi nello scrutare quelli più meritevoli che possono portare ad un vantaggio competitivo considerevole. Per quanto riguarda il mio percorso di stage, quello curricolare, si possono riconoscere altri due vantaggi ed entrambi derivano dalla maggior libertà che si concede allo studente per risolvere problemi che solitamente in azienda si risolverebbero tramite soluzioni già consolidate. La possibilità di sfruttare nuove tecnologie senza la necessità di perdita di tempo del personale aziendale, permette di rimanere aggiornati sul continuo rilascio di

librerie e *framework* che potrebbero portare benefici ai vari team in termini di efficienza, sia sul tempo di sviluppo del software, sia per quanto riguarda le performance del prodotto stesso. La maggiore libertà sulla progettazione e lo sviluppo concessa allo stagista consente anche di valutare vantaggi e svantaggi di soluzioni architetturali che si erano scartate prematuramente o che non erano state considerate a priori.

Alla fine dello stage, l'azienda si aspettava di avere una [web app](#) che esponesse i dati precedentemente recuperati dal [cluster](#), esaminati ed elaborati in modo da facilitare il successivo sviluppo del modello per stimare il target desiderato.

Tutto ciò mi ha portato prima dell'inizio dell'attività di stage, a concordare insieme al tutor aziendale i requisiti del progetto, che durante lo stage sono rimasti fedeli al Piano di Lavoro, senza dover effettuare cambiamenti imprevisti. Per poter gestire al meglio lo stage ed avere una buona elasticità sugli obiettivi da raggiungere, questi sono stati divisi in tre categorie: obiettivi obbligatori, obiettivi desiderabili e obiettivi facoltativi.

Tabella 2.1: Obiettivi dello stage

Obiettivi obbligatori
Caricare, estrarre e compiere semplici operazioni su file presenti nel cluster
Svolgere semplici operazioni con Hive/Impala su tabelle già esistenti, con creazione di nuove tabelle contenenti statistiche riassuntive e porzioni di dati
Svolgimento di una semplice analisi dati con Spark, dall'ingestione all'esame del <i>dataset</i> , salvando i risultati in formato tabellare CSV
Sviluppo di un Web Service RESTful in grado di presentare in formato JSON i risultati CSV
Obiettivi desiderabili
Creazione di nuove tabelle tramite l'utilizzo di <i>query</i> SQL innestate e con operazioni complesse
Sviluppo di una semplice applicazione capace di interfacciarsi con i dati prodotti da Spark e riassumere graficamente i risultati
Sviluppo di un'applicazione Java EE <i>3-tier</i> , composta da un'interfaccia grafica HTML5/Angular in grado di visualizzare i dati dei risultati in modalità grafica e un Web Service RESTful per la presentazione dei dati in formato JSON recuperati con Hive/Impala

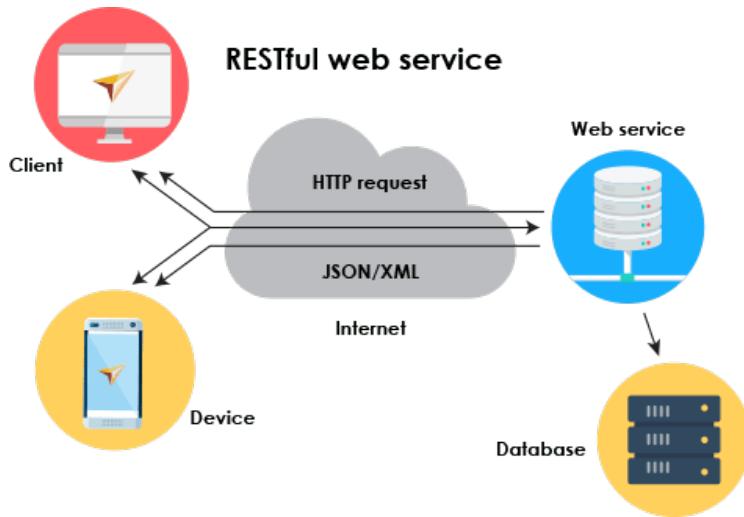


Figura 2.1: RESTful Web Service (Fonte: <https://goo.gl/zvrBJz>)

2.3 Presentazione del progetto

Il progetto di stage oggetto di questa relazione è stato di natura esplorativa. Lo scopo dello stage è diviso in due parti:

- * Analisi ed elaborazione del *dataset* e dei dati di interesse;
- * Realizzazione di una [web app](#) per la presentazione dei risultati ottenuti.

La prima attività era già stata compiuta dal team *big data* circa 3 anni fa in occasione di un concorso a cui l’azienda aveva partecipato. Grazie a questo stage, è stato dunque possibile per l’azienda esplorare l’utilizzo di metodi differenti per l’elaborazione e l’analisi dei dati rispetto a quelli utilizzati in precedenza. Inoltre la [web app](#) sviluppata può essere riutilizzata come semplice interfaccia per dati e risultati finali di successivi progetti.

2.3.1 Analisi ed elaborazione del dataset e dei dati di interesse

Nella prima parte dello stage gli obiettivi erano molteplici.

In primo luogo l’azienda richiedeva allo stagista uno studio teorico sull’architettura del [cluster](#), sul funzionamento di Hadoop e la struttura di [HDFS](#), così da comprendere in pieno il funzionamento del sistema sottostante e di YARN, il tool che gestisce le risorse ed effettua lo *scheduling* dei processi.

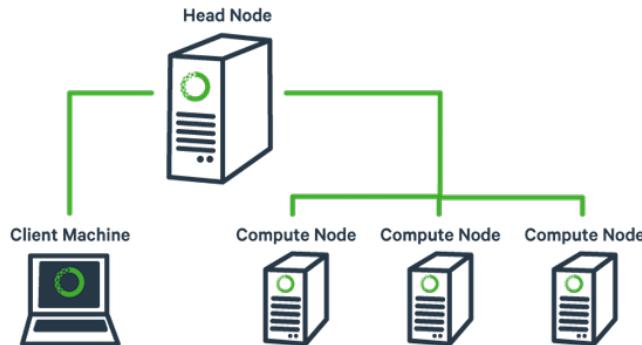


Figura 2.2: Rappresentazione di un cluster Hadoop (Fonte: <https://goo.gl/DgYJ4J>)

Il secondo obiettivo era lo studio dei tool principali che il team *big data* utilizza per l'analisi ed il *processing* dei dati, ovvero Hive, Impala, Spark. Oltre a questi tool di cui si è già dato un breve commento, l'azienda richiedeva lo studio di:

- * **Hue²**: un'applicazione che offre un'interfaccia e semplifica alcune operazioni, come le *query* su Hadoop, di cui è utile la conoscenza di base ma scarsamente utilizzato in quanto risulta relativamente complesso per utilizzi semplici;
- * **Cloudera Manager³**: una web app per la gestione dei servizi e dei tool di Hadoop. Permette di tenere sotto controllo il cluster, quindi le risorse disponibili e gli utenti collegati, e permette di gestire e visualizzare le attività dei tool attualmente in esecuzione e passati. È utilizzato principalmente dai *system administrators* ma si rivela utile anche per il team *big data* per un'analisi veloce del funzionamento *real time* del sistema.

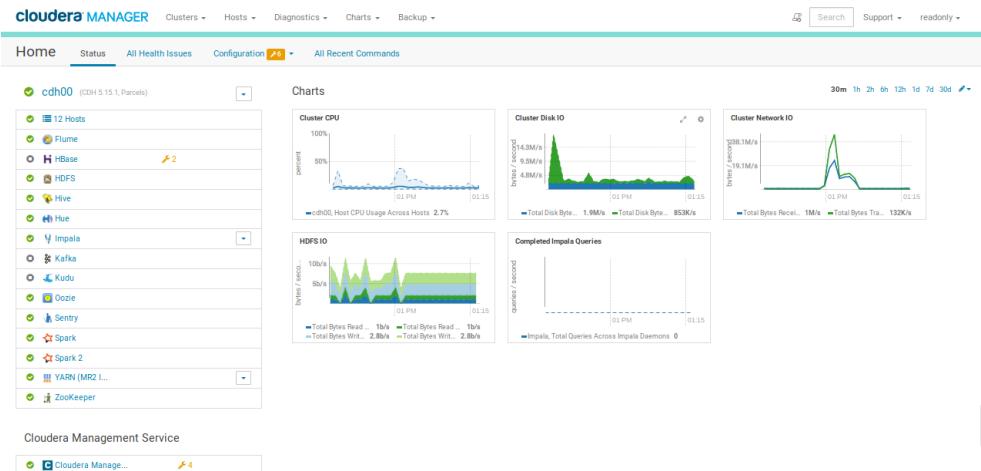


Figura 2.3: Interfaccia di Cloudera Manager

²<http://gethue.com/>.

³<https://www.cloudera.com/products/product-components/cloudera-manager.html>.

Al termine dello studio di questi tool, l'azienda richiedeva la comprensione e la successiva analisi del *dataset* oggetto del progetto, che verrà trattata in dettaglio nel capitolo successivo.

2.3.2 Realizzazione di una web app per la presentazione dei risultati

Dopo aver ottenuto tutti i dati di interesse, l'azienda richiedeva lo sviluppo di una [web app](#) per la presentazione dei risultati in maniera più comprensibile ed appetibile rispetto a tabelle [CSV](#) grezze. Grazie all'utilizzo di grafici, anche semplici, è infatti possibile notare alcune caratteristiche del *dataset* che potrebbero rivelarsi utili ad una successiva analisi per la creazione del modello statistico.

2.4 Vincoli

2.4.1 Vincoli metodologici

Insieme con l'azienda, abbiamo deciso che le attività di stage sarebbero dovute essere svolte presso la sede della società. Questo è stato concordato con lo scopo di favorire il dialogo tra me ed il tutor aziendale e di avere la possibilità di confrontarmi direttamente con programmatore ed analisti più esperti in caso di problematiche durante lo svolgimento delle attività di progettazione, analisi e sviluppo software. Oltre a ciò, l'azienda richiedeva che, al termine di ogni settimana lavorativa, venisse compilato un rapporto di quelle che erano state le attività svolte durante tale settimana. Inoltre, nel corso del primo periodo, prettamente di formazione, l'azienda richiedeva un breve resoconto di quanto compreso, cosicchè, in caso di dubbi, questi venissero risolti da personale esperto prima di passare alla progettazione. In seguito ad ogni rapporto, ed in particolare agli avanzamenti ed ai problemi incontrati descritti in esso, sarebbe stato deciso cosa doveva essere svolto la settimana successiva, così da poter adattare al meglio il progetto al periodo di stage rimanente.

Al termine di tutte le attività, l'azienda richiedeva inoltre una breve presentazione di quanto svolto ad alcuni membri del management. Tale presentazione, esposta in forma verbale, sarebbe servita per illustrare ciò che si era concluso tramite il progetto, elencando pro e contro della soluzione trovata.

2.4.2 Vincoli temporali

Lo svolgimento dello stage prevedeva una durata di 320 ore complessive di lavoro. Queste ore sono state distribuite in modo uniforme in otto settimane lavorative da 40 ore ciascuna. L'orario accordato tra me e il tutor aziendale è stato dal Lunedì al Venerdì dalle 09:00 alle 18:00 con un'ora di pausa pranzo. Prima dell'inizio dello stage il tutor ha redatto nel Piano di Lavoro una scansione temporale delle attività su base settimanale. In alcune occasioni, il lavoro assegnato è stato portato al termine in anticipo, per cui si è scelto di effettuare alcuni approfondimenti su argomenti che mi interessavano maggiormente, tramite lo studio autonomo ma con la possibilità di richiedere chiarimenti al personale più esperto che mi seguiva. La suddivisione del lavoro su base settimanale è stata così ripartita:

- * **Prima settimana:**

- consolidamento utilizzo sistema Unix;

- comprensione mondo *big data*;
- Apache Hadoop Architecture e **HDFS**.

* **Seconda settimana:**

- approfondimenti mondo *big data*;
- apprendimento comandi **HDFS**;
- comprensione tools Cloudera.

* **Terza settimana:**

- approfondimento tools Cloudera;
- studio ed esercitazione di Cloudera Impala ed Apache Hive.

* **Quarta settimana:**

- studio ed esercitazione di Apache Spark;
- comprensione di Cloudera Manager.

* **Quinta e sesta settimana:**

- ripasso Java con attenzione all'ambiente Java EE;
- studio del linguaggio scelto per il *front-end*.

* **Settima e ottava settimana:**

- applicazione delle principali tecnologie apprese al progetto.

Da questa suddivisione deriva il seguente **Diagramma di Gantt**.

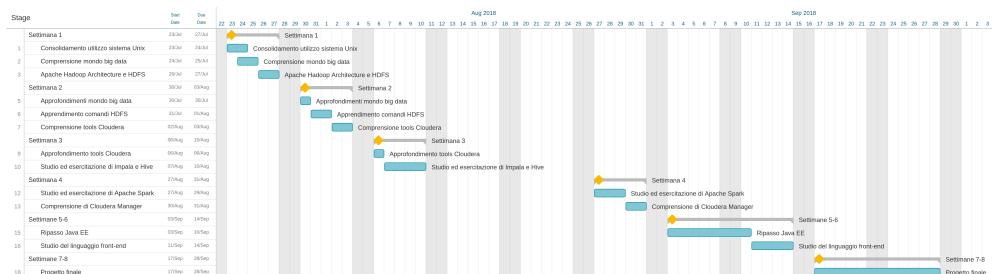


Figura 2.4: Diagramma di Gantt dello stage

2.4.3 Vincoli tecnologici

L'azienda, ad inizio stage, ha posto un unico vincolo per il progetto finale: utilizzare Java EE per la parte di *back-end* del prodotto. Per la parte di *front-end* era invece suggerito l'utilizzo di Angular, in quanto già assiduamente testato ed utilizzato per altri progetti, ma non c'era nessun tipo di vincolo, infatti la scelta finale dipendeva dalle preferenze ed esperienze del tirocinante. Un'alternativa da me proposta è stata l'utilizzo delle librerie React⁴ e Redux⁵ in quanto già in parte conosciute ed utilizzate dal sottoscritto. La scelta finale però è stata, come proposto dall'azienda, Angular, in

⁴<https://reactjs.org/>.

⁵<https://redux.js.org/>.

quanto mi interessava come tecnologia e, vista la natura esplorativa di nuove tecnologie e strumenti dello stage, è stata considerata la scelta migliore per il mio accrescimento culturale.

Per quanto riguarda la parte di *big data*, mi sono affidato completamente al mio tutor aziendale, in quanto non avevo alcuna esperienza in quel campo e quindi non avrei potuto scegliere cosa fosse la scelta migliore per me. Oltre a ciò, gli strumenti utilizzati assieme ad Hadoop sono ormai consolidati e non c'è una grande varietà, quindi la scelta proposta dal personale aziendale si è rivelata obbligatoria.

Oltre a ciò, mi è stata data la possibilità di utilizzare [Git](#) per il versionamento del codice, che ho ben apprezzato ed utilizzato.

2.5 Aspettative personali sul progetto di stage

Successivamente al mio impegno con l'azienda ed alla stesura del Piano di Lavoro, assieme alla definizione degli obiettivi, la mia curiosità verso l'argomento di stage si è intensificata.

Le aspettative che maggiormente sentivo erano:

- * Mettersi in gioco in un'azienda con partner di un certo livello nel mio campo di studi;
- * Instaurare con il personale discussioni su esperienze e punti di vista diversi sulle varie tecnologie;
- * Entrare in contatto con tecnologie nuove e sempre più di largo utilizzo;
- * Conoscere il funzionamento di uno strumento come Hadoop e tutti i tool inerenti;
- * Apprendere come effettuare un'analisi su un insieme di dati distribuiti in un [cluster](#);
- * Imparare come progettare e quali sono le *best practices* per realizzare una [web app](#) utilizzando Java EE.

Capitolo 3

Resoconto dello stage

3.1 Descrizione del progetto

Il *dataset* da elaborare ed analizzare è parte di un concorso a cui l'azienda aveva già partecipato qualche anno fa: questa competizione è stata indetta da BNP Paribas Cardif, il polo assicurativo del Gruppo BNP Paribas, e pubblicata su Keggle¹, nota piattaforma in cui è possibile esporre i propri progetti, visualizzare quelli altrui e proporre sfide in ambito *data science* e *machine learning*.

3.1.1 Il problema

In particolare, il problema proposto, "BNP Paribas Cardif Claims Management"², consiste nella possibilità di classificare le pratiche assicurative in modo che queste possano essere risolte nel minor tempo possibile. A tal proposito, si chiede di prevedere la categoria di un sinistro sulla base delle caratteristiche disponibili nelle prime fasi del processo assicurativo; le due categorie di richieste di indennizzo, su cui basare la classificazione, corrispondono quindi a:

- * Quelle per le quali l'approvazione può essere accelerata, con conseguente maggiore rapidità nel rimborso e minori pratiche da gestire;
- * Quelle per le quali sono richieste informazioni supplementari prima dell'approvazione e del rimborso.

¹<https://www.kaggle.com/>.

²<https://www.kaggle.com/c/bnp-paribas-cardif-claims-management>.

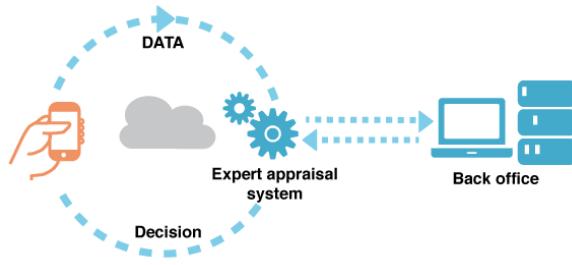


Figura 3.1: Illustrazione del problema proposto (Fonte: <https://goo.gl/AW22at>)

Nella sezione [Elaborazione del dataset di interesse](#) verrà trattata la struttura del *dataset* più nel dettaglio.

3.2 Studio di Hadoop e dei suoi tools

Prima di cominciare a lavorare sul *dataset* del progetto, era necessario studiare la teoria, in quanto la prima parte dello stage considerava argomenti a me quasi totalmente sconosciuti. Autonomamente, ma sempre supervisionato dal tutor aziendale, disponibile a risolvere ogni mio dubbio, ho studiato il materiale necessario per poter eseguire poi al meglio la parte pratica. Oltre a ciò, nel corso della giornata lavorativa, il tutor mi sottoponeva delle esercitazioni da svolgere per consolidare i concetti appresi e risolvere tempestivamente miei eventuali dubbi prima di procedere con gli argomenti successivi. Essendo [HDFS](#) e molti dei tool Hadoop eseguibili principalmente tramite [Bash](#), come prima cosa il tutor mi ha assegnato lo studio autonomo di alcuni capitoli del libro "Learning the bash Shell"³ per ottenere le basi che mi permettessero di utilizzare i comandi che avrei utilizzato in seguito per l'utilizzo dei tool Hadoop.

Dopo aver assorbito i concetti, già in parte di mia conoscenza, il tutor mi ha esposto la struttura del [cluster](#) Hadoop in cui risiedevano i dati e venivano eseguiti i *task*.

3.2.1 Apache Hadoop

Ogni giorno si generano petabytes di dati che, se processati ed analizzati a dovere possono offrire informazioni con un alto valore strategico per un'azienda. Hadoop nasce dall'esigenza di dover gestire e processare questi dati in modo veloce, tramite una soluzione che sia il più possibile economica e scalabile orizzontalmente: aggiungendo nuovi nodi al [cluster](#), la capacità e le performance di questo infatti aumentano proporzionalmente. Per aumentare ulteriormente le prestazioni e la scalabilità del sistema, Hadoop cerca di elaborare i dati sullo stesso nodo in cui questi risiedono: questo permette di ridurre al minimo la *cross-communication* fra i nodi e la necessità di copiare grandi quantità di dati fra questi, eliminando il rischio di *bottleneck* dovuto dalla velocità di trasmissione dei dati. Per gestire il sistema, Hadoop si basa su:

- * [HDFS](#): per la gestione dei dati persistenti;
- * [YARN](#): per lo *scheduling* dei processi (*jobs*) e la gestione delle risorse.

³<http://shop.oreilly.com/product/9780596009656.do>.

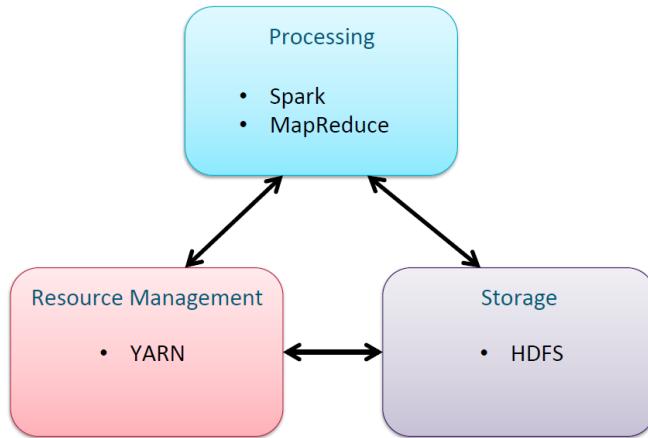


Figura 3.2: Attori principali del sistema Hadoop (Fonte: Cloudera - Developer Training for Spark and Hadoop)

HDFS

Hadoop Distributed File System (HDFS) è un *file system* scritto in Java, basato su Google File System⁴.

Offre performance migliori con un modesto numero di file di grandi dimensioni piuttosto che miliardi di file frammentati, per questo motivo, anche le operazioni di *read*, sono ottimizzate per la lettura in *streaming* piuttosto che quelle casuali. Inoltre, i file sono tutti *write-once* e quindi non modificabili una volta memorizzati. In scrittura, infatti, i dati sono suddivisi in blocchi di dimensione fissata e distribuiti tra i nodi in modo ridondante per prevenire la perdita di informazioni nel caso un nodo non fosse più disponibile in seguito.

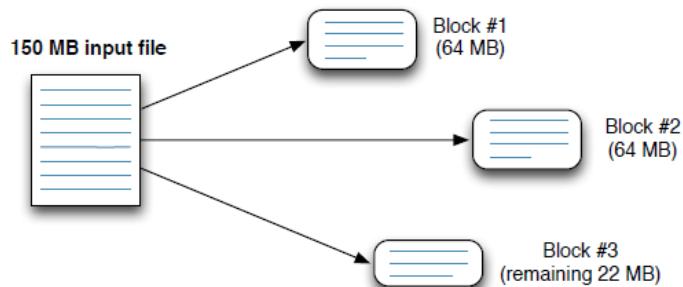


Figura 3.3: Divisione di un file in blocchi su HDFS (Fonte: Cloudera - Administrator Training for Apache Hadoop)

⁴<https://ai.google/research/pubs/pub51>.

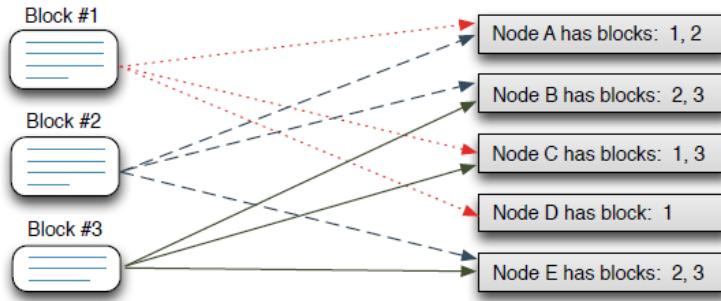


Figura 3.4: Caricamento dei blocchi nei nodi (Fonte: Cloudera - Administrator Training for Apache Hadoop)

Hadoop ha un'architettura di tipo *master-slave*, ovvero in cui il processo *master* ha il controllo su quello *slave*. I nodi, quindi, possono essere di tre tipi:

- * **NameNode**: costituisce il *master daemon*, quindi gestisce tutti i metadati, le informazioni riguardo l'*ownership* e i permessi ad una risorsa, i nomi dei blocchi e la loro locazione. Essendo unico e mantenendo la struttura del *file system*, rappresenta un [single point of failure](#) in HDFS;
- * **DataNode**: costituisce gli *slave daemon*, quindi i nodi che contengono i blocchi di dati veri e propri;
- * **Secondary NameNode**: esegue elaborazioni in supporto al NameNode. Non è un nodo di backup ma la funzione principale è quella di memorizzare una copia del file *FsImage* e di modificare il file di log. *FsImage* contiene un'istantanea dei metadati del *file system* HDFS in un certo momento e *EditLog* è il log delle transazioni che contiene record per ogni modifica dei metadati del *file system*. In questo modo, in qualunque momento, è possibile ricostruire il NameNode a partire da *FsImage* e applicando il record delle transazioni *EditLog*.

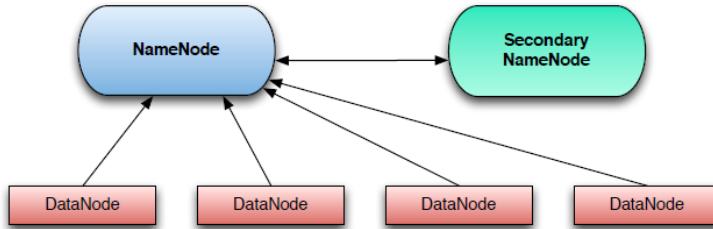


Figura 3.5: Rappresentazione dei nodi in Hadoop (Fonte: Cloudera - Administrator Training for Apache Hadoop)

Nei seguenti esempi sono rappresentati semplici operazioni di scrittura e lettura sui nodi HDFS. La procedura di scrittura di un blocco avviene nel seguente modo:

1. Il client si connette al NameNode;
2. NameNode registra i metadati del file e ritorna il nome del blocco e la lista dei DataNodes al client;

3. Il client si connette al primo DataNode e comincia ad inviare i dati;
4. A sua volta, il primo DataNode si connette al secondo e invia a sua volta i dati;
5. Allo stesso modo, il secondo DataNode si connette al terzo;
6. Ad ogni blocco scritto, al client viene ritornato un *ack packets* dalla *pipeline* di nodi;
7. Una volta ricevuti tutti gli *ack packets*, il client informa il NameNode del completamento della scrittura.

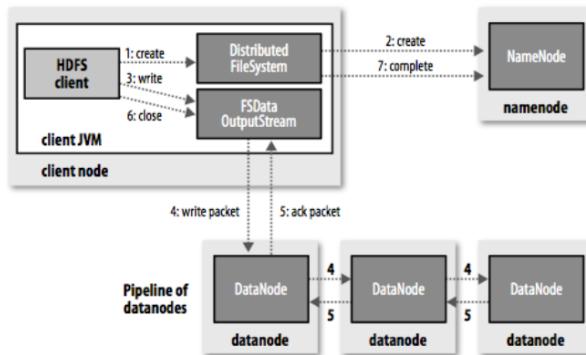


Figura 3.6: Esempio di scrittura di un file in HDFS (Fonte: Cloudera - Administrator Training for Apache Hadoop)

La procedura di lettura di dati avviene nel seguente modo:

1. Il client si connette al NameNode;
2. NameNode ritorna il nome e la locazione dei blocchi del file;
3. Il client si connette ai DataNodes comunicati e legge i blocchi.

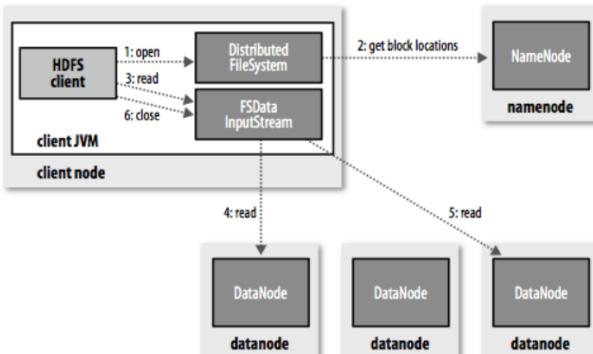


Figura 3.7: Esempio di lettura di un file in HDFS (Fonte: Cloudera - Administrator Training for Apache Hadoop)

YARN

YARN (*Yet Another Resource Negotiator*) è il manager delle risorse di Hadoop. I principali attori del sistema sono:

- * **ResourceManager**: uno per *cluster*, e costituisce il *master daemon*; inizializza le applicazioni e ne pianifica l'utilizzo delle risorse sugli *slave nodes*. È costituito da due componenti principali: uno *scheduler* responsabile per l'allocazione delle risorse delle applicazioni al loro avvio ed un *application manager* che gestisce le applicazioni già in esecuzione sul *cluster*;
- * **NodeManager**: uno per *slave nodes* e costituisce lo *slave daemon*; avvia i processi delle applicazioni e gestisce le risorse sul singolo *slave node*;
- * **JobHistoryServer**: uno per *cluster*, archivia i log ed i *metadata* dei *jobs* terminati;
- * **ApplicationManager**: uno per applicazione, negozia le risorse con il ResourceManager e lavora con il NodeManager; gestisce l'intera vita dell'applicazione, dalla sua inizializzazione alla terminazione.

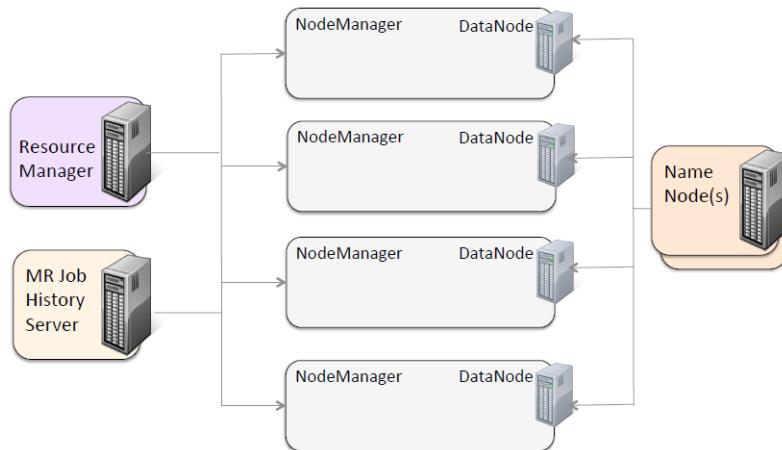
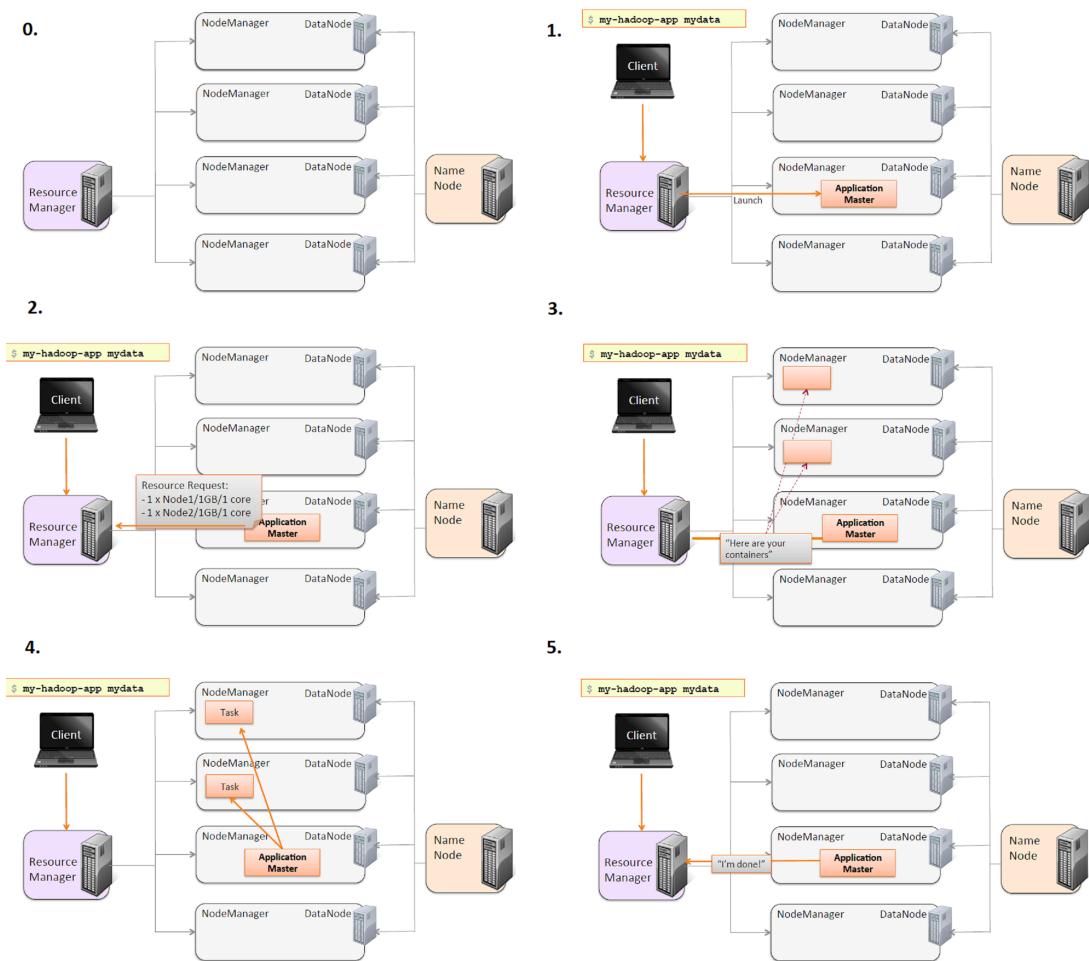


Figura 3.8: Struttura di un cluster che utilizza YARN (Fonte: Cloudera - Administrator Training for Apache Hadoop)

YARN, per avviare un'applicazione sul *cluster*, si comporta nel modo seguente:

1. Il client richiede di eseguire un *job* al ResourceManager che inizializza un *container*, ovvero un numero finito di risorse di un nodo, e avvia l'ApplicationManager sul NodeManager;
2. L'ApplicationManager richiede le risorse necessarie per eseguire tutti i *tasks* al ResourceManager;
3. Il ResourceManager alloca nuovi *containers* su altri nodi del *cluster* e comunica all'ApplicationManager la loro locazione;
4. L'ApplicationManager esegue i *tasks* nei *containers* allocati precedentemente;
5. Una volta completato il *job*, l'ApplicationManager segnala al ResourceManager la conclusione del processo.

**Figura 3.9:** Sequenza avvio applicazione di YARN

3.2.2 Apache Hive, Cloudera Impala e Apache Spark

Apache Hive

Originariamente sviluppato da Facebook per facilitare le interrogazioni di *dataset* su Hadoop, fornisce un metodo per effettuare *query* in HDFS utilizzando un linguaggio simile ad SQL chiamato HiveQL. Le *query* sono poi convertite da Hive in *jobs* che vengono eseguiti da YARN. A causa di questa conversione, i risultati di una *query* sono lenti, infatti ottenere un risultato può richiedere anche minuti se non ore. Inoltre, come detto in precedenza, in HDFS non è possibile modificare i dati, e quindi le operazioni di UPDATE e DELETE, tipiche di SQL, non sono supportate.

Hive interpreta tutti i file di una directory HDFS come contenuti di una tabella e salva le informazioni su come le righe e le colonne della tabella sono delimitate in una locazione, che può essere sulla macchina locale dell'utente o condivisa fra più utenti, chiamata **Hive Metastore**.

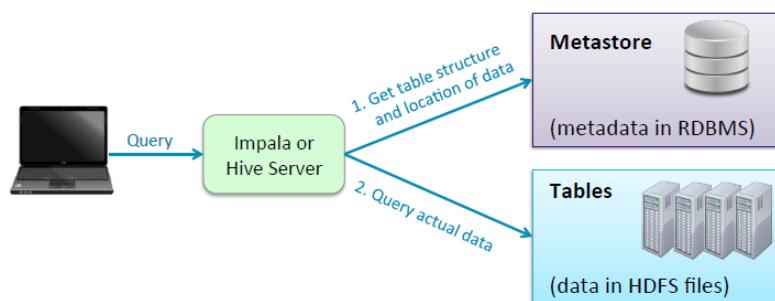


Figura 3.10: Esecuzione di una query (Fonte: Cloudera - Data Analyst Training)

Nel caso Hive Metastore sia condiviso, il client si connette ad un altro attore, chiamato **Hive Metastore Service** utilizzando le [API](#) di Apache Thrift⁵. A sua volta, questo si connette al Metastore attraverso [Java JDBC](#).



Figura 3.11: Rappresentazione delle connessioni ad Hive Metastore (Fonte: Cloudera - Administrator Training for Apache Hadoop)

Cloudera Impala

Impala è derivato da Hive e, come quest'ultimo, permette di effettuare *query* su HDFS utilizzando HiveQL. Inoltre, eredita l'utilizzo dello stesso Metastore condiviso per i metadati delle tabelle ma, a differenza di Hive, non converte le *query* in *jobs*. Infatti le *query* di Impala vengono eseguite su un ulteriore insieme di *daemon*, indicati come **Impala Servers**, sul *cluster* Hadoop. Questo permette ad Impala di eseguire *query* molto più velocemente di Hive ma, in contrapposizione, non supporta tutti i tipi

⁵<https://thrift.apache.org/>.

complessi e le operazioni di Hive.

Gli Impala Servers risiedono su ogni DataNode dell'applicazione, mentre **Impala State Store Server** e **Impala Catalog Server**, che hanno lo scopo di gestire gli Impala Servers, sono unici nel cluster e sono tipicamente locati sul NameNode.

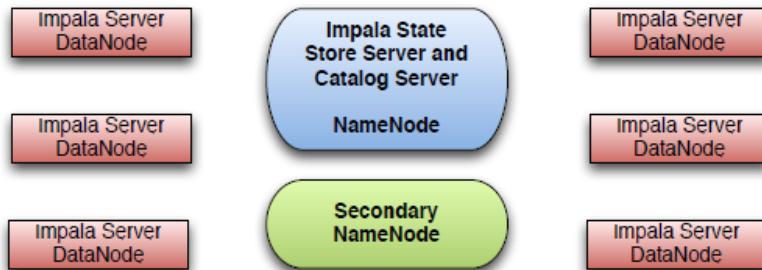


Figura 3.12: Rappresentazione dei daemons Impala (Fonte: Cloudera - Administrator Training for Apache Hadoop)

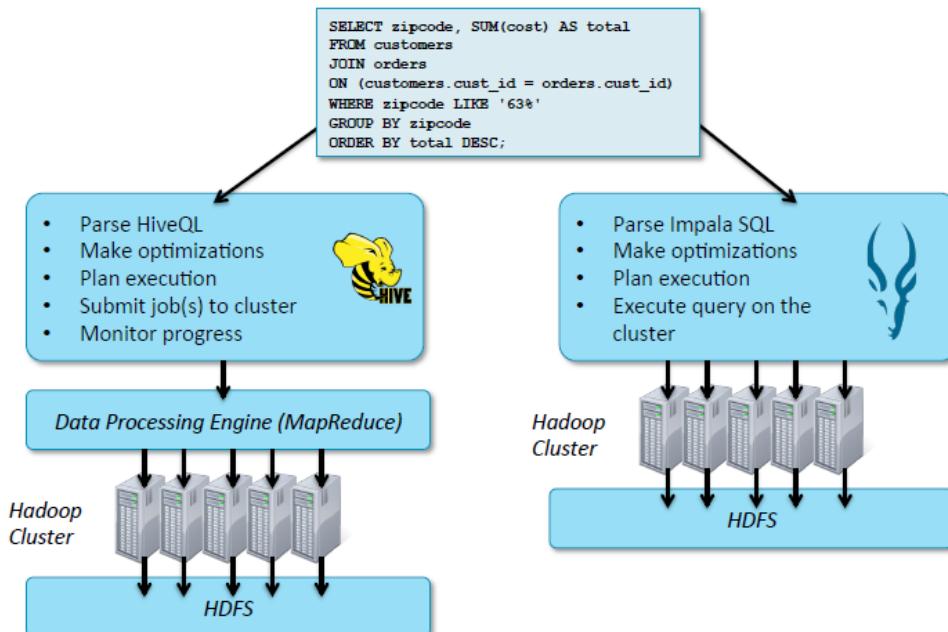


Figura 3.13: Differenza tra l'esecuzione di query in Hive ed Impala (Fonte: - Administrator Training for Apache Hadoop)

Apache Spark

Spark è un *framework* scritto in Scala⁶ per il *processing* dei dati in *dataset* di grandi dimensioni. È possibile eseguire attività di Spark tramite la **Bash** o tramite applicazioni scritte in linguaggio Python, Scala o Java e poi eseguite su Hadoop. La maggior parte

⁶<https://www.scala-lang.org/>.

delle operazioni vengono eseguite tramite l'utilizzo di **RDD** (Resilient Distributed Dataset), che sono la rappresentazione più semplice dei dati in Spark e sono immutabili. Su di un RDD sono possibili due tipi di operazioni:

- * **Actions**: semplici attività terminali che restituiscono un valore, come *count*, *take* o *collect*;
- * **Transformations**: tramite le quali viene creato un nuovo RDD partendo da quello su cui è invocata l'operazione. Questo tipo di attività è necessario se si vuole modificare l'RDD di partenza, essendo questo immutabile. Operazioni tipiche sono *map* e *filter*.

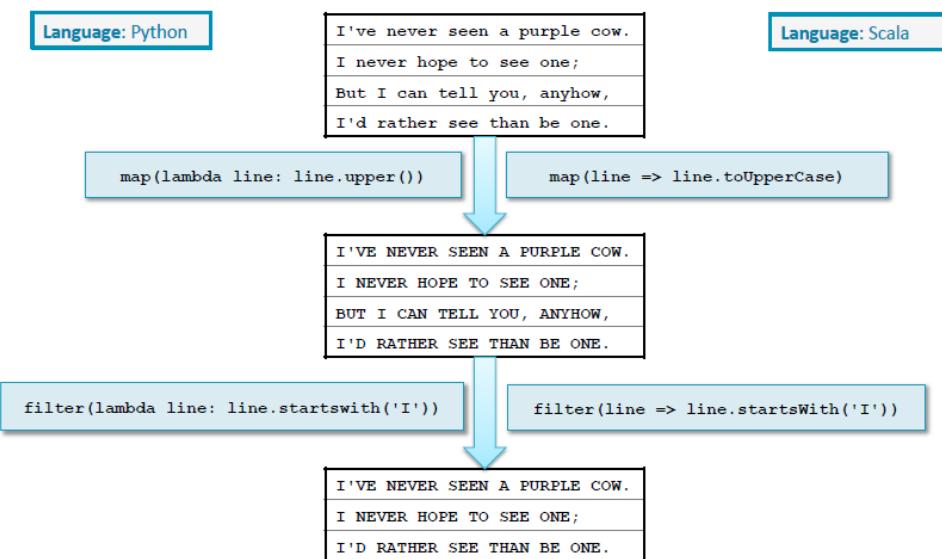


Figura 3.14: Esempio di operazioni in Spark

L'esecuzione delle attività negli RDD è *lazy*, ovvero la *pipeline* di operazioni non viene eseguita finché non incontra un'operazione terminale data da una *action*. Inoltre, durante l'esecuzione della *pipeline* delle operazioni, non viene salvato alcun risultato intermedio finché non viene eseguita un'operazione terminale.

Un altro tipo di struttura dati, oltre ad RDD, supportata da Spark sono i **Data-Frame**: questi sono *dataset* organizzati con uno schema e quindi con colonne nominali. Come risultato si ottengono delle tabelle simili a quelle di un database relazionale ma con ulteriori ottimizzazioni per rendere le operazioni su di questi molto più veloci. Per eseguire queste operazioni, Spark mette a disposizione Spark SQL, un modulo per il *processing* dei dati strutturato, basato sul linguaggio SQL.

3.3 Elaborazione del dataset di interesse

Il dataset oggetto del concorso⁷ è composto da due parti:

- * "train.csv": che contiene i dati su cui effettuare le analisi e su cui basare il modello statistico;
- * "test.csv": che contiene i dati su cui verranno testati i modelli stimati dai partecipanti, ed in base a cui si dichiarerà il vincitore del concorso.

Il *dataset* è anonimo, in quanto contiene dati sensibili provenienti da clienti di BNP Paribas Cardif, e contiene dati sia di tipo categoriale che numerico. È composto da 132 colonne e 114321 righe a cui si aggiunge una colonna, "target", per il *dataset* train, che contiene il risultato su cui stimare il modello. Oltre alla colonna "ID", che identifica in modo univoco l'ennupla, non si ha alcun'altra informazione sui valori.

Per effettuare le operazioni sul *dataset*, che al momento del mio utilizzo era già presente nel *cluster* dell'azienda, è stato utilizzato solamente Spark. Nello specifico, ho utilizzato il linguaggio Scala per scrivere la *pipeline* di operazioni da eseguire sul *cluster*. È stato possibile utilizzare solamente Spark e non Hive o Impala in quanto, utilizzando i DataFrame come spiegato nella sezione precedente, è possibile utilizzare Spark SQL per effettuare le *query* sui dati in modo semplice con un linguaggio simile ad SQL. Questo mi ha permesso di concentrarmi solamente sulla scrittura della pipeline di operazioni in Scala, da cui poi si è ricavato il *JAR* eseguito sul *cluster*. Le operazioni di elaborazione del *dataset* richieste sono:

1. Identificazione del tipo (intero, double o stringa) della colonna e calcolo di statistiche di interesse sui valori contenuti;
2. Calcolo della matrice di correlazione fra le varie colonne del *dataset*.

Tutte le operazioni sui dati sono state effettuate sul *dataset* "train.csv" in quanto soggetto per la stima del modello.

3.3.1 Identificazione tipo e calcolo dati statistici

Per quanto riguarda il primo punto, identificare il tipo delle colonne, ho preventivamente escluso i valori vuoti o nulli. In seguito, ho valutato i valori rimanenti: le colonne contenenti dati di tipo stringa sono state considerate come categoriali, mentre le rimanenti come numeriche. Dopo aver identificato le colonne, in base al tipo di queste, ho valutato i valori e calcolato le statistiche di interesse:

- * Per le colonne di tipo categoriale, il progetto richiedeva l'identificazione dei valori univoci ed il calcolo delle relative occorrenze;
- * Per le colonne di tipo numerico, il progetto richiedeva il valore minimo, massimo, medio e la deviazione standard dei valori presenti.

Per entrambi i tipi, inoltre, il progetto richiedeva il conteggio delle righe vuote per ogni colonna, così da identificare eventuali variabili che, se non significative, sarebbero state scartate durante l'analisi.

⁷<https://www.kaggle.com/c/bnp-paribas-cardif-claims-management/data>.

NAME	TYPE	EMPTY_COUNT	MIN	MAX	AVG	STDDEV	VALUE	VALUE_COUNT
ID	Integer	0	3.0	228713.0	114228.928	65934.487		
target	Integer	0	0.0	1.0	0.761	0.426		
v1	Double	49832	0.00	20.0	1.630	1.082		
v2	Double	49796	-9.817E-7	19.999	7.464	2.961		
v3	String	3457					C	110584
v3	String	3457					A	227
v3	String	3457					B	53
v4	Double	49796	-6.475E-7	19.999	4.145	1.148		
v5	Double	48624	-5.287E-7	20.0	8.742	2.036		
v6	Double	49832	-9.055E-7	20.0	2.436	0.599		
v7	Double	49832	-9.468E-7	19.999	2.483	0.589		
v8	Double	48619	-7.783E-7	20.0	1.496	2.783		
v9	Double	49851	-9.828E-7	20.0	9.031	1.930		
v10	Double	84	-9.875E-7	18.533	1.883	1.393		
v11	Double	49836	-1.459E-7	20.000	15.447	0.790		
v12	Double	86	5.143E-7	18.710	6.881	0.924		
v13	Double	49832	-8.464E-7	20.0	3.798	1.175		
v14	Double	4	-9.738E-7	19.999	12.094	1.443		
v15	Double	49836	-8.830E-7	19.999	2.080	0.732		
v16	Double	49895	-9.978E-7	20.0	4.923	1.791		
v17	Double	49796	-9.066E-7	19.999	3.832	1.911		
v18	Double	49832	4.475E-7	20.0	0.841	0.616		
v19	Double	49843	-5.178E-7	20.0	0.222	0.171		
v20	Double	49840	1.516	20.0	17.773	1.155		
v21	Double	611	0.106	19.296	7.029	1.072		
v22	String	500					OXC	21
v22	String	500					YUL	31
v22	String	500					QVO	8
v22	String	500					NBC	1
v22	String	500					XU	12
v22	String	500					KEB	3
v22	String	500					AEOE	3
v22	String	500					CRS	6
v22	String	500					ADEP	2
v22	String	500					HQN	97
v22	String	500					RKP	21
v22	String	500					VKY	6
v22	String	500					JJS	10

Figura 3.15: Esempio del CSV contenente le statistiche delle variabili ottenute

3.3.2 Calcolo della matrice di correlazione

La matrice di correlazione è un indice che esprime un'eventuale relazione di linearità tra le variabili su cui è calcolato. Si rivela dunque molto utile per la stima del modello statistico, in quanto si dimostra un'analisi preliminare valida per una prima stima delle variabili significative per il modello.

Nel caso particolare di questo progetto, ho costruito la matrice di correlazione utilizzando tutte e 133 le variabili. La colonna le cui correlazioni sono più interessanti è ovviamente la variabile "target", ma anche le correlazioni tra le altre colonne sono di interesse per capire se ci sono interazioni tra loro. Per questo motivo, tramite Spark, mi sono limitato a costruire la matrice di correlazione completa e poi, nella [web app](#) sviluppata, ho implementato la possibilità di visionare la matrice completa o solamente contenente le relazioni tra le variabili e "target".

Una volta concluse queste elaborazioni, ho mostrato il lavoro svolto al tutor che mi ha seguito per la parte *big data* e, dopo essermi confrontato su quanto ottenuto e su cosa visualizzare nella [web app](#) che avrei realizzato a breve, ha validato la conclusione della parte di lavoro di elaborazione del *dataset*.

	ID	target	v1	v2	v3Index	v4	v5
ID		1	-8.00E-05	-2.36E-04	-0.001517811	0.006867591	-0.001632177
target		-8.00E-05	1	-0.017314666	-0.001500525	0.026004862	5.78E-04
v1		-2.36E-04	-0.017314666	1	0.525969095	0.046940329	0.61102371
v2		-0.001517811	-0.001500525	0.525969095	1	0.06945032	0.897505482
v3Index		0.006867591	0.026004862	0.046940329	0.06945032	1	0.082911043
v4		-0.001632177	5.78E-04	0.61102371	0.897505482	0.082911043	1
v5		-6.17E-04	-0.013959644	0.613818277	0.801334489	0.087544123	0.85912468
v6		-9.19E-04	-0.009097705	0.656703042	0.806599474	0.07529378	0.914270725
v7		-0.001524408	-0.008725636	0.687468313	0.86721177	0.066463533	0.929536025
v8		-8.98E-04	-0.015185589	0.32579168	0.098798661	0.057209771	0.234308443
v9		-5.99E-04	-0.02043007	0.666416074	0.813358973	0.061008674	0.85155524
v10		-6.56E-04	0.148033564	0.009535017	0.044460105	0.271980971	0.050917979
v11		-0.002168558	-0.017620974	0.688368674	0.862830453	0.074232771	0.922499554
v12		-1.53E-04	0.050672975	0.0214508	0.052734552	0.233778165	0.059687476
v13		1.86E-04	-0.024761375	0.656372288	0.682123313	0.07621294	0.759314867
v14		7.63E-04	0.13034574	0.025546329	0.065588536	0.25278239	0.083521453
v15		9.76E-04	-0.027781948	0.640531516	0.675812358	0.060344922	0.728012434
v16		-0.002059204	-0.019632782	0.548491805	0.754940092	0.054232468	0.77758468
v17		-0.001004124	0.007079251	0.520768119	0.858526354	0.093630454	0.916787315
v18		-0.002715061	-0.010061702	0.523748313	0.489369457	0.059692822	0.608050387
v19		1.04E-05	-0.018213396	0.451748399	0.466837192	0.045228363	0.51797012
v20		-0.001601544	-0.016201714	0.68934012	0.873621925	0.07726824	0.934631186
v21		-7.31E-05	0.052379249	0.022478009	0.040687069	0.002799601	0.056832588
v22Index		-0.00213228	0.008488717	-0.077722116	0.113470324	-0.0162666	0.0408221
v23		-0.001592269	-0.027607728	0.251252061	-0.009459028	-0.003540402	-0.010526961
v24Index		-0.001849287	0.014172006	0.106442301	0.140084212	0.051367412	0.150132916
v25		-0.001064233	-0.013122581	0.345254437	0.132737517	0.066147559	0.279731047
v26		-2.67E-04	-0.007530306	0.708304144	0.829086392	0.079895206	0.923479362
v27		-0.002333961	-0.007545984	0.652482047	0.828184468	0.061975145	0.909090224
v28		-0.003570605	-0.020500833	0.571202894	0.591327056	0.066890356	0.679411246
v29		-6.23E-04	-0.01380555	0.651775444	0.869509458	0.08228518	0.92412695
v30Index		-0.003858192	-0.008225786	0.190472742	0.231410578	-0.047565978	0.252654702

Figura 3.16: Esempio del CSV contenente la matrice di correlazione tra variabili

3.4 Progettazione e sviluppo della web app

A seguito delle elaborazioni effettuate, mi ha affiancato un nuovo tutor, facente parte del reparto di sviluppo software. Come da [Piano di Lavoro](#), nella seconda parte del periodo di stage era programmato il ripasso di Java con attenzione all'ambiente Java EE. Per fare ciò, mi sono documentato soprattutto tramite la documentazione ed i tutorial ufficiali⁸ forniti da Oracle. Rispetto alla prima parte di stage, relativa ai *big data*, lo studio e lo svolgimento di esercizi riepilogativi che mi permettessero di mettere in pratica le conoscenze acquisite è stato più autonomo. Questo perchè, essendo alcune conoscenze di Java e sullo sviluppo di [web app](#) già in mio possesso, non ho incontrato particolari difficoltà, se non alcuni dubbi che il tutor ha risolto tempestivamente. Per quanto riguarda la scelta dello strumento per sviluppare la parte di *front-end*, inizialmente ho optato per le librerie React e Redux, in quanto già di mia conoscenza. Confrontandomi con il tutor, però, sono arrivato alla conclusione che l'utilizzo di Angular sarebbe stata la miglior scelta: in questo modo avrei potuto sfruttare il periodo in azienda per acquisire queste nuove conoscenze affiancate da personale qualificato ed esperto che avrebbe potuto aiutarmi in caso di incertezze.

I requisiti individuati per la [web app](#) seguono questa convenzione:

R[Tipo**][**Importanza**][**Codice**]**

Dove:

- * **Tipo:** specifica la tipologia del requisito e può assumere i valori:
 - F: requisito funzionale;
 - V: requisito di vincolo.
- * **Importanza:** specifica l'importanza del requisito e può assumere i valori:
 - O: requisito obbligatorio;
 - D: requisito desiderabile;
 - P: requisito opzionale;
- * **Codice:** è il codice gerarchico univoco di ogni requisito espresso in numeri.

⁸<https://docs.oracle.com/javaee/7/tutorial/index.html>.

Tabella 3.1: Requisiti per la web app

Codice	Descrizione
RFO01	L'utente deve poter visualizzare le statistiche delle variabili
RFO02	L'utente deve poter visualizzare la matrice di correlazione completa
RFO03	L'utente deve poter visualizzare la correlazione fra "target" e le altre variabili
RFO04	L'utente deve poter effettuare la ricerca di una specifica variabile
RFD01	L'utente deve poter visualizzare dei grafici statistici per le variabili numeriche
RFD02	L'utente deve poter visualizzare dei grafici statistici per le variabili categoriali
RFD03	L'utente deve poter visualizzare le tre maggiori correlazioni di una variabile
RFD04	L'utente deve poter impostare un limite inferiore della correlazione tra "target" e le altre variabili
RVO01	Utilizzo di Java EE
RVD01	Sviluppo di un'architettura three-tier

Inoltre, ho concordato assieme al tutor di utilizzare le risorse create a seguito dell'elaborazione del *dataset* salvate in locale e non sul *cluster*, in quanto l'amministratore di sistema, durante il mio ultimo periodo di stage, era impossibilitato a configurare l'accesso ad Hadoop tramite **Java JDBC**.

Data la natura didattica e la relativa semplicità della **web app**, dopo il consiglio del tutor aziendale, ho scelto di utilizzare GlassFish⁹ come *application server* in quanto ben supportato. Prima di cominciare con la progettazione del prodotto, ho inoltre considerato le operazioni necessarie sui dati da presentare già in possesso: era infatti richiesta la sola lettura di questi, quindi non mi sono preoccupato di impostare un sistema che fosse in grado di dare la possibilità all'utente di modificare i dati, essendo questa operazione non necessaria per la **web app**.

3.4.1 Architettura software

Come architettura del software, ho scelto una semplice architettura *three-tier*, che si compone di tre elementi principali:

- * **Presentation tier:** (primo livello) responsabile della presentazione e dell'interazione con l'utente;
- * **Business Logic tier:** (secondo livello) i processi in questo livello gestiscono la logica dell'applicazione. Si frappongono fra il primo ed il terzo livello, cosicché il client (primo livello) non comunicherà mai direttamente con i dati persistenti del livello inferiore; in questo modo, in questo strato, è possibile elaborare i dati e gestire la sicurezza prima dell'accesso alla risorsa persistente;

⁹<https://javaee.github.io/glassfish/>.

- * **Data tier:** (terzo livello) contiene i dati salvati in un database o in un *filesystem*; questi dati possono essere acceduti solamente dal livello superiore.

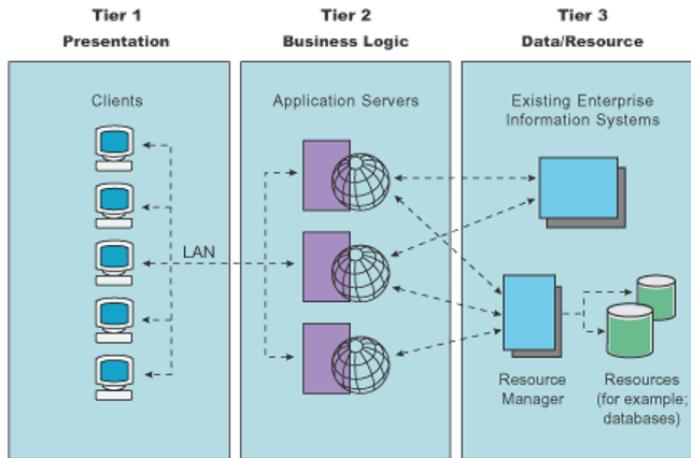


Figura 3.17: Esempio di architettura three-tier generica (Fonte: <https://goo.gl/nuFksu>)

Nella figura sottostante è possibile vedere come sia applicata un'architettura *three-tier* in ambiente Java EE.

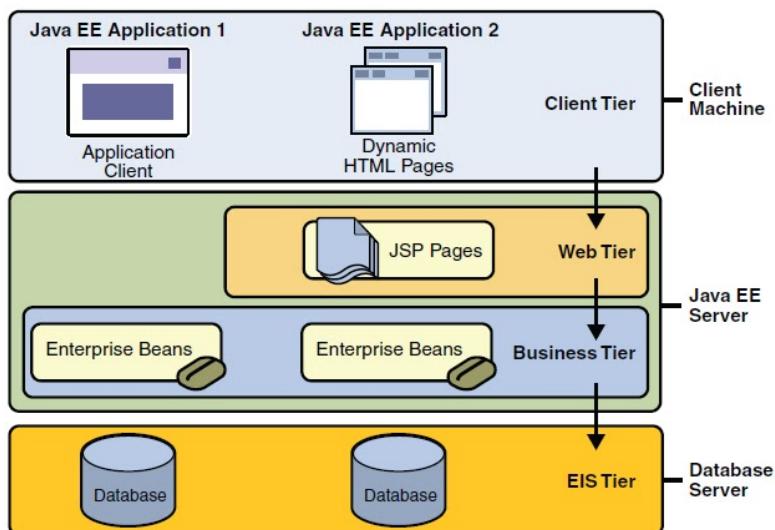


Figura 3.18: Esempio di architettura three-tier in Java EE (Fonte: <https://goo.gl/S6RYgT>)

Per lo sviluppo del prodotto, ho deciso di utilizzare un insieme di tecnologie offerte da Java EE.

JavaServer Pages

Le JavaServer Pages (JSP) permettono di sviluppare pagine web generate dinamicamente, innestando codice Java all'interno di semplici pagine HTML. Per eseguire il codice Java e generare i contenuti dinamici, questo viene eseguito a *runtime* e trasformato in servlet dal server in cui la [web app](#) è in esecuzione e poi il risultato elaborato viene restituito al client. Nel mio caso ho sviluppato tre JSP:

- * corrMatrix.jsp: contenente la pagina di visualizzazione della matrice di correlazione;
- * corrTarget.jsp: contenente la pagina di visualizzazione delle correlazioni fra "target" e le altre variabili;
- * listData.jsp: contenente le statiche delle variabili.

Java Servlet

I servlet sono oggetti che operano all'interno di un server web (es. Tomcat) oppure un server per applicazioni (come nel mio caso, GlassFish) permettendo la creazione di applicazione web e l'elaborazione di richieste da parte di un client. I servlet sono associati ad uno specifico *URL* a seconda dei parametri di richiesta inviati dal client browser dell'utente al server e, una volta invocati, sono responsabili della pagina da visualizzare o quale parte dell'applicazione invocare.

Il flusso di dati di un servlet avviene nel seguente modo:

1. Un client invia una richiesta (*request*) per un servlet ad un server;
2. Qualora si tratti della prima richiesta, il server istanzia e carica il servlet in questione avviando un *thread* che gestisca la comunicazione con il servlet stesso. Nel caso, invece, in cui il servlet sia già stato caricato in precedenza allora verrà, più semplicemente, creato un ulteriore *thread* che sarà associato al nuovo client, senza la necessità di ricaricare ancora il servlet;
3. Il server invia al servlet la richiesta pervenutagli dal client;
4. Il servlet costruisce ed imposta la risposta (*response*) e la inoltra al server;
5. Il server invia la risposta al client.

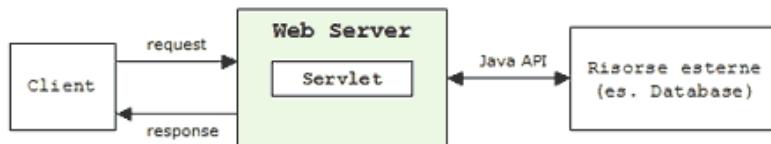


Figura 3.19: Flusso di dati in un servlet

Nel mio caso ho sviluppato i seguenti servlet:

- * DataServlet.java: contenente il servlet per l'inizializzazione dei contenuti di listData.jsp e dei relativi dati;
- * MatrixServlet.java: contenente il servlet per l'inizializzazione dei contenuti di corrMatrix.jsp e dei relativi dati;

- * TargetServlet.java: contenente il servlet per l'inizializzazione dei contenuti di corrTarget.jsp e dei relativi dati;
- * UtilCorrServlet.java: risponde inviando oggetti di tipo **JSON** alle richieste di dati da parte degli script Angular per la visualizzazione dei grafici relativi alla correlazione fra "target" e le altre variabili;
- * UtilStatsServlet.java: risponde inviando oggetti di tipo **JSON** alle richieste di dati da parte degli script Angular per la paginazione dei risultati e la visualizzazione dei grafici relativi alle statistiche delle variabili.

Enterprise JavaBeans

Gli Enterprise JavaBeans (EJB) sono i componenti software che implementano, lato server, la logica di business di una [web app](#) all'interno della piattaforma Java EE, per definire il contenuto dinamico da mostrare al client, che può essere una pagina JSP, un servlet o anche un altro EJB o una classe generica Java. Risiedono dunque su un *application server* e sono tipicamente utilizzati in architetture software di tipo *multi-tier*. Nel mio caso ho sviluppato tre beans:

- * Data.java: contenente il modello dei dati delle variabili e delle relative statistiche;
- * Matrix.java: contenente il modello dei dati della matrice di correlazione;
- * Target.java: contenente il modello dei dati delle correlazioni fra "target" e le altre variabili;

Oltre a queste classi, contenenti i modelli base degli oggetti, ho sviluppato altre tre classi per gestire alcuni metodi statici di servizio per semplificare il modello dei beans, ad esempio per la lettura dei dati dai file [CSV](#).

La composizione di questi tre componenti offerti da Java EE permette di ottenere un'architettura *three-tier* che implementa il pattern architettonicale MVC (Model-View-Controller).

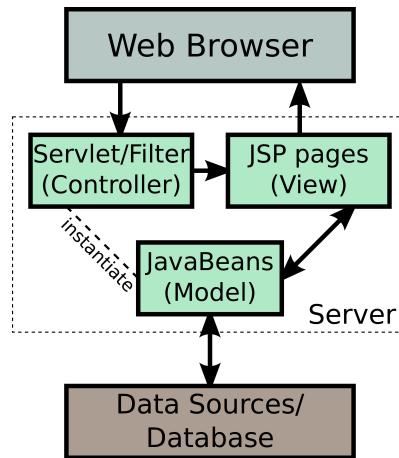


Figura 3.20: Esempio di interazione fra JSP, Servlet ed Enterprise JavaBeans (Fonte: <https://goo.gl/bWRWty>)

3.5 Risultati ottenuti

Durante tutta la durata dello stage ho prodotto i seguenti documenti:

1. Ripasso dei comandi della **Bash**;
2. Come funziona Hadoop e componenti principali;
3. Funzionamento e comandi di Hive ed Impala;
4. Funzionamento e comandi di Spark ed utilizzo in applicazioni Scala;
5. Comandi base di Cloudera Manager;
6. Architettura del software;
7. Presentazione del progetto finale.

Questa documentazione mi ha permesso di concretizzare i concetti studiati durante la prima fase e, nella successiva, di poter usufruire di una fonte di informazioni in caso di dubbi.

Per l'implementazione del prodotto ho scritto 1659 righe di codice.

Non ho implementato test di verifica del prodotto, come già concordato inizialmente con il tutor aziendale, in quanto il tempo a disposizione per il progetto finale era esiguo e l'interazione con l'utente era limitata alla visualizzazione di risultati e semplici richieste di elaborazione di dati.

Il bilancio del soddisfacimento dei requisiti è di seguito riportato, in cui, nella colonna "Stato", sono presenti dei codici così riepilogati:

- * **SO**: requisito soddisfatto;
- * **PS**: requisito parzialmente soddisfatto;
- * **NS**: requisito non soddisfatto.

Tabella 3.2: Riepilogo soddisfacimento requisiti per la web app

Codice	Descrizione	Stato
RFO01	L'utente deve poter visualizzare le statistiche delle variabili	SO
RFO02	L'utente deve poter visualizzare la matrice di correlazione completa	SO
RFO03	L'utente deve poter visualizzare la correlazione fra "target" e le altre variabili	SO
RFO04	L'utente deve poter effettuare la ricerca di una specifica variabile	SO
RFD01	L'utente deve poter visualizzare dei grafici statistici per le variabili numeriche	SO
RFD02	L'utente deve poter visualizzare dei grafici statistici per le variabili categoriali	SO
RFD03	L'utente deve poter visualizzare le tre maggiori correlazioni di una variabile	SO
RFD04	L'utente deve poter impostare un limite inferiore della correlazione tra "target" e le altre variabili	SO
RVO01	Utilizzo di Java EE	SO
RVD01	Sviluppo di un'architettura three-tier	SO

Tabella 3.3: Bilancio soddisfacimento requisiti per la web app

Importanza	Individuati	Soddisfatti	%
Obbligatori	5	5	100
Desiderabili	5	5	100
Opzionali	0	0	100

Glossario

API insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per l'espletamento di un determinato compito all'interno di un certo programma. La finalità è ottenere un'astrazione, di solito tra l'hardware e il programmatore o tra software a basso e quello ad alto livello semplificando così il lavoro di programmazione. [24](#)

Bash shell testuale del progetto GNU usata nei sistemi operativi Unix e Unix-like, specialmente in GNU/Linux. Si tratta di un interprete di comandi che permette all'utente di comunicare col sistema operativo attraverso una serie di funzioni predefinite, o di eseguire programmi e script.

Bash è in grado di eseguire i comandi che le vengono passati, utilizzando la redirezione dell'input e dell'output per eseguire più programmi in cascata in una pipeline software, passando l'output del comando precedente come input del comando successivo. Oltre a questo, essa mette a disposizione un semplice linguaggio di scripting nativo che permette di svolgere compiti più complessi, non solo raccogliendo in uno script una serie di comandi, ma anche utilizzando variabili, funzioni e strutture di controllo di flusso. [7](#), [18](#), [25](#), [35](#)

Cluster (indicato anche come computer cluster) insieme di macchine connesse tra loro che lavorano in parallelo. L'utilizzo di questi sistemi permette di distribuire un'elaborazione molto complessa tra le varie macchine, aumentando la potenza di calcolo del sistema e/o garantendo una maggiore disponibilità di servizio, a prezzo di un maggior costo e complessità di gestione dell'infrastruttura: per essere risolto, il problema che richiede molte elaborazioni, viene infatti scomposto in sottoproblemi separati i quali vengono risolti ciascuno in parallelo su tutti i nodi che compongono il cluster. [iii](#), [3](#), [11](#), [12](#), [16](#), [18](#), [22](#), [24](#), [25](#), [27](#), [31](#)

Comma-separated values formato di file basato su file di testo utilizzato per l'importazione ed esportazione di una tabella di dati. ogni riga della tabella (o record della base dati) è normalmente rappresentata da una linea di testo, che a sua volta è divisa in campi (le singole colonne) separati da un apposito carattere separatore, ciascuno dei quali rappresenta un valore. [14](#), [34](#)

Daemon programma eseguito in background, cioè senza che sia sotto il controllo diretto dell'utente, tipicamente fornendo un servizio all'utente. Spesso vengono avviati al boot del sistema per rispondere a richieste di rete, attività hardware o altri programmi eseguendo alcuni task. [20](#), [22](#), [24](#)

Diagramma di Gantt strumento di supporto alla gestione dei progetti, così chiamato in ricordo dell'ingegnere statunitense Henry Laurence Gantt (1861-1919), che

si occupava di scienze sociali e che lo ideò nel 1917. Tale diagramma è usato principalmente nelle attività di *project management*, ed è costruito partendo da un asse orizzontale - a rappresentazione dell’arco temporale totale del progetto, suddiviso in fasi incrementali (ad esempio: giorni, settimane, mesi) - e da un asse verticale - a rappresentazione delle mansioni o attività che costituiscono il progetto. [15](#)

Git Git è un software di controllo versione distribuito utilizzabile da interfaccia a riga di comando, creato da Linus Torvalds nel 2005 con lo scopo di essere un semplice strumento per facilitare lo sviluppo del kernel Linux, e diventato poi uno degli strumenti di controllo versione più diffusi al mondo. [5](#), [6](#), [16](#)

GitLab GitLab è un manager di *repository* Git basato su interfaccia web, che include anche funzioni quali una wiki per ogni progetto e un sistema di tracciamento issue. Esso è stato sviluppato da GitLab Inc. ed è distribuito gratuitamente con licenza *open source*. [6](#)

HDFS Hadoop Distributed File System è un file system distribuito, portatile e scalabile scritto in Java per il framework Hadoop. Un cluster in Hadoop tipicamente possiede un singolo NameNode (su cui risiedono i metadati dei file) e un insieme di DataNode (su cui risiedono, in blocchi di dimensione fissa, i file dell’HDFS). [3](#), [7](#), [12](#), [15](#), [18](#), [20](#)

IDE (in lingua inglese *Integrated Development Environment* ovvero IDE, anche *integrated design environment* o *integrated debugging environment*, rispettivamente ambiente integrato di progettazione e ambiente integrato di *debugging*) è un software che, in fase di programmazione, aiuta i programmatore nello sviluppo del codice sorgente di un programma. Spesso l’IDE aiuta lo sviluppatore segnalando errori di sintassi del codice direttamente in fase di scrittura, oltre a tutta una serie di strumenti e funzionalità di supporto alla fase di sviluppo e *debugging*. [7](#)

Java Archive formato di file utilizzato per aggregare più classi Java e associare i metadati e le risorse (testo, immagini, etc.) in un unico file per agevolare la distribuzione. [27](#)

Java JDBC connettore (driver) per database che consente l’accesso e la gestione della persistenza dei dati sulle basi di dati da qualsiasi programma scritto in linguaggio Java, indipendentemente dal tipo di DBMS utilizzato. L’architettura di JDBC prevede l’utilizzo di un *driver manager*, che espone alle applicazioni un insieme di interfacce standard e si occupa di caricare a *runtime* i driver opportuni per gestire gli specifici DBMS. Le applicazioni Java utilizzano le JDBC [API](#) per parlare con il JDBC *driver manager*, mentre il *driver manager* usa le JDBC [driver API](#) per parlare con i singoli driver che pilotano i DBMS specifici. [24](#), [31](#)

JavaScript Object Notation nell’ambito della programmazione web, JSON, acronimo di JavaScript Object Notation, è un formato adatto all’interscambio di dati fra applicazioni *client-server*. [34](#)

Minimum Viable Product prototipo più semplificato possibile che è possibile presentare ad una cerchia di possibili clienti (early adopter). È il mezzo con cui

testare e validare le idee e il prodotto stesso, senza sprecare tempo e soldi a sviluppare il prodotto completo, per poi constatare che quel prodotto non interessa alla clientela. [5](#)

Software Development Lifecycle processo di divisione del lavoro di sviluppo software in fasi distinte per migliorare la progettazione, la gestione del prodotto e la gestione del progetto. [1](#)

Single Customer View rappresentazione olistica del cliente che integra tutti i dati e gli eventi del cliente, e consente di arrivare ad un'interpretazione completa e contestuale dei suoi comportamenti indipendentemente dai canali utilizzati. [2](#)

Single Point Of Failure (SPOF) parte del sistema, hardware o software, il cui malfunzionamento può portare ad anomalie o addirittura alla cessazione del servizio da parte del sistema. Solitamente si cerca di evitare ogni SPOF nel momento della progettazione, soprattutto in sistemi pensati specificatamente per essere attivi costantemente, avvalendosi di componenti ridondanti che ne garantiscono il funzionamento anche in caso di guasto. [20](#)

Web App sistema di tipo client-server in cui l'interfaccia utente e la logica client-side viene eseguita in un browser web. [iii](#), [11](#), [12](#), [14](#), [16](#), [28](#), [30](#), [31](#), [33](#), [34](#)

Web Service sistema software in grado di mettersi al servizio di un'applicazione comunicando su di una medesima rete tramite il protocollo HTTP. Un Web service consente quindi alle applicazioni che vi si collegano di usufruire delle funzioni che mette a disposizione. [11](#)