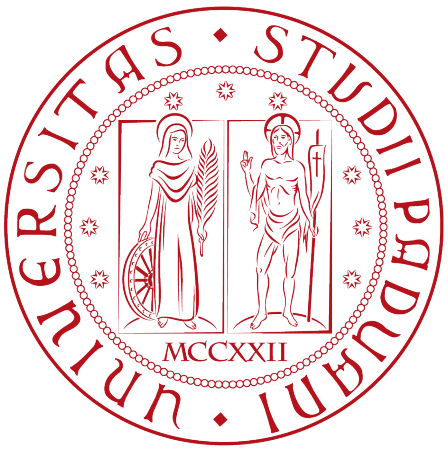
Corso di Laurea in Informatica

Università degli Studi di Padova



Sviluppo di API REST in ambito IoT e di una webapp per il controllo qualità

Tesi di laurea triennale

Anno Accademico 2021-2022

Laureando

Elvis Murtezan

Relatore

Dott. Michele Scquizzato

Sommario

Il presente documento descrive la formazione ricevuta in ambito lavorativo, svolta nel periodo di stage della durata complessiva di 320 ore, presso l'azienda Méthode S.r.l. di San Vendemiano. Lo scopo primario dello stage è stato lo studio di alcune tecnologie al fine di sviluppare delle componenti per interfacce utente (*UI*) di una webapp SPA[G] e lo sviluppo di alcune API[G] REST[G] in un contesto che usa tecnologie IoT[G], in modo che lo stage potesse dare una visione più chiara ed amplia dello sviluppo nelle sue diverse fasi e competenze, ovvero in ottica fullstack[G].
Tale documento nello specifico è suddiviso in 4 parti principali:

- Il **primo capitolo** presenta la realtà aziendale, il suo dominio applicativo e le tecnologie utilizzate in tale contesto;
- Il **secondo capitolo** descrive invece lo stage in termini di obiettivi, metodi e vincoli da rispettare;
- Il **terzo capitolo** documenta lo svolgimento dello stage presentando le attività svolte suddivise per tipologia di sviluppo;
- Il **quarto** ed **ultimo capitolo** contiene una valutazione retrospettiva personale dello svolgimento dello stage rispetto agli obiettivi iniziali e alle conoscenze acquisite.

Inoltre, al fine di chiarire eventuali dubbi, nel presente documento in appendice è consultabile un glossario che fornisce una breve descrizione e/o concisa definizione di qualunque termine o concetto di comprovata natura tecnica o che fornisca una chiave di lettura molto vicina a quella voluta dall'autore. Alcuni dati ed informazioni presenti nel documento, più specificatamente tutti quelli riportati negli *screenshot* proposti di componenti (frontend[G] o backend[G]) realizzate come progetto didattico risulteranno offuscati o non veritieri. Questa scelta è necessaria per motivi di *privacy* e riservatezza nei confronti delle realtà coinvolte.
Segue una guida all'uso delle tipologie di *link*:

- i *link* in blu: indicano un *link* esterno (*url*) implicito o esplicito;
- i *link* in rosso (ex. Capitolo 1): indicano un *link* interno (ex. sezioni, sottosezioni, ecc.)
- mente i *link* in rosso seguiti da una G ad apice (ex. IoT[G]): indicano una voce del glossario.

Ringraziamenti

Desidero innanzitutto ringraziare il Dott. Michele Scquizzato, relatore della mia tesi, per l'aiuto attento e preciso che è riuscito a darmi durante la sua stesura (nonché per la pazienza dimostrata), sono molto contento di aver lavorato con Lei.

Ringrazio tutte le persone con cui sono venuto a contatto durante lo stage presso Méthode S.r.l. , in modo particolare il mio tutor aziendale Davide Zanatta, che mi ha seguito per tutta la durata dello stage. Grazie per avermi coinvolto molto, sia nel team working che nei vari progetti a cui ho partecipato, permettendomi in questo modo di avere una prospettiva (seppur non dettagliata) dell'intero processo di sviluppo professionale.

Un grazie di cuore va a tutta la mia Famiglia.

Ringrazio la community dei futuri informatici dell'università di Padova (FIUP) per incentivare la collaborazione tra noi studenti.
Ringrazio per le risate, il sostegno e l'aver sopportato le mie strane trasformazioni in persona noiosa e antipatica prima degli esami: Andrea, Christian, Enrico, Gerardo, Kole, Lucas, Samuel, Singh, Vincenzo e tutti gli altri, sono grato di aver conosciuto persone speciali come voi.

Infine, ringrazio me stesso per non aver mollato ed essere arrivato fin qui.

"Più sai, più ti rendi conto di non sapere"
-Socrate.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

L'azienda

1.1 Profilo aziendale

Méthode Srl nasce nel 2004 quando alcuni giovani professionisti decidono di puntare sulla specializzazione nella Business Intelligence^G. I valori fondanti sono la specifica focalizzazione, l'attenzione ai processi da portare alle aziende e la costante ricerca di innovazione nel campo della Business Intelligence e Advanced Analytics^G. Inizialmente *partner* di Business Objects, è dal 2008 partner SAP: ad oggi Méthode è SAP Gold Partner (SAP Recognized Expertise per la Business Intelligence, SAP Hana e Enterprise Information Management), inoltre ha anche *partnership* con Microsoft, Qlink e Tableau. Opera principalmente nel Nord e nel Centro Italia tramite una propria sede legale ed operativa in provincia di Treviso ed una sede operativa di Milano. In forte e recente espansione, in tale ottica nel 2018 è stato creato un team che si dedica soltanto allo sviluppo *software*. Méthode si presenta come una struttura snella e compatta di consulenti specializzati capaci di realizzare complessi progetti di Business Intelligence^G. L'ambiente di lavoro è al tempo stesso giovane, stimolante e dinamico; i diversi *team* ed i relativi singoli componenti lavorano a stretto contatto permettendo lo sviluppo di progetti di primo piano con aziende *leader* di settore. Tale ecosistema garantisce il confronto e l'interazione tra colleghi dalle diverse competenze ed aiuta i tirocinanti durante il loro ingresso nella realtà aziendale. A prova di tutto ciò, l'azienda è certificata Great Place to Work^G da diversi anni.

1.2 Organizzazione interna

L'azienda, ai suoi esordi, non aveva una struttura (dei ruoli) ben definita, ma nel corso degli anni è stata protagonista di una forte crescita incrementando il proprio personale fino ad arrivare ad un collettivo di oltre 70 persone suddivise nelle due sedi aziendali. Tale crescita ha portato ad una riorganizzazione interna dove il personale è suddiviso in *team* ripartiti per ambito di competenza specifica:

- **team leader**: figura di riferimento del proprio *team* che collabora con i *project manager* nel gestire i requisiti in entrata dei vari progetti ed in particolare gestisce la schedulazione di ciascun membro del proprio *team* di competenza;
- **project manager**: figura che analizza e sovrintende i vari progetti e i *team* coinvolti (comunicando con il *team leader*) dall'inizio alla fine di un progetto (a mio avviso figura molto più vicina al *Product Owner* in questo caso, dato che la gestione giornaliera dei team è affidata ai *team leader* di competenza);

- **consulente**: personale con competenze specifiche ad un dominio e riferisce al relativo *team* (ex. programmatore *software* che riferisce al team di sviluppo, il sistemista che riferisce al *team* di infrastruttura...).
- Inoltre, vi sono i reparti predisposti per la gestione:
  - delle risorse umane;
  - dell'amministrazione;
  - del *marketing*.

L'azienda cerca di migliorarsi in modo continuo, cercando di perseguire principalmente i seguenti obiettivi:

- essere pronti ad un'eventuale ulteriore crescita del personale;
- migliorare l'interazione tra i clienti ed i consulenti aziendali;
- accrescere e perfezionare la gestione clienti per soddisfare in modo più efficace ed efficiente le diverse necessità;
- perseguire l'incremento della qualità dei processi aziendali e dei prodotti finali.

1.3 Dominio applicativo e soluzioni aziendali

Méthode è un'azienda specializzata nel settore della Business Intelligence^G. La Business Intelligence^G rappresenta un ramo della cosiddetta informatica aziendale, improntata su strategie e tecniche utili alle aziende per colmare la distanza tra i dati, le informazioni e le persone che devono prendere decisioni: è un percorso che va dall'estrazione del dato, al suo trattamento, fino alla sua presentazione nella forma migliore per fini di gestione aziendale di varia tipologia.

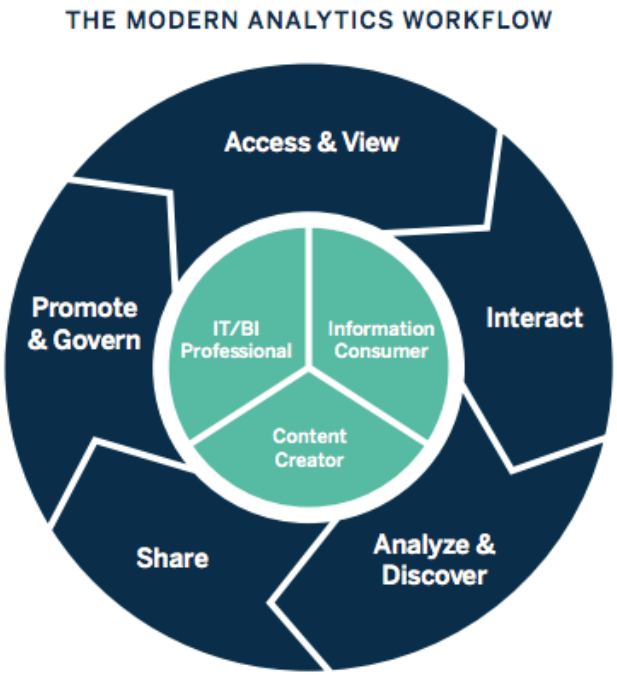


Figura 1: Il ciclo di analisi nella BI moderna che evidenzia la natura interattiva
fonte: tableau.com

Méthode propone diverse soluzioni per soddisfare queste esigenze aziendali (che hanno specifico rilievo nel mondo del *business*), le principali soluzioni offerte ricadono nei seguenti campi:

- **Business Analytics**

  Per decisioni di *business* basate su informazioni più precise, dettagliate ed aggiornate; offre funzioni di report^G, data discovery^G e dashboard^G;

- **Predictive & Advanced Analytics**

  Basata sull'analisi predittiva, supera la natura descrittiva della Business Intelligence^G classica associando ad un evento la probabilità che accada.
  Supporta tutti gli ambiti aziendali, dalle campagne pubblicitarie mirate, alla fidelizzazione del cliente, alla pianificazione di cicli manutentivi.

- **Process Mining**

  Focalizzato sui propri processi aziendali per individuarne i punti deboli e le criticità,

permette di comprendere se il proprio modello di business è efficiente, nel caso di anomalie ne favorisce la loro identificazione e correzione;

- **BigData**

  Fondamento delle successive analisi, dove vengono impiegati più dispositivi e fonti informative (come sensori, *social network*, e strumenti connessi alla rete) dove è di vitale importanza istanziare processi e tecnologie per la gestione e l'immagazzinamento di queste grandi moli di dati; si ricorre pertanto a basi di dati efficienti, efficaci e flessibili che consentano in seguito data integration^G, data quality^G o elaborazioni real-time^G;

- **Cloud Solutions**

  Il cloud^G costituisce un nuovo approccio all'infrastruttura tecnologica, potenzialmente può ridurre i costi (TCO^G) ed offrire possibilità più idonee all'Internet of Things (IoT^G) e maggiormente scalabili;

- **Corporate Performance Management**

  Per una migliore *governance* aziendale basata su strumenti per il *budgeting*, la pianificazione ed il controllo;

- **Smart Financial Planning**

  È una soluzione flessibile, veloce e di facile utilizzo per la gestione del *cash flow*. Viene offerta una progressiva visione efficacie della situazione e degli impatti economici, per garantire la liquidità dell'azienda. Consente di analizzare e controllare l'intero processo di pianificazione economico finanziaria in modo rapido e intuitivo.

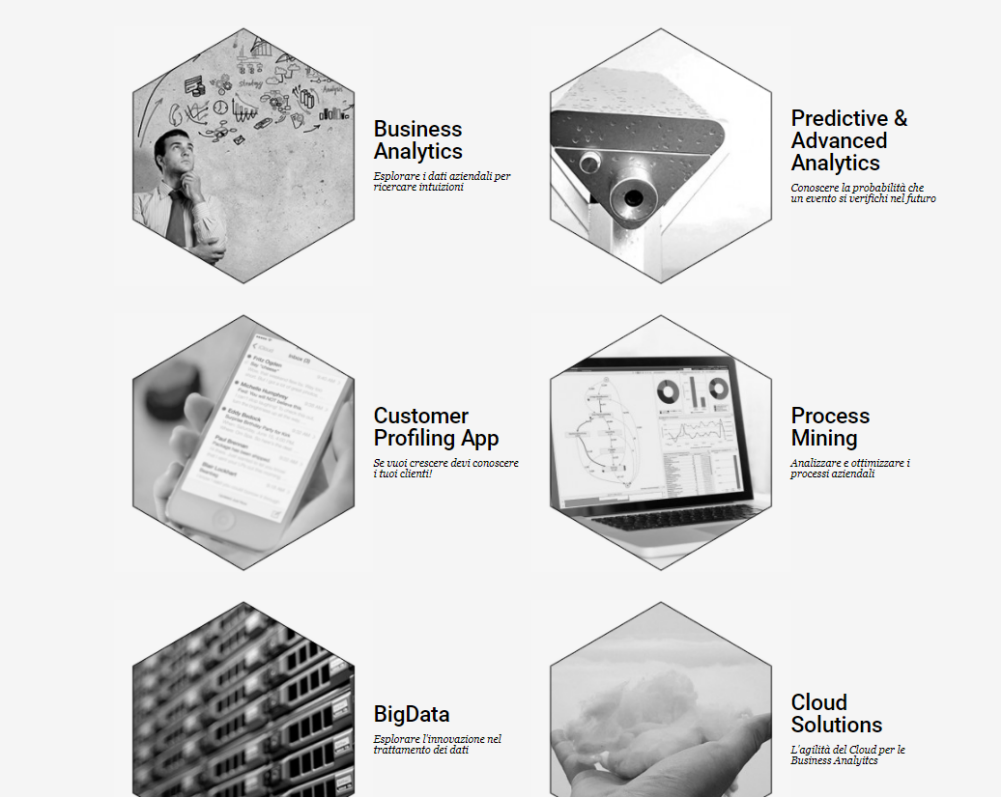


Figura 2: Infografica delle tipologie di soluzione offerte da Méthode
fonte: methode.it

1.4 Tecnologie Usate

Data la complessità del dominio d'interesse (sezione sezione 1.3), Méthode basa la quasi totalità della sua offerta su prodotti, strumenti e pacchetti *software* forniti da SAP, Microsoft, Qlik e Tableau (*partner* dell'azienda), queste aziende software propongono molteplici prodotti per ogni esigenza e coprono l'intero *stack* tecnologico necessario a tali fini, ovvero partendo dall'immagazzinamento dei dati fino ad arrivare alla visualizzazione di report^G finali.
I sudetti *software* operano prevalentemente per via grafica, questo approccio permette di focalizzarsi sulla logica del problema e demanda agli strumenti le ottimizzazioni di basso livello. Sono inoltre d'aiuto nella gestione e lo sviluppo di progetti in contesti estesi e complessi. Mentre per *software* con personalizzazione dedicata se ne incarica il *team* di sviluppo scrivendo codice tramite diverse tecnologie che generalmente almeno in parte includono lo *stack* offerto dai *partner* aziendali. In particolare, la maggior parte delle attività aziendali gravita attorno al principale partner, ovvero SAP ed i suoi svariati strumenti. Segue una panoramica dei principali strumenti utilizzati in azienda suddivisa per area di sviluppo.

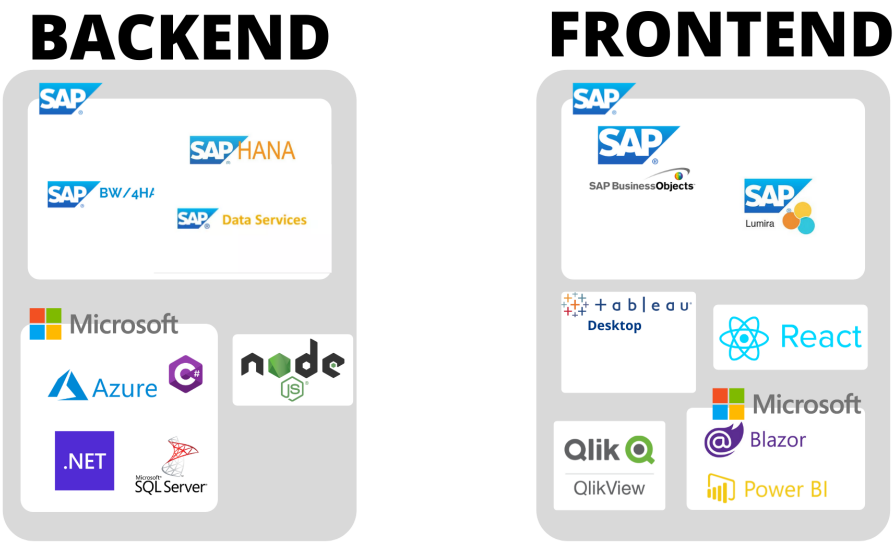


Figura 3: Infografica di parte dello *stack* tecnologico aziendale principale, suddiviso per area di sviluppo

1.4.1 Backend

- **SAP Data Services**

  Strumento per l'*Extract*, *Transform*, *Load* (ETL^G), utilizzato principalmente per le sue funzionalità di data integration^G, data quality^G e *data cleansing*.

- **SAP Hana (Platform)**

  Composto di vari moduli, usato per il suo *database engine* colonnare *in-memory* e per la modellazione dei dati.

- **SAP BW/4Hana**

  Per la creazione e la gestione di *Enterprise Data Warehouse*, progettato per operare al di sopra di Hana Platform e sfruttarne la tecnologia *in-memory* per migliori performance.

- **SQL Server Integration Services (SSIS)**

  Piattaforma software di Microsoft, offre funzionalità di ETL^G e di data integration^G.

1.4.2 Frontend

- **SAP BusinessObjects Business Intelligence suite**

  Piattaforma composta di strumenti per la creazione di report^G e analisi, include:

  – Information Design Tool (IDT): per la gestione del livello semantico ("universo"

nella terminologia SAP) che mappa i dati sottostanti in oggetti più familiari al mondo business (come prodotti, clienti e fatture);

– Web Intelligence (WebI): per la realizzazione dei report^G.

- **SAP Lumira**

  Per la realizzazione di dashboard^G e lo sviluppo di applicazioni *client*.

- **QlikView**

  Strumento di Qlink per la creazione e la visualizzazione di dashboard^G.

- **PowerBI**

  Controparte Microsoft per la realizzazione di dashboard^G e la creazione di report^G.

- **Tableau**

  Soluzione proprietaria per la creazione di dashboard^G interattive.

1.4.3 Tecnologie varie

La maggior parte di queste tecnologie viene utilizzata prevalentemente dal *team* di sviluppo (di cui ne ho fatto parte) per personalizzazioni software dedicate, per interagire con le tecnologie proposte dai *partner* in modo opportuno o dalle necessità del contesto(ex. analisi di dati). Segue una breve descrizione di quelli visti usare dai colleghi di altri *team* o che ho utilizzato in prima persona.
N.B. per le tecnologie da me utilizzate si rimanda alla sezione 3.1.

- **Structured Query Language (SQL)**

  In varie sue declinazioni, per l'interazione con database quali Oracle e SQL Server di Microsoft, in particolare nelle attività di backend^G che richiedono tale iterazione.

- **Python, R**

  Python come linguaggio di scripting ed R come linguaggio specifico per l'analisi statica (utilizzate per lo più dal team dedicato all'analisi dei dati).

- **Java, JavaScript, TypeScript, C#**

  Linguaggi su cui si basano i principali strumenti utilizzati dal team di sviluppo.

- **React, Redux**

  Vengono utilizzati solo dal team di sviluppo, sono delle librerie per il frontend^G open-source JavaScript per facilitare lo sviluppo di interfacce grafiche (anche di una certa complessità ex. dashboard^G) e la relativa interazione con esse.Sono state scelte come specializzazione frontend^G da parte del *team* di sviluppo nel 2018 per offrire personalizzazione grafica dedicata.

- **Node.Js**

  È un runtime system^G *open source* multi-piattaforma orientato agli eventi per l'esecuzione di codice JavaScript, costruita sul motore JavaScript V8 di Google Chrome. Questa tecnologia viene usata soltanto dal *team* di sviluppo.

- **Blazor**

  Libreria *open-source* per interfacce grafiche usata da applicazioni web col linguaggio C# (ex. sviluppo della parte frontend^G di sistemi ETL^G sviluppati con la piattaforma .NET).

L'azienda, comunque, offre anche consulenza utilizzando altre tecnologie, a secondo delle necessità del cliente.

1.5 Strumenti di supporto

Il personale di Méthode ha a sua disposizione degli strumenti di supporto, generalmente tali strumenti sono parte integrante dell'ambiente e della configurazione aziendale; essi facilitano la gestione del lavoro, la comunicazione interna ed esterna con i clienti. Segue una lista di questi ultimi.

1.5.1 Web Office (WO)

Sviluppato all'interno dell'azienda, è il principale strumento di supporto ed offre una serie di servizi per la gestione delle risorse non informatiche, per la gestione delle attività del personale e della regolamentazione aziendale. I servizi offerti sono:

- **Bacheca**

  Utilizzata per la pubblicazione di documenti e procedure generali destinate a tutto il personale, come: regolamenti, piani di evacuazione e informative sulla privacy.

- **Gestione dei rapportini**

  Sezione in cui ogni lavoratore deve rendicontare le attività giornaliere svolte.

- **Scheduler sale e auto**

  Riporta le prenotazioni e le assegnazioni delle sale riunioni e delle auto aziendali.

- **Scheduler utente**

  Sezione dedicata alla gestione delle attività, compilata dai *team leader* e dai project manager, elenca tutto il personale e le relative attività assegnate a ciascuno, mostra inoltre se la persona è in sede o presso un cliente; questo permette di sapere con anticipo su cosa si dovrà lavorare, e consente la gestione e l'assegnazione delle risorse.

- **Gestione delle trasferte e delle note spesa**

- **Gestione di permessi, ferie e dello *smart working***

1.5.2 FreshService

Portale web, è il sistema di ticketing^G dell'azienda per il supporto interno, permette di:

- prenotare una risorse condivisa (smartphone, tablet, ecc);
- richiedere l'installazione o l'abilitazione a *software* particolari;
- richiedere le credenziali e l'accesso alle macchine e ai servizi di sede;
- gestire le sostituzioni della dotazione *hardware* (come schermi, tastiere);
- segnalare anomalie alle infrastrutture o alle dotazioni.

A seconda delle necessità occorre aprire un *ticket* che verrà preso in carica dal team di supporto interno che provvederà, in base a urgenza e disponibilità, alla sua risoluzione. Dopo la chiusura viene chiesto un *feedback*: sia funzionale (la richiesta è stata evasa interamente e con successo) sia qualitativo (una valutazione di gradimento dell'intervento).

1.5.3 Google suite

Insieme di servizi *software* di Google, dove quelli principalmente usati sono:

- **GMail**

  Il servizio di posta elettronica tramite il quale avvengono le comunicazioni di una certa rilevanza (ex. con i clienti).

- **Chat**

  La *chat* integrata funge da strumento di comunicazione rapida interna, utilizzata in modo molto frequente da tutto il personale.

- **GDrive**

  Spazio di archiviazione cloud^G, dove ogni team ha un suo spazio in cui sono condivise documentazioni, file d'interesse generale e materiale per l'auto-formazione.

- **Meet**

  Strumento utilizzato per riunioni e discussioni da remoto o per una veloce consultazione tra colleghi.

1.6 Clienti

Méthode si rivolge a piccole, medie e grandi imprese intenzionate a ampliare la visione che il loro management ha del proprio business. Più in generale si rivolge a quelle aziende che desiderano conoscere il loro stato e le loro performance in modo innovativo, più efficiente e più rapido, proponendo inoltre attività di analisi predittiva a partire dallo storico aggiornato dei dati aziendali. Tali aziende appartengono ai settori più disparati, tra i quali: alimentare, farmaceutico, manifatturiero, metalmeccanico, siderurgico e della moda. Solitamente i servizi sono sviluppati a partire da esigenze del cliente: tali esigenze possono riguardare porzioni ristrette del loro business (come la gestione di dati telemetrici di specifiche macchine) o aree più ampie che riguardano la quasi totalità dei dati in loro possesso (come migrazioni o integrazioni di basi di dati con servizi preesistenti). L'azienda si occupa anche della manutenzione e dell'evoluzione dei servizi sviluppati qualora i clienti riscontrino problematiche, cambino alcune logiche o richiedano nuove funzionalità.

1.7 Innovazione

Lo sforzo innovativo di Méthode si concentra principalmente nelle seguenti aree:

- **Innovazione organizzativa interna**

  La rapida crescita ha portato l'azienda a ripensare completamente la propria organizzazione interna, con lo scopo di migliorare e rinnovare molti dei processi aziendali, produttivi e valutare nuove opportunità di mercato, come esempio la nascita del team dedicato completamente allo sviluppo nel 2018 con le relative implicazioni del caso per offrire soluzioni dedicate ai clienti. Offrono inoltre occasioni di stage, allo scopo di incorporare nuove menti e nuove idee e di formare i nuovi membri nelle discipline correlate al dominio di interesse.

- **Innovazione tecnologica**

  Come partner di SAP, Qlik, Microsoft, Tableou e come consulente di aziende leader di mercato, Méthode mira costantemente all'innovazione delle proprie soluzioni e quindi delle tecnologie e degli strumenti di lavoro. Propone già soluzioni di analisi predittiva il cui futuro è vicino al machine learning e all'intelligenza artificiale, ricerca e impiega le più moderne tecniche per la gestione di database e grosse moli di dati e recentemente sta lavorando su alcuni progetti di IoT^G (il mio stage curricolare è parte integrante di tale ottica aziendale) e di real-time^G *monitoring*. Quando una nuova tecnologia o soluzione è stata provata e giudicata utile e stabile vengono organizzati dei corsi interni per la formazione del personale che la utilizzerà(ex. sezione 1.4.3 voce: React per il team di sviluppo), si redigono poi guide all'uso e collezioni di best practice che si espanderanno nel tempo. A riprova di ciò come già citato in precedenza l'azienda tende ad aggiornarsi sulle nuove tecnologie e a valutare nuove opportunità di business da poter perseguire.

- **Promozione aziendale**

  Organizza approfondimenti, eventi, *webinar* ed altre iniziative destinati ad enti esterni per pubblicizzare e diffondere i vantaggi dei nuovi approcci della Business Intelligence^G e delle nuove soluzioni tecnologiche.

Capitolo 2

Lo stage

2.1 Motivazioni

Questa sezione spiega le motivazioni per le quali ho scelto di svolgere lo stage presso Méthode S.r.l. e quelle per cui l'azienda è interessata ad adottare tale strumento di assunzione.

2.1.1 Motivazioni personali

Ho avuto modo di conoscere tale realtà lavorativa grazie all'evento di Stage-IT. Le ragioni principali che mi hanno spinto a scegliere Méthode per il mio stage curricolare sono le seguenti:

1. **Sede:** La sede dell'azienda si trova a poco meno di quattro chilometri dalla mia residenza, ed è raggiungibile in circa un quarto d'ora di bici. Grazie a questa vicinanza ho risparmiato tempo e costi di trasporto;
2. **Progetti:**
   - Lo *stack* tecnologico dell'azienda mi è sembrato molto stimolante, approfondire alcune librerie JavaScript come React, che vengono attualmente utilizzate sempre di più nello sviluppo di interfacce grafiche di applicazioni web moderne;
   - La tematica trattata (anche se in modo parziale) dallo stage, relativa allo sviluppo di alcuni servizi cloud^G (API^G REST^G) in ambito IoT^G, l'ho ritenuta molto interessante da esplorare;
   - L'Obiettivo finale di dare una visione d'insieme in ottica fullstack^G, ovvero partecipare allo sviluppo sia frontend^G che backend^G mi è sembrato molto stimolante.
3. **Altre:**
   - esplorare il mondo della Business Intelligence^G (ambito a me sconosciuto);
   - ne avevo sentito parlare bene da ex-studenti che poi hanno iniziato a lavorare lì dopo lo stage.

2.1.2 Motivazioni aziendali

Proponendo contratti di stage a giovani pronti a mettersi in gioco può rivelarsi un'ottima scelta dal punto di vista aziendale:

1. La scelta di attivare progetti di *stage* universitari, o l'assunzione di giovani laureati/laureandi con un contratto di *stage*, è ritenuta un'ottima opportunità per le strategie aziendali.

2. L'azienda ha un potenziale guadagno, in quanto avendo a disposizione nuove risorse umane da formare come consulenti negli ambiti di interesse. L'azienda può rientrare da questo investimento grazie ad una potenziale futura assunzione dello stagista, avendo già una visione complessiva delle sue abilità.

3. Le nuove conoscenze che lo stagista ha, fresco dal percorso di studi compiuto e spesso aggiornato su qualche nuova tecnologie può portare ai colleghi che lavorano già da anni in azienda nuovi spunti d'innovazione e riflessioni costruttive.

2.2 Il progetto

Méthode, attraverso lo strumento dello *stage*, mira sia ad identificare personale da formare sia ad esplorare le novità e gli aggiornamenti di strumenti e tecnologie. Il progetto di stage oggetto di questa relazione è di natura sopratutto formativa e pratica.
Il progetto si divide in tre parti principali:

- lo studio delle tecnologie aziendali ed in particolare del linguaggio JavaScript (parte dello *stack* tecnologico utilizzato in azienda dal team di sviluppo);

- l'esplorazione di parte delle soluzioni cloud^G dedicate al data warehouse^G, data entry^G in ambito IoT^G, utilizzando piattaforme cloud^G dedicate;

- mettere in pratica le conoscenze acquisite, in particolare contribuire in modo discretamente autonomo allo sviluppo di prodotti professionali.

Inoltre, punta fortemente a formare una potenziale risorsa, facendomi partecipare allo sviluppo della parte frontend^G con lo *stack* tecnologico utilizzato dall'azienda in ottica fullstack^G.

2.2.1 Perché esplorare l'IoT

La necessità di quest'esplorazione da parte dell'azienda, deriva dal fatto che la Business Intelligence^G è direttamente collegata ad essi, visto la mole di dati che questi oggetti "intelligenti" possono tracciare ed elaborare ed i relativi casi d'uso che ne derivano, in modo da poter offrire soluzione sempre più all'avanguardia ai suoi clienti.

2.2.2 Team designato

L'azienda è internamente suddivisa in *team* per aree di competenza (vedi sezione 1.2). Per poter svolgere al meglio il mio stage sono stato inserito nel ***team* di sviluppo** composto da circa 8 persone (di diversi livelli di *seniority*), il cui team leader è stato anche il mio *tutor*. Per ciascuna tipologia di sviluppo, in seguito nel capitolo 3, Frontend sezione 3.2 e Backend sezione 3.3/3.4 verrà data una descrizione chiara di ciascuna tecnologie impiegata, il contesto di utilizzo ed eventuali approfondimenti dove ritenuti interessanti.

2.3 Vincoli

In questa sezione vengono indicati i diversi vincoli relativi al piano di lavoro e delle attività pianificate dello *stage*.

2.3.1 Vincoli metodologici

In accordo con l'azienda, si è deciso che avrei svolto lo *stage* presso la sede principale a San Vendemiano (TV), per favorire il dialogo, il confronto col *tutor* ed il resto del personale (nonostante i protocolli sanitari imponessero una capienza limitata del personale durante quel periodo).
La metodologia aziendale richiede inoltre che, su base giornaliera, ogni persona compili un rapportino telegrafico per la rendicontazione dell'attività svolte mediante lo strumento interno Web Office (sezione 1.5.1).
Io ed il *tutor*, abbiamo stabilito di effettuare un confronto orale più dettagliato che permettesse un costante allineamento ed il tempestivo emergere di problematiche.
Per il periodo iniziale il resoconto avrebbe avuto luogo al termine di ogni attività, riducendone la frequenza sulla base del grado di autonomia e affidabilità raggiunta di settimana in settimana. Nei casi in cui uno scambio orale non fosse praticabile l'avremmo sostituito con la *chat* di Google descritta in sezione 1.5.3.
Avrei svolto tutte le attività con il computer portatile ad uso personale fornitomi, portandomelo a casa, visto che in azienda viene praticato anche lo *smart working*, che ho potuto praticare per alcuni giorni durante la mia permanenza.

2.3.2 Vincoli temporali

Lo *stage* curricolare prevede una durata compresa tra le 300 e le 320 ore complessive. L'azienda ha ripartito uniformemente queste ore in 10 settimane lavorative da 40 ore l'una (eccetto le prime 4 settimane su mia necessità), per un totale di 320 ore che avrei svolto nel corso di due mesi e mezzo in osservanza dell'orario lavorativo aziendale:

- dal Lunedì al Venerdì, dalle 09:00 alle 18:00
- pausa pranzo tra le 13:00 e le 14:00 circa
- flessibilità oraria d'ingresso dalle 08:30 alle 09:15.

Nella proposta di *stage*, l'azienda ha indicato una ripartizione temporale su base settimanale delle attività e i contenuti di massima previsti:

Legenda Tabella 1:
Formazione
affiancamento attività clienti
affiancamento attività clienti con grado di autonomia elevato

Tabella 1: Sintesi piano di lavoro iniziale, suddiviso per tipologia di attività

Settimana/e	Ore	Attività	Descrizione
1	20	Concetti di Node.Js	Formazione strumenti aziendali, JavaScript
2	20	Concetti di sviluppo servizi cloud	Formazione infrastruttura di AWS^G(ed Azure) e documentazione relativa.
3,4,5	80	Sviluppo applicazione Node.Js(Azure)	Affiancamento risorse senior su attività clienti per mettere in pratica i concetti appresi.
6	40	Concetti di sviluppo interfacce grafiche web	Formazione framework React e Redux ed affiancamento attività clienti
7,8	80	Sviluppo applicazione su react/affiancamento attività clienti	Sviluppo prettamente autonomo con React e Redux
9,10	80	Sviluppo applicazione con React e Node.Js	Sviluppo fullstack^G prettamente autonomo con JavaScript, SQL, Azure

2.3.3 Criteri di adattamento piano lavorativo

Nei primi giorni ho discusso con il *tutor* alcune tematiche relative allo stage e al piano di lavoro:

- **Evitare esercizi fini a sé stessi**

Le attività di formazione prevedono degli esercizi per consolidare quanto visto e accertarne la comprensione; qualora l'argomento fosse stato compreso velocemente avremmo evitato di investire tempo in esercizi sterili, sostituendoli, secondo necessità, con piccole attività per conto di clienti;

- **Adattamento pianificazione temporale**
  È difficile stilare un piano preciso senza aver avuto modo di conoscere la persona e le sue capacità concrete; per migliorare la comprensione degli argomenti più ostici ed evitare ripetizioni infruttuose avremmo modificato la durata prevista per le attività secondo necessità;

- **Possibilità di adattare il percorso formativo**
  Parte delle tecnologie potrebbero creare problemi, l'indisponibilità temporale di risorse umane necessarie, il completamento in anticipo di un'attività e nel momento in cui un avveniva, avremmo vagliato soluzioni alternative insieme al *tutor*.

2.4 Aspettative

2.4.1 Aziendali

L'azienda al termine del periodo di stage si aspetta che le componenti software pianificate per il mio stage siano state sviluppate. È necessaria, dunque, una buona comprensione degli argomenti esposti durante i corsi di formazione e l'acquisizione di conoscenze sufficienti a garantire un buon grado di autonomia nell'utilizzo delle tecnologie richieste. L'azienda è interessata ad approfondire parte della tematica dell'IoT^G, in particolare la parte relativa all'infrastruttura(cloud^G) con cui si possono integrare i relativi servizi di backend^G. Gli obiettivi, di natura formativa e di natura esplorativa, sono stati aggiornati, integrati e ripartiti secondo due tipologie: obbligatori e desiderabili. Gli obiettivi aggiuntivi invece sono emersi in corso d'opera dall'opportunità su richiesta di un cliente o per concordate secondo necessità insieme al *tutor*.

Tabella 2: Tabella riassuntiva obiettivi fissati dall'azienda

Tipo	Descrizione
Obbligatorio	Formazione nelle tecnologie richieste dallo stage ovvero: JavaScript, NodeJS, React, Redux, SQL...
Obbligatorio	Concetti di sviluppo servizi cloud^G e creazione collaborativa di alcuni di essi (con Azure)
Obbligatorio	Stesura documentazione infrastruttura (progetto basato su AWS^G)
Obbligatorio	Concetti di sviluppo di componenti grafiche web semplici con la libreria React
Obbligatorio	Test di componenti frontend^G in produzione, sviluppate con la libreria React
Desiderabile	Sviluppo prettamente autonomo di parte delle API^G REST^G con NodeJs sulla piattaforma Azure
Desiderabile	Sviluppo prettamente autonomo di componenti frontend^G con la libreria React
Aggiuntivo	Sviluppo con C# (Blazor) e SAP Data Services di componenti frontend^G

Tabella 2: Tabella riassuntiva obiettivi fissati dall'azienda

Tipo	Descrizione
Aggiuntivo	formazione base dati relazionale(column-oriented)
Aggiuntivo	Stesura documentazione infrastruttura deploy progetti basati su SAP Hana e React
Aggiuntivo	Sviluppo di cloud^G *functions* per il data entry^G nella piattaforma cloud Azure

2.4.2 Personali

Questa è la mia prima esperienza lavorativa in un contesto aziendale in ambito informatico una opportunità di mettere in pratica quanto appreso negli anni di studi.
Nel corso degli studi l'applicazione pratica dei concetti appresi è sempre avvenuta per mezzo dei progetti didattici culminati con il progetto di Ingegneria del Software. Queste esperienze formative mi hanno permesso di fissare i concetti studiati anche alla luce delle problematiche incontrate e risolte, ma restano comunque almeno in parte di natura accademica.
In particolare, vedere le differenze e i diversi approcci tra il mondo accademico degli studenti e quello aziendale con i suoi vincoli metodologici, scadenze e clienti.
I progetti accademici mi hanno facilitato l'apprendimento dato che oltre ad imparare nuove nozioni, si "impara" ad imparare; confido che l'esperienza lavorativa, con il suo approccio più pratico, costituisca il proseguo di questa strada aggiungendo ulteriori sfide da affrontare e potenzialmente nuovi ostacoli da superare.
Mi aspetto che il mondo lavorativo possa offrire nuovi stimoli e possibilità di approfondimento e crescita diversi, rispetto all'ambiente accademico; in particolare che non sia solo una mera applicazione di quanto studiato in aula o il consolidamento di nozioni teoriche preesistenti, ma che sia fonte di nuovi argomenti e scenari che valgano la pena essere approfonditi ed appresi in un'ottica di una crescita professionale.

Capitolo 3

Resoconto dello stage

Durante il periodo iniziale dello stage ho discusso il piano di lavoro predisposto con il *tutor* aziendale individuando le attività principali. L'esito di questa discussione è argomentato nel capitolo 2: in particolare nella sezione 2.3.3, l'eventuale adattamento formativo delle attività inizialmente previste dalla tabella Pianificazione delle attività. Come accennato in sezione 2.3.1, al termine di ogni attività ho effettuato un resoconto orale o telematico che è stato di volta in volta spunto di partenza per le attività successive ed occasione per evidenziare eventuali problematiche incontrate.
Durante lo stage ho praticato formazione individuale(documentazione,tutorial,...), collaborativa(partecipando ad attività di DevOps^G e di pair programming^G), ho documentato l'infrastruttura stilando una guida per la gestione ed il *deploy* di alcuni progetti, in particolare ho lavorato in attività clienti, sia lato frontend^G che backend^G, con la supervisione da parte di alcuni programmatori. Al fine di raggiungere un grado di autonomia e affidabilità che mi consentisse di svolgere in autonomia le attività designatemi.
Nelle successive sezioni vengono discussi caso per caso gli ambiti di sviluppo e le tecnologie impiegate nell'esecuzione delle attività pianificate.
Inoltre, nella sezione 3.7 vengono approfondite più in dettaglio le attività svolte e le relative tempistiche con una autovalutazione personale del grado di autonomia rispetto alla tabella degli obiettivi aziendali in sezione 2.4.1.

3.1 Stack principale

A seguire lo *stack* tecnologico principalmente utilizzato durante il mio stage, ovvero un sottoinsieme di quello utilizzato in azienda dal *team* di sviluppo.

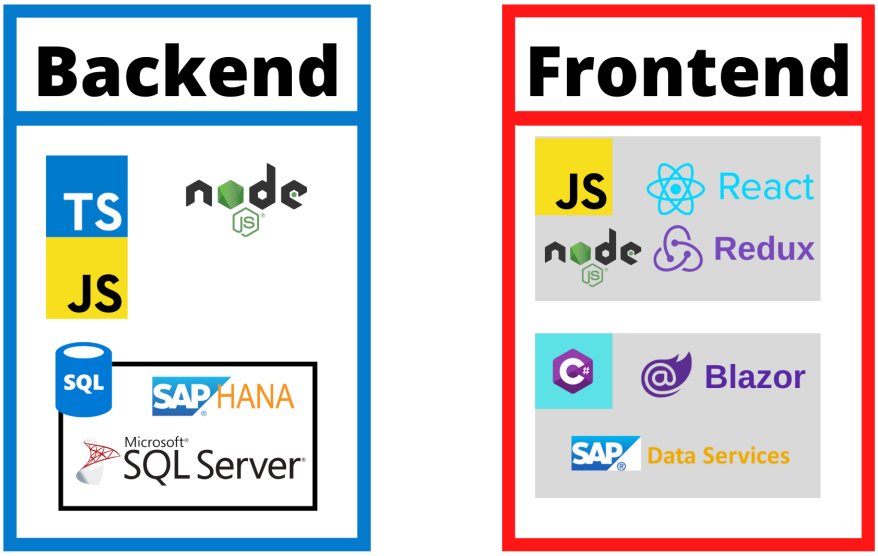


Figura 4: Infografica dello *stack* tecnologico principalmente utilizzato da me durante lo stage

Come si può notare dall'immagine sopra, NodeJs ed il linguaggio JavaScript sono stati utilizzati sia dalla parte backend^G che da quella frontend^G, visto che le librerie richieste (pacchetti npm ex. React), si basano sul linguaggio JavaScript e necessitano di un *engine* (ex. V8 usato sulla maggior parte dei browser odierni per eseguire codice JavaScript) affinché il codice possa essere compilato e quindi successivamente utilizzato. Nel caso di sviluppo backend^G è **strettamente** necessario l'utilizzo di un vero e proprio runtime system^G (ex. Node.Js), mentre per il (frontend^G) si potrebbe includere lo *script* come una classica pagina Web per poterlo utilizzare.

3.1.1 Node.Js

Node.Js è una tecnologia non trattata (al momento) dalla laurea triennale, ma l'avevo già usata in progetti personali. Questa tecnologia, è un runtime system^G, che permette di creare applicazioni completamente in JavaScript, ovvero tramite il paradigma ***JavaScript everywhere***, rispetto a prima dell'uscita di quest'ultima, dove la parte di backend^G doveva essere per forza scritta in un linguaggio differente(esempio PHP).
Tale tecnologia (Node.Js) l'ho utilizzata sia nella parte frontend^G per installare le relative librerie come ex. React, generare il relativo codice (ex. pagine html, script in JavaScript, ...), inoltre, l'ho utilizzato anche nel backend^G installando i pacchetti(npm) necessari allo sviluppo del codice relativo ai servizi cloud^G e successivamente nell' esecuzione di tale codice.

3.1.2 npm

npm (abbreviazione di *Node Package Manager*) è un gestore di pacchetti per il linguaggio di programmazione JavaScript. È il gestore di pacchetti predefinito per l'ambiente di runtime (runtime system^G) JavaScript Node.js. Consiste in un *client* da linea di comando, chiamato anch'esso npm ed offre un database online di pacchetti/librerie pubblici e privati, chiamato *npm registry*. Il *registry* è accessibile via *client* e i pacchetti disponibili sono consultabili sul sito web di npm. Il gestore di pacchetti e il *registry* sono gestiti da **npm, Inc**. In particolare, ho utilizzato questo registro per la selezione e la visione della relativa documentazione delle librerie necessarie a svolgere le attività richieste, in particolare nella parte di backend^G.

3.2 Sviluppo frontend

Parte delle tecnologie utilizzate dal team di sviluppo per il frontend^G, richiede la conoscenza del linguaggio JavaScript, data la mia precedente esperienza col linguaggio, ho potuto ripassare velocemente (in circa 4 ore) i concetti principali in prima persona.
Per la maggior parte del tempo dedicato alla parte frontend^G si è utilizzata la metodologia Agile^G in particolare Scrum^G, con tale approccio si cerca di incrementare le *features* di un prodotto tramite dei cosiddetti *sprint*. Lo sviluppo a cui ho partecipato nella parte frontend^G, si può suddividere in sviluppo di componenti grafiche basato su:

1. JavaScript (React, Redux...) progetto webapp(SPA^G) in ambito controllo qualità;
2. C#(.NET,Blazor,SAP Data services),tecnologie relative ad un attività non inclusa nel piano di lavoro iniziale in sezione 2.3.2, ma nata da un'opportunità relativa ad un cliente(vedi tabella degli obiettivi aziendali in sezione 2.4.1), intrapresa concordandola prima col *tutor*(vedi sezione 3.2.3).

Proseguendo in questa sezione verranno trattate le tecnologie utilizzate durante lo stage relativo allo sviluppo di componenti frontend^G, dandone una descrizione per ognuna di esse (modalità di apprendimento,contesto di utilizzo,eventuali confronti con altre tecnologie o criticità riscontrate durante l'utilizzo).

3.2.1 React

React è una libreria *open-source*, dedicata al front-end che utilizza JavaScript per facilitare la creazione di UI (interfacce utente). È mantenuta da Meta (ex Facebook), da una comunità di singoli sviluppatori e altre aziende. La "potenza" ed il nome di questo *framework* derivano dal fatto che consente di progettare interfacce per ogni stato di una applicazione. Ad ogni cambio di stato la libreria consentirà di aggiornare efficientemente solamente le parti della *UI* che dipendono da tali dati.

- **Formazione:** Tale libreria non è argomento specifico della triennale, ma avevo seguito breve corso fatto da alcuni studenti della magistrale in informatica di Padova ed inoltre lo ho utilizzato per lo sviluppo del mio portfolio. Ho ripassato tramite un

video-corso individuale i concetti principali da conoscere del *framework* e seguendo programmatori senior in attività clienti per poi applicarli in autonomia, chiarendo eventuali dubbi.

- **Contesto di utilizzo:** Questa tecnologia l'ho sfruttata in diversi progetti di applicazioni web durante lo stage, in particolare insieme a librerie come *Material UI* per la realizzazione di componenti frontend[G]. Solitamente la libreria Material UI viene utilizzata assieme ad altre librerie/moduli come Redux e React Router che rispettivamente consentono di gestire gli stati delle varie componenti grafiche(in modo particolarmente efficiente) ed il *routing* delle pagine.

Inoltre, ho adottato anche un'estensione per il browser che aiuta nel *debug* dando in dettaglio informazioni di ciascuna componente *UI* sviluppata col *framework* React.

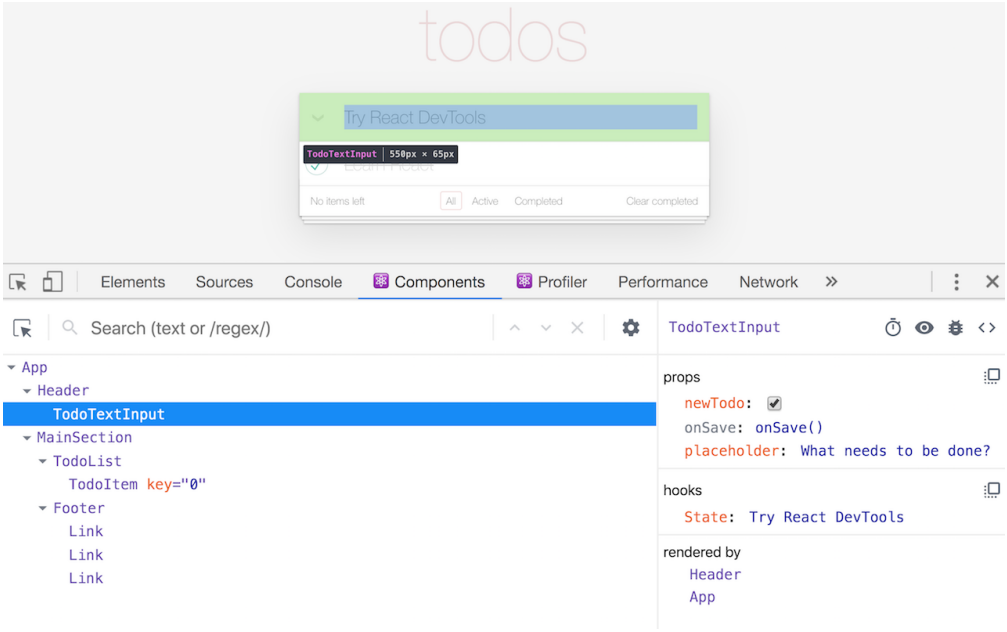


Figura 5: L'estensione React Developer Tools, esempio di cosa si può vedere da questa estensione.

fonte: mozilla.org

Di seguito vengono forniti alcuni *screenshot* delle componenti *UI* sviluppate per la webapp (*SPA*[G]):

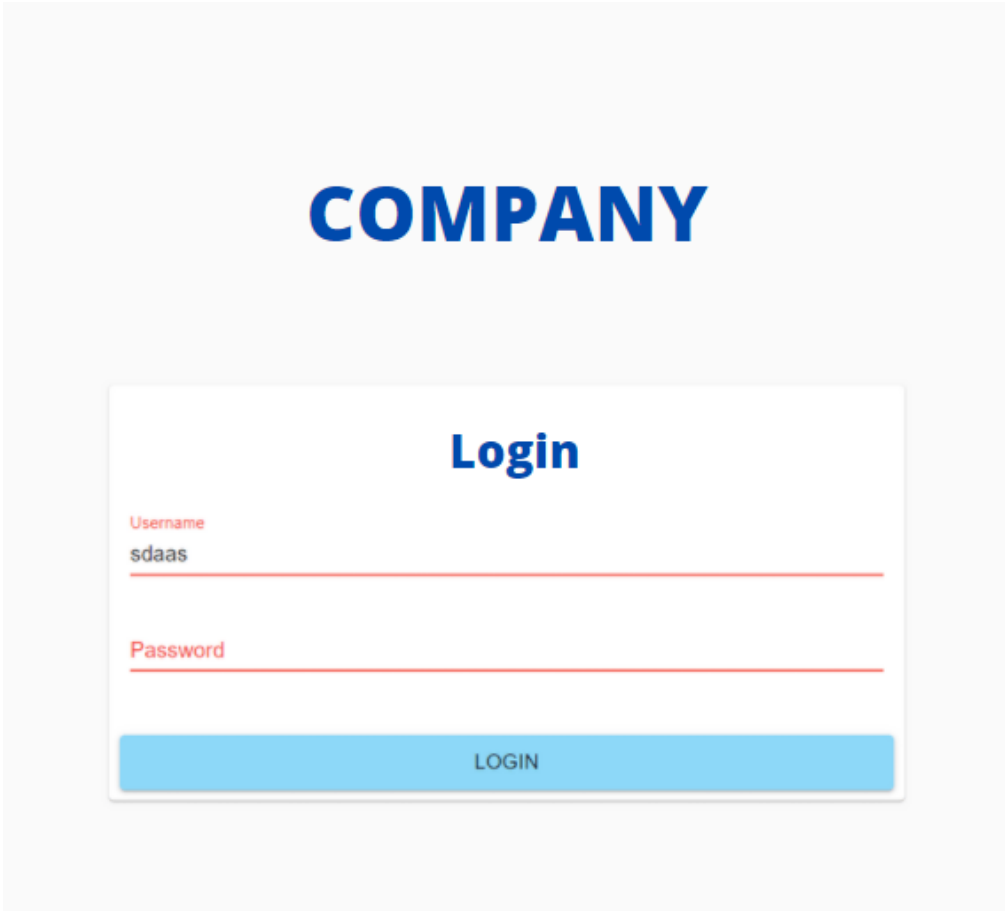


Figura 6: *Screenshot* errore *form* di *login* sviluppata con React e MaterialUI per la webapp (SPA^G)

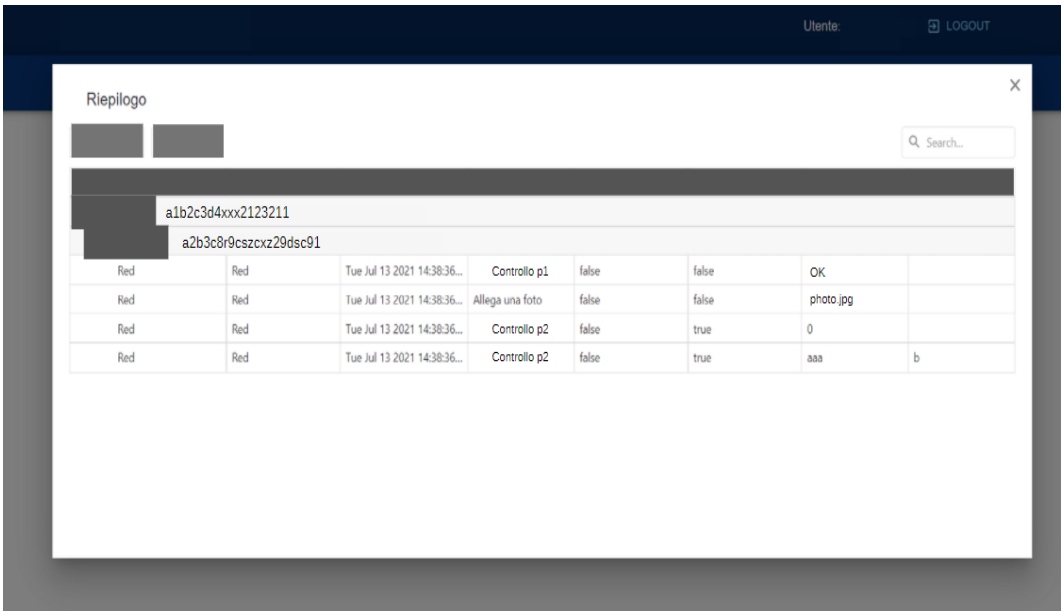


Figura 7: *Screenshot* UI tabella di riepilogo sviluppata per la webapp(SPA) con React, Redux ed un'altra libreria(DevExtreme) basata su React

3.2.2 Redux

Redux è una libreria JavaScript *open-source* per la gestione e la centralizzazione dello stato dell'applicazione. È più comunemente usata con librerie come React e altre librerie JavaScript dedicate al frontend^G.

- **Formazione:** Tale libreria mi era completamente sconosciuta, per cui ho ritenuto opportuno approfondire lo studio tramite un breve videocorso (di circa 3 ore) gratuito fatto dal creatore della libreria, ovvero Dan Abramov(che ho seguito una seconda volta in modo da capirne bene il suo funzionamento). Dopo il videocorso ho partecipato a qualche attività di DevOps^G e pair programming^G, in cui all'occorrenza ponevo domande ai programmatori senior che chiarivano i dubbi che mi sorgevano.

- **Contesto di utilizzo:** Come pianificato inizialmente, in autonomia ho iniziato ad utilizzare tale libreria in attività clienti sviluppando componenti frontend^G, con la supervisione del codice da parte degli sviluppatori con più esperienza nell'uso della libreria fino ad ottenere un grado di autonomia abbastanza elevato da eseguire in autonomia tali attività.

Insieme a questa libreria (Redux) ho utilizzato un'estensione browser(sviluppata dai creatori di quest'ultima), molto utile per attività di debugging dello stato, utilizzandola in modo simile agli strumenti per sviluppatori proposto dai browser.

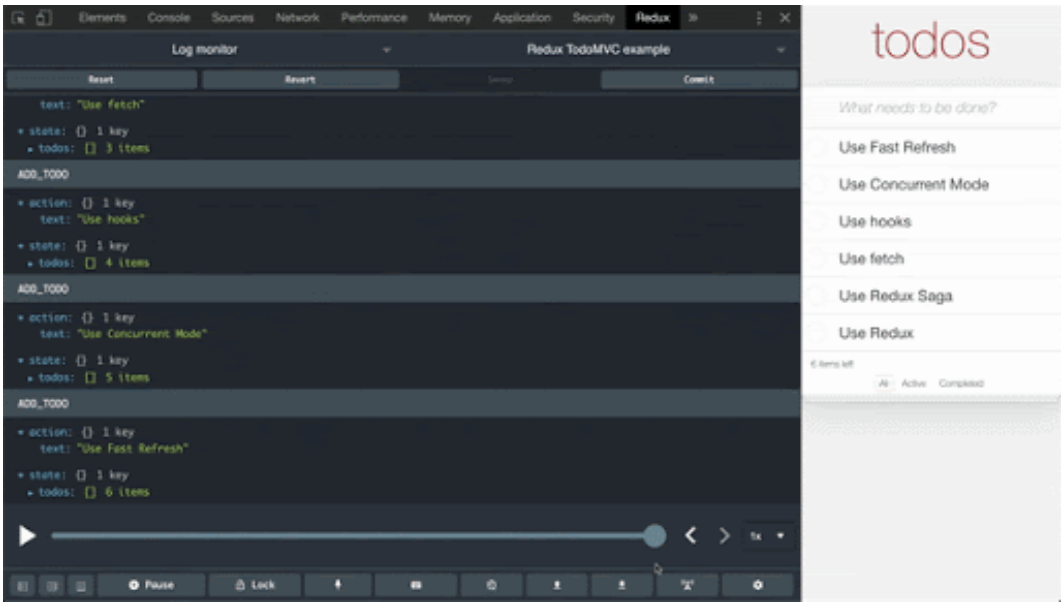


Figura 8: Screenshot dell' estensione Browser che permette di vedere dei "log" dei cambiamenti di stato delle varie componenti.
fonte: logrocket.com

Di seguito una delle componenti che ho sviluppato che consente di simulare l'elaborazione di dati, in modo da migliorare l'esperienza utente (UX), nella quale ho potuto apprezzare la gestione dello stato da parte della libreria Redux.

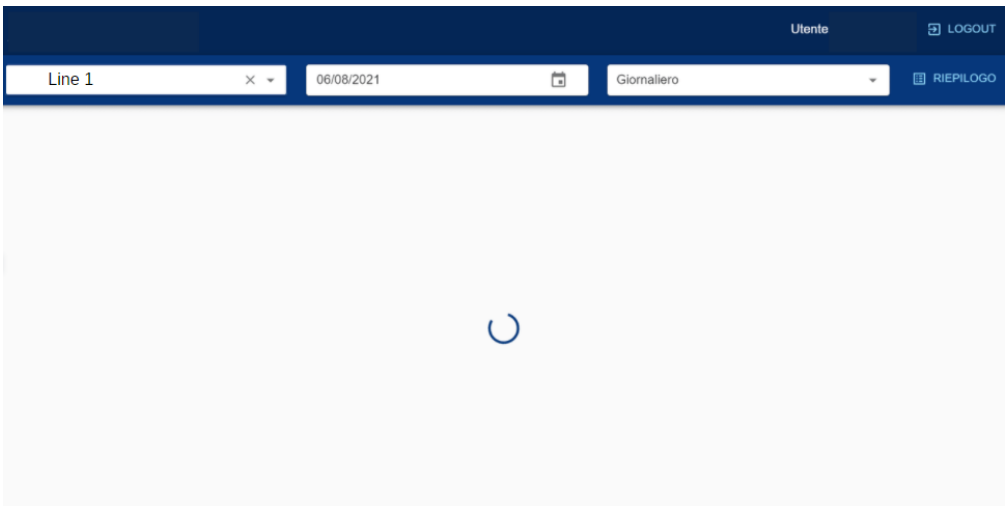


Figura 9: *Screenshot* di una delle componenti che fa comparire l'icona di caricamento quando lo stato (centralizzato) dell'applicazione sta per essere aggiornato(in questo caso specifico quando si clicca su riepilogo avendo selezionato una delle *entry* dai 3 menu a tendina), sviluppato per la Web app (SPA^G), relativa al controllo qualità.

3.2.3 SAP Data services, Blazor

In azienda la maggior parte del personale tecnico conosce almeno una porzione della *suite software* di SAP, avendo una *partnership* ufficiale di lunga data, visto che l'azienda è specializzata nella Business Intelligence^G, inoltre dato che lo stage ha anche il fine di una potenziale assunzione (nel mio caso mi era già stato proposto un ulteriore stage extracurricolare, come accennato in sezione 2.1.2), ho avuto modo di utilizzare parte di questa suite software di SAP.
Nello specifico ho utilizzato:

- SAP Data Services è un insieme di strumenti molto potente che consente di integrare dati tramite il processo di ETL^G, in particolare offre un insieme di *software* (classi, designer...) che facilitano molto l'integrazione di questo tipo di sistemi (ex. tramite un'interfaccia grafica), mappando dati (ex. database di vario genere, file csv...) e generando una UI con cui l'utente interagisce, al fine di ottenere un applicazione web completa. Tale software è basato sul SDK di Microsoft .NET (SAP e Microsoft sono partner di lunga data), che funziona in modo molto simile alla *JVM*. SAP comuque offre integrazioni simili anche con altri linguaggi (ex. Java).
  Rispetto alla controparte (SSIS) Microsoft, l'impressione ricevuta è stata che SAP offra una suite software vasta, efficiente e vantaggiosa (ex. tramite interfaccia grafica si possono fare operazioni molto complesse come mappare di tabelle in *join*, fonti multiple di dati anche in formati diversi, definire relazioni molto complesse tra oggetti di business...) per i vari casi d'uso della Business Intelligence^G;
- SSIS(Sql server integration services) di Microsoft (*partner* aziendale), sono gli strumenti che consentono di integrare dati tramite il processo di ETL^G, similmente alla controparte di SAP che mappa i dati sottostanti secondo logiche di business aziendali(requisiti), con cui si genera un'interfaccia utente, al fine di ottenere un applicazione web completa. L'impressione principale che ho avuto, è che il *framework* grafico Blazor consente una personalizzazione maggiore dell'interfaccia grafica rispetto alla controparte di SAP.

In conclusione, entrambi le soluzioni hanno lo scopo di integrare la logica di business aziendale, ovvero solo i dati necessari (ex. tabelle, viste di database,file csv...) contestualizzati nel business di competenza dei clienti finali e di generare un'interfaccia grafica con cui interagire generalmente tramite dei file di *design*, per avere un sistema (ex. gestionale) completo.

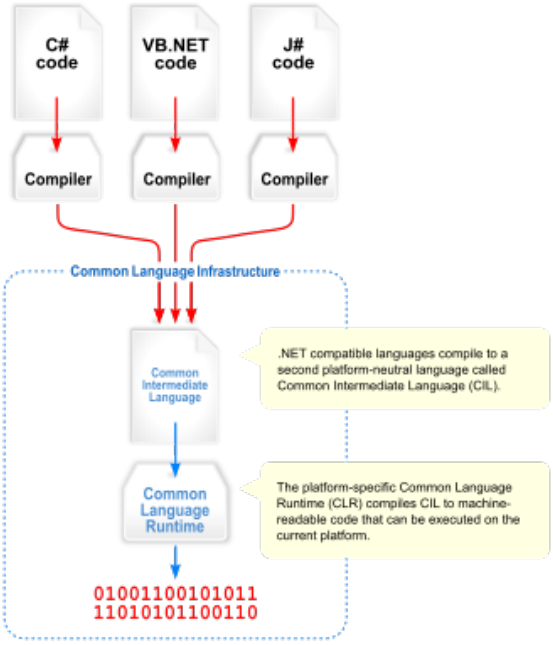


Figura 10: Esempio di come avviene il processo di compilazione del codice nella piattaforma Microsoft .NET.

fonte: Wikipedia.org

- **Formazione:** queste tecnologie (SAP Data Services, Blazor) mi sono state spiegate da un membro del team di sviluppo con esperienza rilevante in tali tecnologie, in 8 ore circa. La formazione è stata principalmente di carattere pratico, ovvero incentrata sul come utilizzare i vari *framework* chiarendo le loro principali differenze(pratiche), tramite esempi di codice.
- **Contesto di utilizzo:** ho utilizzato queste tecnologie per circa 40 ore, principalmente per la creazione di applicazioni web (sistemi gestionali), utilizzando come linguaggio principale C# (data la natura della piattaforma come citato in precedenza in questa sezione), senza l'ausilio di strumenti come interfacce grafiche(ex. designer). In sostanza ho avuto un'esperienza limitata con queste tecnologie rispetto ad altri *software* utilizzati con maggior intensità in altre attività dello stage.

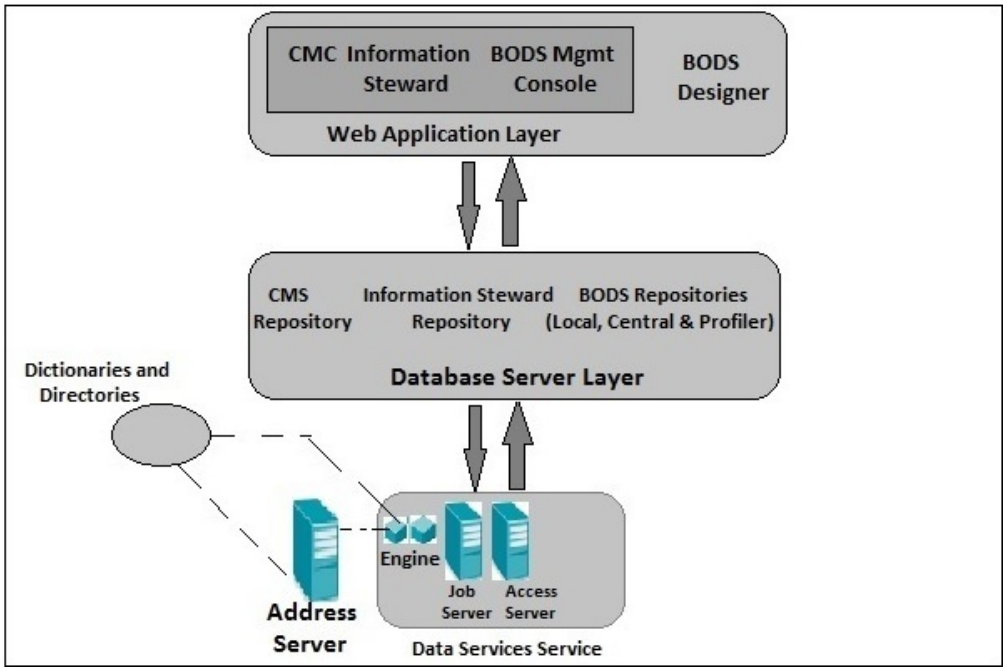


Figura 11: Esempio di un'architettura BODS (Business Objects Data Services) di SAP.
fonte: tutorialspoint.com

3.2.4 IDE

I principali IDE^G utilizzati per il frontend^G sono stati: Webstorm(JavaScript,React...), Visual Studio(.NET,C#), Eclipse(SAP Data Services), ciascuno dei quali facilità lo sviluppo offrendo diverse funzionalità(ex. autocompilazione di frammenti di codice...) oppure è necessario in base al contesto (ex. sviluppo su una piattaforma specifica).

3.3 Sviluppo backend

Lo sviluppo backend^G si è incentrato per lo più sullo sviluppo di:

- alcune API^G REST^G in un contesto IoT^G, insieme ad altre attività affini allo stesso progetto:
  - alcuni servizi cloud^G per il *data entry*^{G};
  - formazione *database* relazionali (*column-oriented*);
- stesura di documentazione relativa ad altri progetti in produzione(DevOps^G) attività concordata durante lo *stage* insieme al *tutor*.

Nelle successive sezioni di questo capitolo (3.3 e 3.4) saranno descritte:

- le attività lavorative a cui ho partecipato per la parte di backend^G
- il contesto principale (sviluppo dei servizi cloud^G)
- le tecnologie utilizzate per sviluppare tali servizi dandone una descrizione di esse (apprendimento, contesto di utilizzo, ...) con eventuali criticità riscontrate.

3.3.1 Servizi Cloud di AWS e Documentazione

Per svolgere tale compito, ho partecipato ad attività di DevOps^G e pair programming^G di un progetto basato sull'infrastruttura di AWS^G, per il quale dovevo stendere la documentazione relativa al *deploy* sull' infrastruttura cloud^G utilizzata e delle tecnologie impiegate, in particolare scrivere anche l'approccio da seguire per eventuali nuove estensioni del software (esempio: il come testare le nuove componenti) e le cose che ritenevo necessarie(come eventuali design pattern da seguire). Inoltre, ho partecipato ad attività di DevOps^G e pair programming^G in altri due progetti basati su React e SAP HANA (piattaforma cloud^G) stendendone la relativa documentazione(parte di queste attività non erano previste dal piano iniziale). Queste attività hanno richiesto circa 48 ore per il loro completamento.

3.3.2 Sviluppo servizi cloud su Azure

Una delle attività principali in cui ho impiegato circa 90 ore complessive, è stato lo sviluppo di alcuni servizi cloud^G API^G REST^G ed altri servizi per il data entry^G, tramite delle cloud^G *functions*, per un progetto che fa uso di IoT^G.

- **Contesto:** Ho partecipato, nello specifico alla parte di trasformazione, collezione dei dati(data entry^G,data warehouse^G) e la relativa trasmissione di questi, formattati in un determinato modo sulla rete, al fine di leggerli per scopi analitici(ex.dashboard^G), tramite questi servizi cloud^G (API^G REST^G).

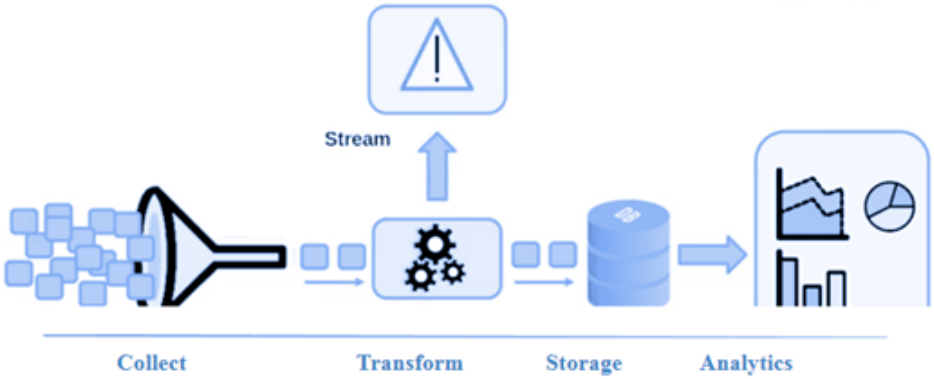


Figura 12: Esempio di pipeline di un progetto IoT^G.
fonte: faun.pub

- **Metodologia:** Per sviluppare questi servizi cloud^G principalmente ho utilizzato il metodo Agile^G dove iterativamente, concordando prima col *tutor*, si decidevano le attività da portare avanti, dandone un feedback al completamento delle stesse o su necessità.

- **Formazione:** Mi sono formato tendenzialmente durante il periodo iniziale dedicato a questo progetto, ma anche durante lo svolgimento, collaborando con altre risorse in base alle necessità, che si possono contraddistinguere nei seguenti modi:

  1. **attività collaborativa passiva:** Per fare ciò ho partecipato ad una sessione di introduzione all'infrastruttura cloud^G di Microsoft (uno dei partner dell'azienda), ovvero Azure, affiancando una risorsa dell'azienda che stava gestendo tale infrastruttura;
  2. **attività collaborativa attiva:** Per impostare il progetto e capire un po' quali librerie si potevano utilizzare, abbiamo fatto attività di pair programming^G insieme allo sviluppatore che gestiva i servizi dell'infrastruttura, in modo da discutere *best practices*, eventuali perplessità e problemi riguardo alle librerie da utilizzare, per poi agevolmente implementare i requisiti relativi al progetto in questione;
  3. **attività individuale:** andandomi a informare dalla documentazione ufficiale della piattaforma cloud^G di Azure e delle librerie utilizzate e proposte dal *partner*. Lo sviluppo di tali servizi cloud^G ed il relativo *testing* è stato fatto in modo prettamente autonomo, con la supervisione da parte del programmatore che gestiva tale infrastruttura.
  4. **collaborativa (con altro *team*):** durante lo sviluppo ho collaborato per la parte di *database* con il *team* dedicato a tale compito in azienda, cercando di comunicare in modo proattivo(aggiornando frequentemente la documentazione

relativa ai requisiti di sviluppo necessari per ciascun endpoint^G). Inoltre, durante lo sviluppo c'era la supervisione da parte di alcuni sviluppatori del *team* su mia richiesta o al termine di ogni attività.

Nella successiva sezione (3.4) viene trattato più in dettaglio lo *stack* tecnologico utilizzato per lo sviluppo di questi servizi cloud^G.

3.4 Stack tecnologico backend

Segue una descrizione riepilogativa dello *stack* tecnologico utilizzato per lo sviluppo di questi servizi cloud^G, ovvero delle:

- API^G REST^G seguendo la specifica di OpenAPI ;
- funzioni cloud^G per il data entry^G.

3.4.1 Node.Js

Ho utilizzato questa tecnologia installando i moduli npm necessari allo sviluppo dei servizi cloud^G, tale sistema è disponibile nella piattaforma Azure di Microsoft permettendo lo sviluppo *serverless*(senza dover gestire l'infrastruttura sottostante) in JavaScript come detto nella sezione 3.1.1(esempio una webapp completa oppure solo un servizio come possono essere delle API^G REST^G,...).

3.4.2 npm

È il gestore dei pacchetti JavaScript per Node.Js che ho utilizzato:

- **Contesto:** Oltre all'utilizzo e la visione della documentazione di librerie già installate nelle varie applicazioni a cui ho partecipato nello sviluppo. Ho potuto cercare, scegliere ed utilizzare alcune librerie necessarie a sviluppare questi servizi con la supervisione di risorse *senior*.
- **Problemi riscontrati** Uno di questi pacchetti presentava un problema di *encoding* con alcuni tipi di dato, quando si effettuava l'operazione di **BULK INSERT**, nonostante venisse consigliata dal partner ufficiale Microsoft. Ho segnalato la criticità agli *stakeholder* di interesse (*tutor*, colleghi e i manutentori della libreria).Inizialmente quando ho segnalato la *issue* ai manutentori della libreria non mi hanno dato particolari indicazioni, ciononostante sono andato avanti con lo sviluppo di questi servizi anche se alcuni endpoint^G avevano un problema di *performance*, ovvero l'inserimento o l'aggiornamento di dati di migliaia di righe richiedeva diversi minuti. L'ultima settimana di stage ho segnalato di nuovo il problema aprendo un'altra *issue* sempre sulla *repo* della libreria, questa volta spiegando più in dettaglio la parte dello *stack* di librerie dove ritenevo poteva esserci il problema, per poi ricevere risposta poche ore dopo da uno dei manutentori che mi ha informato che questa *issue* era già stata segnalata in altre due *issues*. Mi è stata quindi

data rassicurazione che egli avrebbe aumentato la priorità di quest'ultima *issue* e che probabilmente sarebbe stato risolto nel prossimo aggiornamento della libreria. Quindi l'unico punto rimasto in sospeso è stata l'ottimizzazione delle *performance* di alcuni di questi endpoint^G (necessaria vedi sezione 3.5.3). Tale inconveniente dovrebbe essere stato risolto dalla nuova versione della libreria.

3.4.3 TypeScript

È un linguaggio di programmazione *open-source* sviluppato da Microsoft. Si tratta di un *superset* di JavaScript che basa le sue caratteristiche su ECMAScript 6. Il linguaggio estende la sintassi di JavaScript in modo che qualunque programma scritto in JavaScript sia anche in grado di funzionare con TypeScript senza nessuna modifica. È stato progettato per lo sviluppo di grandi applicazioni ed è destinato ad essere compilato in JavaScript per poter essere interpretato da qualunque *web browser*.Tale linguaggio aggiunge una tipizzazione debole(i controlli non sono stretti come in C++ esempio, il controllo dei tipi è solo "statico"), ciò comunque consente di evitare errori e scrivere codice più pulito ("riduce notevolmente il numero di righe necessarie rispetto a JavaScript") dopo aver compilato il codice viene convertito in codice JavaScript per avere compatibilità con il resto delle tecnologie JavaScript coinvolte.

- **Formazione:**
  Dato la mia conoscenza di JavaScript, il relativo ripasso ed essendo TypeScript un *superset* mi sono documentato solo sulle cose dove avevo dei dubbi (ex. sintassi specifica), eventualmente usando la sintassi di JavaScript per poi effettuare un *refactoring* del codice se necessario (data la somiglianza di sintassi).

- **Contesto di utilizzo:**
  È stato il linguaggio principalmente utilizzato durante tutto lo sviluppo dei servizi cloud^G.

3.4.4 JSON, YAML

Sono 2 formati di file rispettivamente con estensione .json e .yaml/.yml. JSON è un formato di file *standard*(open-source) di interscambio dati che utilizza testo leggibile dall'uomo per archiviare e trasmettere oggetti(dati) costituiti da coppie chiave-valore e array (o altri valori da serializzare). È un formato di dati comune con usi diversi nello scambio di dati elettronici, incluso quello delle applicazioni web con i *server*.

- **Formazione:**
  Tale formato file lo avevo già utilizzato in ambito accademico, sia nel progetto del corso di programmazione ad oggetti, che in quello di ingegneria del software. Mentre il formato YAML non lo avevo utilizzato ma è molto simile al formato JSON dato che è un *superset* del JSON; infatti, si possono usare in contesti simili, anche se come convenzione, per i file di configurazione si tende a preferire il formato YAML perché ha regole più rigide (richiede l'indentazione per indicare il livello di profondità che si vuole dare all'oggetto) ed è di più facile lettura rispetto al JSON.

- **Contesto di utilizzo:**
  Nello specifico ho utilizzato il formato YAML per il file di configurazione relativo ai vincoli di validazione dei dati in *input* e *output* dai vari endpoint^G, mentre i dati in input ed output (la trasmissione di essi) era in formato JSON.

3.4.5 Azure Cloud Functions (Serverless)

Per sviluppare i servizi cloud^G l'infrastruttura che ho utilizzato è stata la piattaforma Azure(Microsoft),in particolare ho utilizzato uno dei servizi offerti, ovvero le cloud^G *functions*, detto anche *Function as a service*, che è un servizio cloud^G *serverless*, nel quale si paga solo le risorse che vengono utilizzate effettivamente, con una tariffa a tempo, con un costo minimo per ogni chiamata a tali funzioni. Anche in termini di sicurezza si possono definire politiche di sicurezza relativi all'accesso di queste risorse come esempio: filtraggio tramite IP, definire dei veri e propri *token* di autenticazione, come quelli generalmente utilizzati nel web per garantire l'accesso a determinate risorse in modo sicuro (ex. profilo utente di un *social network*), limitare l'accesso al tipo di servizi (ex. un singolo endpoint^G o solo uno specifico servizio...) Quest'infrastruttura supporta diversi linguaggi, oltre alle tecnologie (ex. Node.Js) e linguaggi già citati precedentemente in questa sezione. Come IDE^G ho utilizzato Visual Studio Code con diverse estensioni (spesso direttamente consigliate e sviluppate da Microsoft) per lavorare e testare simulando lo stesso ambiente cloud^G in locale per poi effettuarne il *deploy*. Ho ampiamente utilizzato la documentazione di Azure, in particolare quella relativa allo sviluppo di questo tipo di servizi(API^G REST^G), in cui vengono date diverse linee guida e consigli sulle eventuali librerie da utilizzare a seconda dei casi d'uso e delle tecnologie che si vuole utilizzare, oltre ai *template* di codice da cui partire.

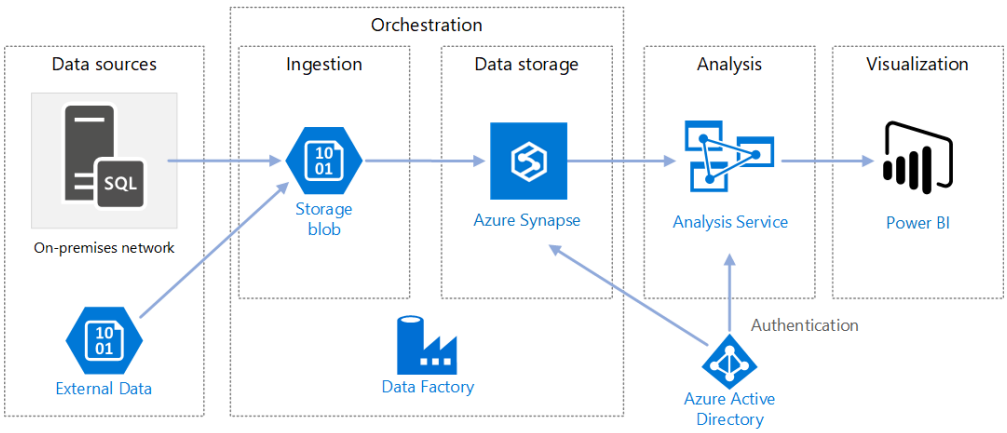


Figura 13: Esempio di architettura basata sull'infrastruttura di Azure per casi d'uso connessi con la Business Intelligence^G.
fonte: docs.microsoft.com

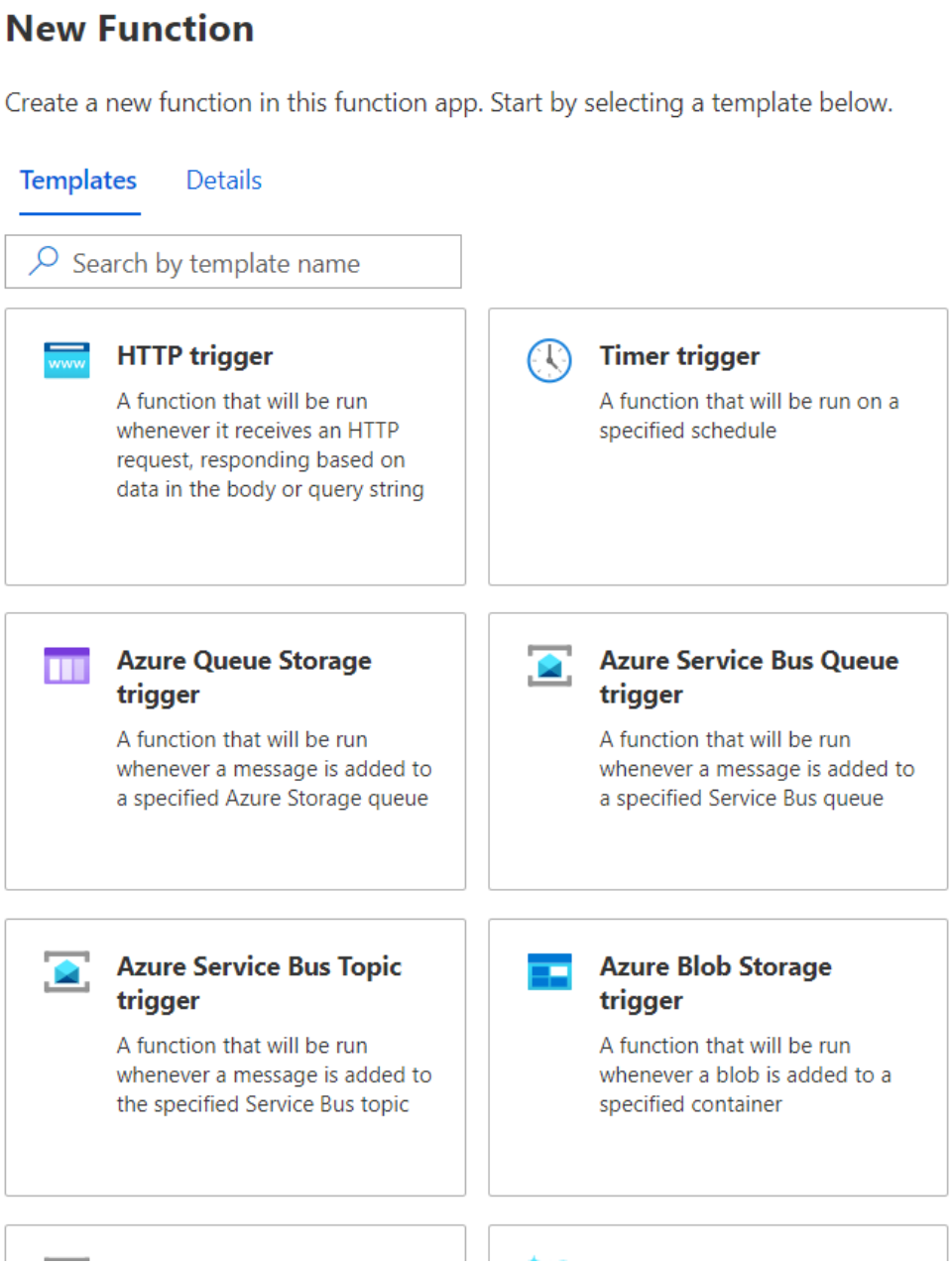


Figura 14: Schreenshot di parte della dashboard^G disponibile dentro la piattaforma Azure che permette di creare una funzione utilizzando dei *template* di partenza in base al caso d'uso.

fonte: docs.microsoft.com

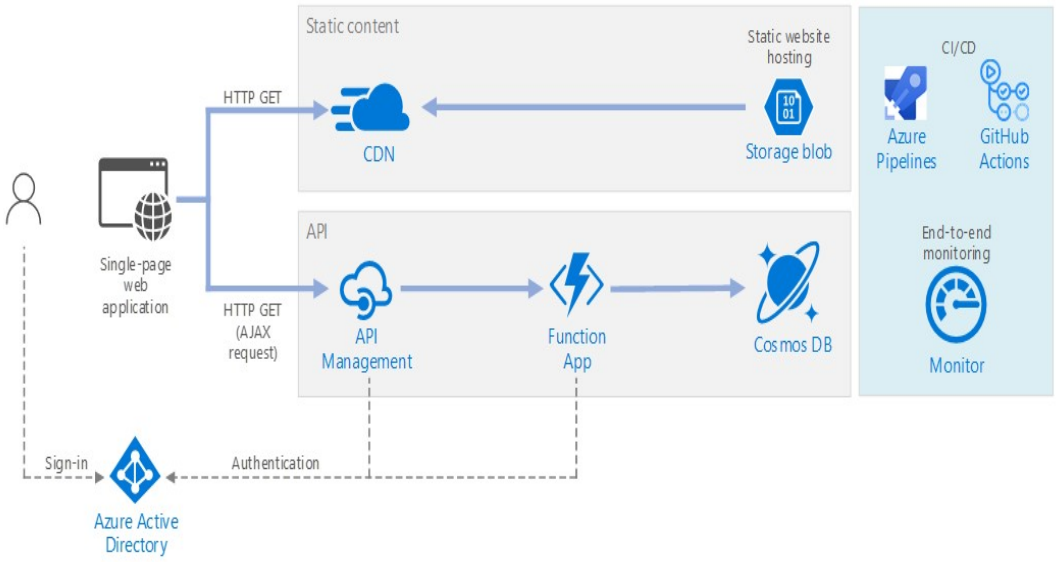


Figura 15: Esempio di architettura di un'applicazione Web *serverless*. L'applicazione rende disponibile il contenuto statico di Archiviazione *BLOB* di Azure e implementa una API[G] REST[G] che usa uno uno dei servizi di funzioni cloud[G] di Azure. L'API legge i dati di Cosmos DB e restituisce i risultati per la Webapp.

fonte: docs.microsoft.com

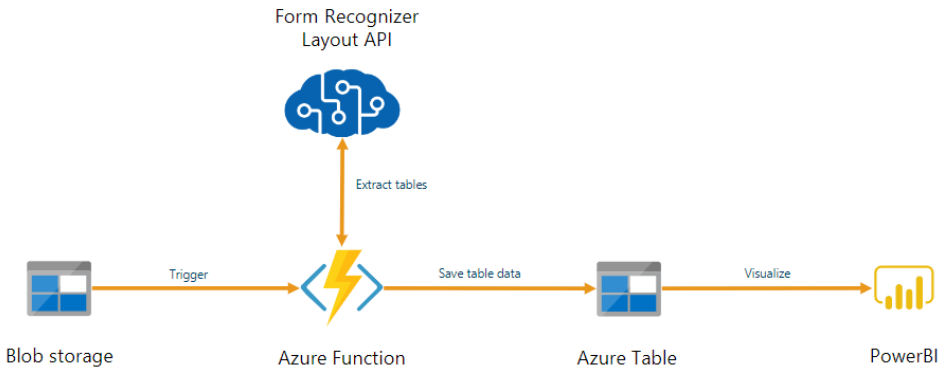


Figura 16: Esempio di utilizzo di una funzione di Azure per elaborare i documenti archiviati, dove un *trigger*(evento) che richiama esecuzione di tale funzione (ex. il caricamento di un file, immagine...).

fonte: docs.microsoft.com

3.4.6 Postman

È una piattaforma API per la creazione e l'utilizzo di API. Postman semplifica ogni fase del ciclo di vita dell'*API* e ottimizza la collaborazione in modo da poter creare API migliori, più velocemente. Quando si parla di sviluppo di API, l'utilizzo di Postman può essere considerata una *best practice*.

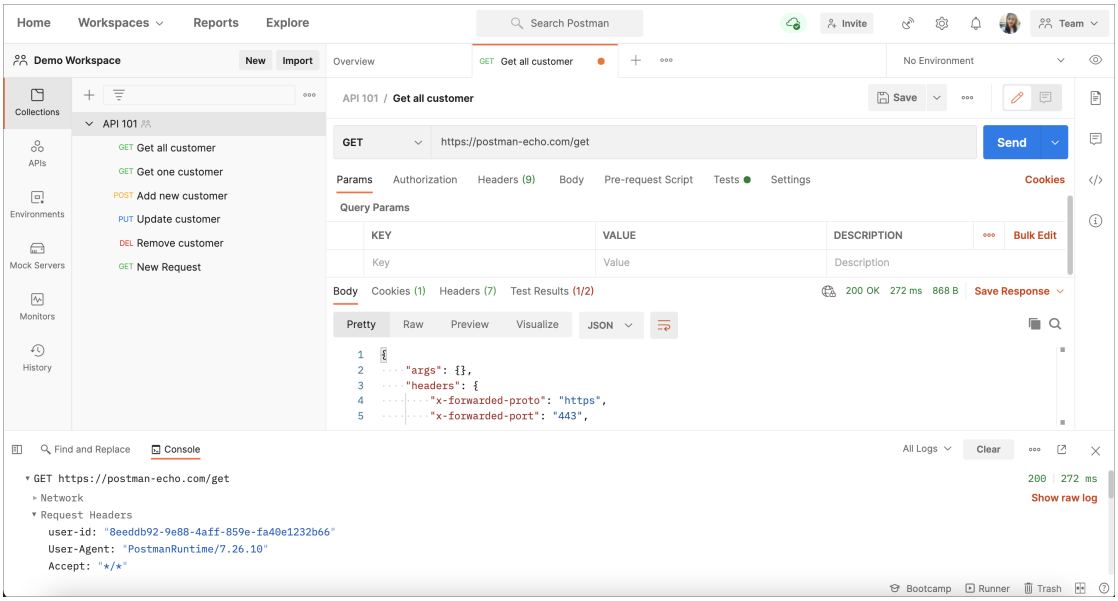


Figura 17: esempio dell'interfaccia di Postman nel quale si possono simulare diversi tipi di richieste.

fonte: postman.com

3.5 Database relazionali column e row oriented

Durante lo stage ho potuto utilizzare dei RDBMS[G] relazionali, in particolare Microsoft SQL Server e quello di SAP HANA. Ho potuto lavorare ed assistere ad alcune attività insieme a consulenti *senior* che gestivano tali *database*, formandomi facendo domande e documentandomi online, in particolare il mio utilizzo effettivo si è limitato a:

- segnalare requisiti di sviluppo relativo e discuterne assieme risorse del team dedicato;
- eseguire query tramite i servizi di backend[G] in particolare uno dei tanti dialetti SQL;

Le tabelle di un RDBMS[G] costituiscono un'astrazione dei dati salvati in memoria e l'operazione più costosa in termini di *performance* è il loro recupero; pertanto, i diversi approcci si concentrano sull'ottimizzare la memorizzazione per migliorare le prestazioni negli scenari d'interesse. Nei database colonnari o *column-oriented* le tabelle dei dati vengono memorizzate per colonne anziché per righe come invece accade nei database "tradizionali" o *row-oriented*.

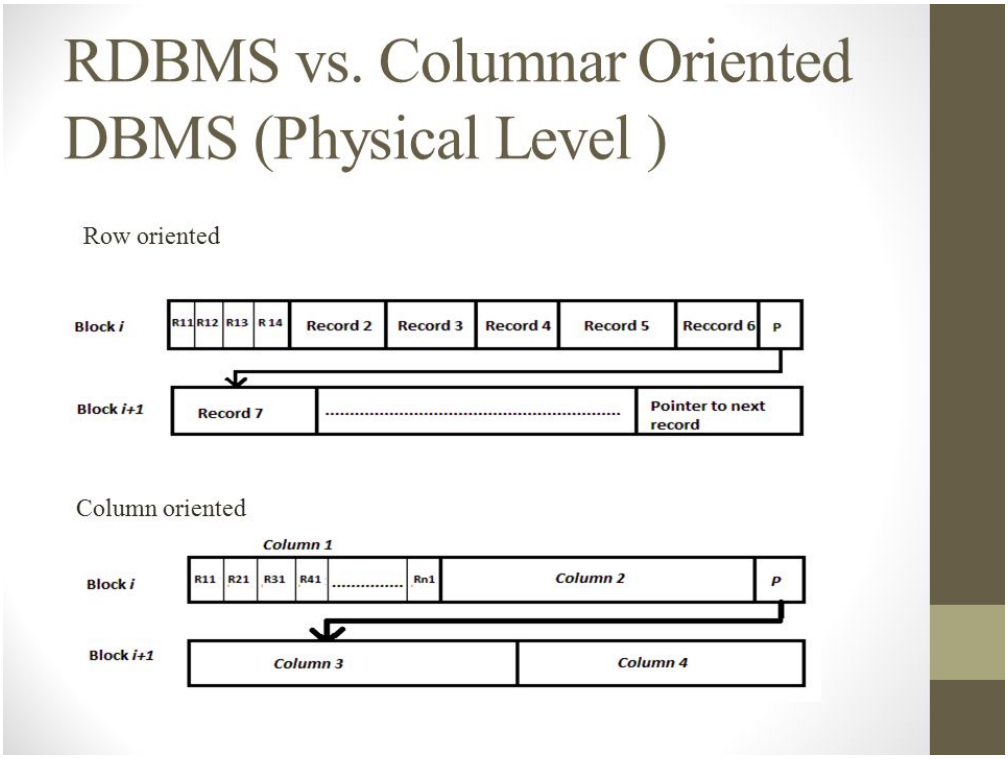


Figura 18: Rappresentazione visiva di come vengono memorizzati i dati nelle 2 tipologie di DBMS.

fonte: thecustomizewindows.com

3.5.1 Confrono row-oriented vs colomn oriented

All'uso pratico i RDBMS^G di una o dell'altra tipologia consentono le stesse funzionalità tramite linguaggi di tipo SQL, le differenze emergono analizzando le tipiche operazioni da eseguire sui dati memorizzati rendendo un approccio migliore dell'altro in base al caso d'uso.

Tabella 3: Principali differenze tra RDBMS *row-oriented* e *colomn-oriented*

Row-oriented	**Column-oriented**
I dati vengono archiviati e recuperati una "riga" alla volta e quindi potrebbero leggere dati non necessari se sono necessari alcuni dei dati di una riga.	I dati vengono memorizzati e recuperati in colonne (vedi immagine poco sopra) e quindi può leggere solo i dati necessari.
I record sono più facili(veloci) da scrivere e mentre lettura è tendenzialmente più lenta per grandi quantità di dati rispetto ai *column-oriented* oriented (dipende dal caso d'uso)	l'operazione di lettura di scrittura è più lenta rispetto a quelle *row-oriented* ma occupa meno spazio grazie alla compressione dei dati memorizzati, inoltre per grandi quantitativi la lettura è più veloce rispetto a un RDBMS *row-oriented*.
Non sono efficienti nell'esecuzione di operazioni applicabili all'intero set di dati (ex.WHERE,JOIN...) e quindi l'aggregazione orientata in riga è un tipo di operazione costosa.	Sono efficienti nell'esecuzione di operazioni(ex.WHERE,JOIN...) applicabili all'intero set di dati e quindi consentono l'aggregazione su molte righe e colonne con *performance* maggiori rispetto ai *row-oriented*.
Gli RDBMS^G *row-oriented* sono più adatti per il sistema di transazioni online(OLTP^G).	Gli RDBMS^G *colomn-oriented* sono più adatti per l'analisi dei dati online(OLAP^G).

3.5.2 Casi d'uso

Quindi i *database* che lavorano per righe sono più diffusi nei sistemi con esigenze di tipo OLTP^G in cui sono frequenti le richieste di uno o pochi record interi (come le pagine di dettaglio di un prodotto in un *e-commerce*) o in cui sono frequenti le operazioni localizzate di scrittura o aggiornamento (ad esempio nei sistemi di transazioni bancarie).

I database che lavorano per colonne sono invece più diffusi nei sistemi con necessità di tipo OLAP^G che tipicamente compiono interrogazioni complesse distribuite su una grande quantità di dati a scopi analitici (scopo principale dei data warehouse^G).

3.5.3 Ottimizzazione inserimento di record multipli

Una delle operazioni necessarie ad ottimizzare l'inserimento di *record* multipli nella stessa transazione è l'operazione chiamata **BULK INSERT**, tale operazione permette di:

- ridurre il numero di byte necessari all'*encoding* e tempo di *decoding* dei dati da trasmettere, perché già in partenza il tipo di dato da inserire è noto, grazie ad una tabella virtuale (o direttamente un file con le direttive necessarie al RDBMS^G, quindi solo i dati necessari da inserire vengono codificati, rispetto ad un operazione tramite stringa da codificare);
- inserire dati in modo massivo tramite un file o una tabella virtuale (dichiarando la struttura della tabella con i relativi nomi delle colonne e tipo relativo a ciascuna colonna), con incremento di performance notevole rispetto ad inserire n singoli record;
- riducendo il numero di connessioni necessarie (il tutto avviene generalmente in una singola connessione o in poche connessioni);
- si può disabilitare anche i controlli su chiavi e valori (ex. valori NULL) se necessario incrementando ulteriormente le performance.

Tale operazione era richiesta al soddisfamento dei requisiti di *performance*, nello sviluppo di alcuni endpoint^G delle API^G REST^G che sostanzialmente dovevano effetuare operazioni di inserimento o modifica. Come accennato nella sezione 3.4.2 tale ottimizzazione non è stata effettuata ma dovrebbe incrementeare notevolmente le performance come mostra l'immagine a seguire, rispetto ad altre tecniche in database relazionali.

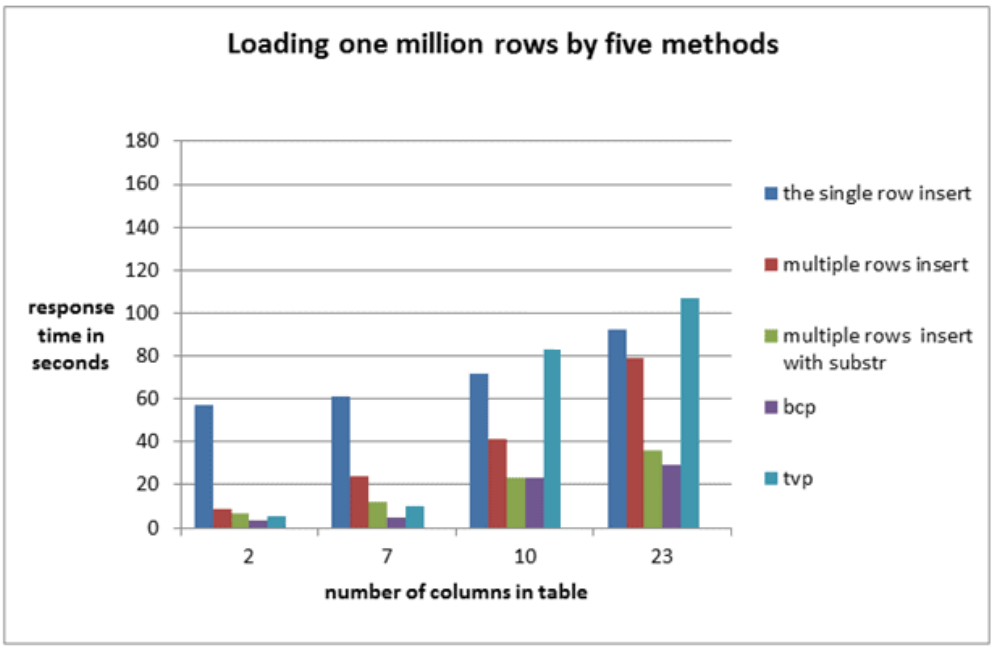


Figura 19: Comparazione tra tecniche di inserimento di righe multiple su un database relazionale (SQL Server 2008 R2- database row oriented) dove sia il server che il *client* sono all'interno della stessa macchina(quindi in condizioni "ottimali" per tempi di latenza.
fonte:red-gate.com

Come si può notare in questi test(immagine sopra) non è stato contato il tempo di trasmissione e quindi del *encoding* e *decoding* dei dati sulla rete, ma da un idea del perché sia necessaria come operazione in casi d'uso dove sono rilevanti le operazioni di inserimento di moltissimi *record* di dati(ex.IoT^G,Advanced Analytics^G...) dove oltre a dover leggere i dati in modo veloce, su necessità bisogna inserire molti *record* in tempi accettabili.

3.6 Altri strumenti di supporto allo sviluppo

Oltre ai strumenti aziendali già citati nel Capitolo 1 e gli IDE^G utilizzati nel frontend^G e backend^G, l'azienda usa diversi servizi cloud^G per la gestione di *repository* di codice, i principali che ho visto utilizzare in azienda sono:

- Github
- Azure Repos

Entrambi supportano appieno Git, un *software open source* di controllo versione distribuito, sviluppato da Linus Torvalds. Esso viene utilizzato sia da riga di comando, ma anche tramite *software* di terze parti che utilizzano un'interfaccia utente semplificandone

il suo utilizzo (Fork, Github Desktop, GitKraken, ...), anche Méthode fa uso di questi software tramite UI. Git, come tutti i *software* di controllo versione presenti sul mercato, basa il suo utilizzo sul concetto di *repository*, che può essere definito come un ambiente all'interno del quale vengono immagazzinati i metadati che possono essere recuperati ed aggiornati da chiunque ne abbia accesso. Infatti, il grande potenziale dei *software* di controllo versione sta nella possibilità di tenere traccia delle modifiche effettuate ad un insieme di file, dando la possibilità di ripristinare versioni precedenti e di aggiornare l'ultima versione tramite commit con un messaggio che introduce i cambiamenti effettuati. L'utilizzo di Git e di *repository* salvati in cloud^G è praticamente lo standard attuale se si vuole sviluppare *software*.

3.7 Riepilogo delle attività svolte

Alcune delle attività inizialmente non pianificate, sono frutto di un adattamento della pianificazione delle attività di frontend^G arrivata da esigenze di clienti e mi hanno permesso di conferire un valore aggiunto allo *stage*, incrementando lo *stack* di tecnologie utilizzato rispetto a quello previsto inizialmente.
Alla fine dello *stage* ho steso un breve documento dove ho documentato i punti rimasti in sospeso che ho condiviso con gli *stackholders* di interesse (il *tutor* e altri miei colleghi aziendali).
Segue un riepilogo delle attività svolte durante lo *stage* con un dettaglio superiore rispetto alla tabella degli obiettivi aziendali.

Tabella 4: Riepilogo attività svolte

Periodo	Tipologia attività	Grado di autonomia	Ore	Sommario attività	Descrizione
Settimana 1	Formazione generale	Bassa	8	Formazione strumenti aziendali di supporto	Introduzione ed installazione dei principali software necessari (sistema di ticketing^G, librerie, IDE^G...).
Settimana 1	Formazione generale	Alta	4	Formazione JavaScript	Studio tecnologie JavaScript individuale.

Tabella 4: Riepilogo attività svolte

Periodo	**Tipologia attività**	**Grado di autonomia**	**Ore**	**Sommario attività**	**Descrizione**
Settimana 1	Formazione frontend	Bassa	8	*intro* webapp(SPA^G)	Introduzione progetto webapp partecipando (in modo passivo) ad attività collaboartiva di pair programming^G con risorse *senior* vedendo in azione la libreria React.
Settimana 2	Formazione backend	Bassa	12	DevOps^G *deploy* Progetto AWS^G	Partecipazione a DevOps^G con simulazione pratica del *deploy* e discussione dei *pattern* di sviluppo utilizzati in questo progetto.

Tabella 4: Riepilogo attività svolte

Periodo	Tipologia attività	Grado di autonomia	Ore	Sommario attività	Descrizione
Settimana 2	Formazione frontend	Bassa	8	Formazione SAP Data Services,C#,Blazor	Formazione collaborativa insieme ad un programmatore con esperienza rilevante con queste tecnologie, con *focus* sul lato pratico di queste tecnologie tramite esempi su attività clienti finalizzata con attività di pair programming^{G}.
Settimana 2	Sviluppo frontend	Media	4	Partecipazione componenti frontend SAP Data Services,C#,Blazor	Affiancamento sviluppo componenti frontend, *controller* di una di queste componenti, in attività clienti.

Tabella 4: Riepilogo attività svolte

Periodo	Tipologia attività	Grado di autonomia	Ore	Sommario attività	Descrizione
Settimana 3	Documentazione	Alta	8	stesura documentazione progetto AWS^G	Stesura della documentazione relativa all'infrastruttura, *deploy* del progetto e del relativo backend^G e frontend del progetto basato su AWS^G, discusso in settimana 2.
Settimana 3	Sviluppo frontend	Media	4	Scrittura *unit test* di componenti frontend^G	scrittura codice per test di componenti frontend React utilizzando il linguaggio JavaScript progetto basato su AWS^G.
Settimana 3	Sviluppo frontend	Media	8	Sviluppo componenti frontend SAP Data Services,C#,Blazor	Sviluppo componenti frontend con il *framework* Blazor.

Tabella 4: Riepilogo attività svolte

Periodo	Tipologia attività	Grado di autonomia	Ore	Sommario attività	Descrizione
Settimana 4	Formazione backend	Bassa	12	Introduzione Infrastruttura Azure	Introduzione all'infrastruttura cloud^G di Azure, collaborazione tramite pair programming^G in attività clienti.
Settimana 4	Sviluppo frontend	Media	4	SAP Data Services,C#,Blazor	Sviluppo componenti frontend con il *framework* Blazor.
Settimana 5	Sviluppo backend	Bassa	16	Selezione e test librerie npm	*Setup* del *environment* di sviluppo e collaborazione in attività di *pair programmig* per lo sviluppo di servizi cloud^G sulla piattaforma Azure.
Settimana 5	Sviluppo frontend	Media	8	SAP Data Services,C#,Blazor	Sviluppo componenti frontend con il *framework* Blazor.

Tabella 4: Riepilogo attività svolte

Periodo	Tipologia attività	Grado di autonomia	Ore	Sommario attività	Descrizione
Settimana 5	Sviluppo backend	Media	16	Inizio sviluppo servizi cloud(API^G) con Azure	Inizio sviluppo API^G REST^G sulla piattaforma Azure, testando/selezionando le librerie da usare, partecipando ad attività di pair programming^G e sviluppando in autonomia i primi endpoint^G.
Settimana 6	Formazione Frontend	Alta	12	Formazione React e Redux	Formazione individuale, approfondendo in particolare, i concetti delle 2 librerie relativi allo *state management*.

Tabella 4: Riepilogo attività svolte

Periodo	Tipologia attività	Grado di autonomia	Ore	Sommario attività	Descrizione
Settimana 6	Formazione Frontend	Bassa	12	Formazione React e Redux	Formazione collaborativa in passaggio di consegna(DevOps^G) di un *template* di progetto con risorse *senior* in attività di pair programming^G.
Settimana 6	Sviluppo backend	Alta	12	Partecipazione API^G REST^G con Azure	Aggiunta nuovi endpoint^G e approfondimento documentazione librerie npm utilizzate notato problema di *encoding* dell'operazione di BULKINSERT relativo ad una delle librerie).
Settimana 6	Documentazione	Alta	4	Documentazione progetto basato in React	Stesura documentazione relativo al progetto dopo passaggio di consegna con *focus* principale sui *pattern* di sviluppo utilizzati.

Tabella 4: Riepilogo attività svolte

Periodo	Tipologia attività	Grado di autonomia	Ore	Sommario attività	Descrizione
Settimana 7	Formazione generale	Bassa	12	Formazione webapp in React	Formazione collaborativa in passaggio di consegna(DevOps^G) di un progetto con risorse senior in attività di pair programming^G.
Settimana 7	Documentazione	Alta	8	Documentazione *template* di progetto	Stesura documentazione relativo ad un *template* di progetto, citato sulla riga sopra.
Settimana 7	Sviluppo frontend	Alta	8	SAP Data Services,C#,Blazor	Sviluppo componenti frontend con il framework Blazor.
Settimana 7	Sviluppo frontend	Media	12	Sviluppo componenti frontend webapp(SPA^G)	Partecipazione ad attività clienti (webapp SPA^G) in stesura codice per la creazione di componenti frontend utilizzando la libreria React.

Tabella 4: Riepilogo attività svolte

Periodo	Tipologia attività	Grado di autonomia	Ore	Sommario attività	Descrizione
Settimana 8	Sviluppo frontend	Media	16	Sviluppo componenti frontend	Partecipazione ad attività clienti in stesura codice per la creazione di componenti frontend utilizzando la libreria React e Redux.
Settimana 8	Sviluppo frontend	Alta	8	SAP Data Services,C#,Blazor	Sviluppo componenti frontend con il *framework* SAP Data Services e Blazor.
Settimana 8	Sviluppo backend	Alta	16	Sviluppo servizi cloud^G su Azure	Partecipazione e *test* dei servizi cloud^G per il data entry^G (scomposizione dati).
Settimana 9	Sviluppo frontend	Alta	16	Sviluppo componenti frontend webapp(SPA^G)	Partecipazione ad attività clienti in stesura codice per la creazione di componenti frontend in modo individuale con le librerie React e Redux.

Tabella 4: Riepilogo attività svolte

Periodo	**Tipologia attività**	**Grado di autonomia**	**Ore**	**Sommario attività**	**Descrizione**
Settimana 9	Sviluppo backend	Alta	24	Sviluppo API[G] REST[G] su Azure	*Testing* degli [G] sviluppati ed eventualmente relativo *fix* dei *bug* trovati.
Settimana 10	Sviluppo frontend	Alta	16	Sviluppo componenti frontend webapp(SPA[G])	Partecipazione ad attività clienti in stesura codice per la creazione di componenti frontend in modo individuale con le librerie React e Redux.
Settimana 10	Sviluppo backend	Alta	24	Sviluppo API[G] REST[G] su Azure	Sviluppo e *testing* di altri endpoint[G].

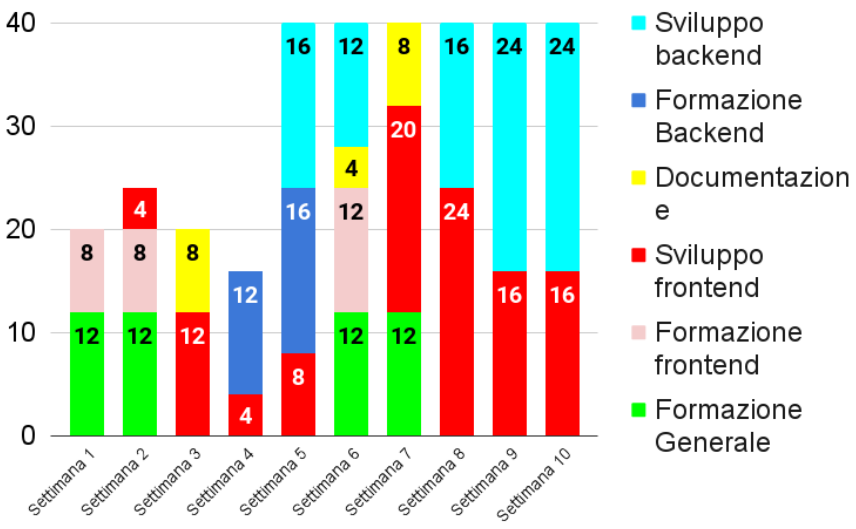


Figura 20: Distribuzione oraria effettiva delle attività durante il periodo di stage

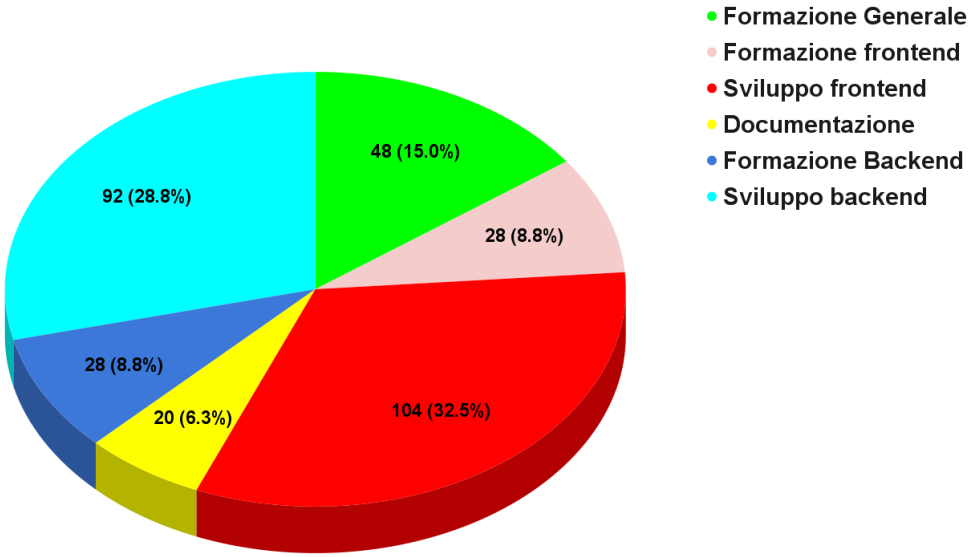
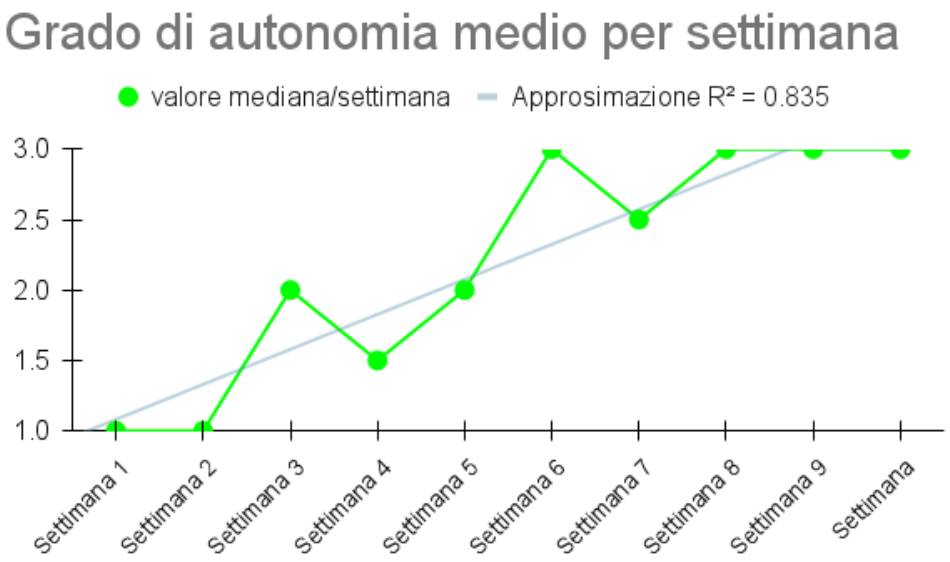


Figura 21: Aerogramma relativo all'investimento orario (delle 320 ore complessive) su ciascuna tipologia di attività



1: Bassa 2: Media

3: Alta

Figura 22: Grafico che stima il grado di autonomia che **secondo me** ho raggiunto durante le attività svolte (tabella riepilogo attività svolte) al variare della settimana, come si può notare dalla settimana 5 in poi ho mantenuto un grado di autonomia sopra la media.

Capitolo 4

Valutazione retrospettiva

4.1 Riepilogo obbiettivi

Come riportato in sezione 2.3, all'inizio dell'attività di *stage*, insieme al *tutor* abbiamo discusso il piano di lavoro per delineare gli obiettivi da raggiungere nelle 320 ore previste. Gli obiettivi obbligatori riguardano fondamentalmente lo studio e l'utilizzo supervisionato di tecnologie impiegate in azienda. Gli obiettivi desiderabili invece consistono nell'utilizzare tali tecnologie in casi d'uso concreti e con autonomia. Mentre gli obiettivi aggiuntivi che consistevano in attività clienti nate da opportunità clienti non inizialmente previste (ex. SAP Data Services... vedi tabella obiettivi fissati dall'azienda).
Ho cercato di raggiungere gli obiettivi formandomi nel primo periodo (e su mia necessità) su quanto in uso in azienda ed utilizzando le conoscenze acquisite nello sviluppo di codice, affiancando risorse *senior* in attività clienti per poi svolgerle in autonomia.

4.1.1 Valutazione personale

Confido di essere riuscito a soddisfare la totalità degli obiettivi eccetto per alcune attività (connesse ad obbiettivi desiderabili) non completate:

- **backend:** l'ottimizzazione dei tempi di risposta da parte di alcuni endpoint^G relativi alle API^G REST^G (che necessitano della BULK INSERT vedi sezione 3.5.3)
- **frontend:** una componente grafica abbastanza complessa, nella quale mancava una parte relativa all'esperienza utente (UX), non conclusa del tutto per problemi di schedulazione temporale.

Valutando l'insieme, penso di aver contribuito (nei miei limiti) ed aver raggiunto un buon grado di autonomia nello svolgere le attività assegnatemi durante lo stage(vedi grafico autonomia), motivo per cui mi sento molto soddisfatto.

4.1.2 Valutazione aziendale:

Man mano che mi venivano assegnate i *task*, ricevevo un *feedback* verbale da parte del *tutor* sulle attività svolte, generalmente con frequenza settimanale: tale *feedback* era basato sulla valutazione in prima persona del *tutor* e valorizzato da eventuali risorse del *team* di sviluppo o con cui collaboravo. Generalmente ho ricevuto *feedback* positivi eccetto per la frequenza di utilizzo di alcune *best practices*.

4.2 Conoscenze acquisite

Prima dello *stage* non avevo idea di cosa fosse la Business Intelligence^G, la mia permanenza durante questo periodo a Méthode mi ha permesso di scoprire in modo limitato anche

questo ambito, scoprendo la realtà associata e lo *stack* di tecnologie impiegate in tali contesti.
Inizialmente ho incontrato difficoltà nel comprendere alcune conoscenze sull'infrastruttura e l'uso di alcune librerie ma poi mediante studio individuale e partecipando a diverse attività collaborative con risorse più esperte, son riuscito a superarle.
Segue una breve trattazione delle conoscenze che ritengo di aver ampliato o aggiunto al mio bagaglio formativo, suddivisi per tematica nelle sottosezioni seguenti (4.2.x).

4.2.1 Database

Anche se avevo già delle conoscenze sui RDBMS^G e sul loro utilizzo, durante lo stage ho potuto espandere la mia cultura relativa ai *database* colonnari, che non possedevo inizialmente. Ho avuto la possibilità di vedere le dimensioni, la complessità e la necessità delle basi di dati utilizzate nel mondo produttivo; abituato alle limitate situazioni di studio non avevo idea di quanto complessi potessero essere.

4.2.2 Tecnologie Frontend

Adottando tali tecnologie in casi d'uso reali ho potuto constatare e vedere sia la qualità che la quantità di codice che può richiedere.
Oltre all'opportunità di approfondire il linguaggio JavaScript e le librerie frontend^G come React, che in ottica professionale è attualmente molto richiesto per lo sviluppo di applicazioni web. Oltre a ciò, ho potuto utilizzare ed apprendere una metodologia di lavoro a me poco nota come Scrum^G, un tipo di metodo organizzativo delle attività da svolgere della metodologia Agile^G.

4.2.3 Servizi cloud

Mi ritengo soddisfatto anche delle conoscenze acquisite in ambito cloud^G. Essendomi stata concessa la possibilità di sviluppare delle API^G REST^G e delle funzioni in cloud^G per la trasmissione e trasformazione di dati ho potuto incrementare le mie competenze in tale ambito, in particolare lo sviluppo di API^G è sempre più rilevante in contesti professionali.Nello sviluppare tali servizi ho potuto quindi apprendere anche diverse *best practices* che non conoscevo oltre allo stack di tecnologie coinvolte.

4.2.4 Best practices

Penso di aver messo in pratica diverse best practices durante il mio *stage*, che ritengo molto utili per il futuro, avendone apprezzato il valore aggiunto dalla loro messa in pratica. Tra di esse le principali sono:

- comunicare in modo proattivo tramite documentazione (organizzazione lavorativa, tipologie di comunicazione);
- metodologie di lavoro Agile^G;

- l'utilizzo di specifici strumenti (Postman,IDE[G],estensioni...), linguaggi(TypeScript), *filesystem*(YAML) che possono aiutare gli sviluppatori (specialmente in determinati contesti), sia nella stesura di codice(ex. *autocomplete*), che ad evitare potenziali bug, errori, incompatibilità. In conclusione, permettono di salvare molto tempo aumentando sia l'efficacia che l'efficienza dello sviluppatore nelle sue mansioni.

4.3 Università vs lavoro

4.3.1 Realtà aziendale

L'azienda presso cui ho effettuato lo *stage* si occupata di Business Intelligence[G], tema che non viene trattato nel corso degli studi da me intrapreso, in particolare ho trovato difficoltà nell'apprendere il modello di *business* dei clienti di riferimento necessario per molte delle attività svolte dall'azienda. Quindi nel periodo trascorso in azienda ho avuto la possibilità di assistere e lavorare in una realtà lavorativa diversa ma in un *team* affine ai miei studi. Discutendo con i colleghi e confrontandomi con i responsabili ho percepito un clima altamente professionale, dove si ha la possibilità di crescere non solo dal punto di vista professionale ma anche umano, visto il clima collaborativo che c'è in azienda. Ho potuto apprezzare l'impatto che possono avere l'utilizzo di *best practices* in un contesto professionale. Inoltre, mi sono fatto un'idea più chiara, di cosa sia l'attività di sviluppo a livello professionale: l'approccio, le difficoltà, le metodologie ed i relativi vincoli (ex. scadenze temporali).

4.3.2 Università

Devo dire che il percorso di studi in generale mi ha fornito delle basi solide da cui partire per sviluppare le competenze richieste. L'uso di tecnologie a me non note, mi ha chiesto un impegno minore di quanto mi immaginassi, probabilmente dovuto al fatto di aver svolto diversi progetti con diverse tecnologie durante il corso di studi e nel tempo libero, oltre alla disponibilità del personale con grande esperienza hanno aiutato a ridurre sicuramente i tempi. L'unico consiglio che mi sento di dare per il corso di studi, è di utilizzare tecnologie più vicine al mercato, in particolare nei progetti didattici.

4.3.3 Conclusione

Sono felice di avere avuto una buona autonomia durante buona parte del periodo dello stage ed allo stesso tempo, ho apprezzato molto la disponibilità e l'esperienza dei colleghi pronti ad aiutare nell'evenienza. In conclusione, mi ritengo molto soddisfatto dello stage svolto.

Bibliografia

Riferimenti bibliografici

Siti web consultati

[1] JavaScript: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous

[2] Typescript: https://www.typescriptlang.org/

[3] React:

- Documentazione: https://it.reactjs.org/;
- Tutorial: https://www.youtube.com/watch?v=JPT3bFIwJYA&list=PL55RiY5tL51oyA8euSROLjMFZbXaV7skS.

[4] Redux:

- Documentazione: https://redux.js.org/;
- Tutorial: https://egghead.io/courses/fundamentals-of-redux-course-from-dan-abramov-bd5cc867;
- Dev Tools: https://chrome.google.com/webstore/detail/redux-devtools/lmhkpmbekcpmknklioeibfkpmmfibljd?hl=it.

[5] Documentazione Azure:

- Cloud Functions: https://docs.microsoft.com/en-us/azure/azure-functions/functions-reference-node?tabs=v2; esempio di casi d'uso: https://docs.microsoft.com/en-us/azure/azure-functions/functions-overview;
- data warehousing: https://azure.microsoft.com/it-it/services/synapse-analytics/#overview;
- Node.Js: https://docs.microsoft.com/it-it/sql/connect/node-js/step-3-proof-of-concept-connecting-to-sql-using-node-js?view=sql-server-ver15

[6] Open API documentazione REST API: https://swagger.io/specification/.

[7] pacchetti npm:

- https://www.npmjs.com/package/openapi-backend
- https://www.npmjs.com/package/mssql
- https://www.npmjs.com/package/tedious

- https://www.npmjs.com/package/adm-zip

[8] Material-UI: https://v4.mui.com/.

[9] JSON,YAML: https://www.geeksforgeeks.org/what-is-the-difference-between-yaml-and-json/

[10] Best practices test API REST: https://nordicapis.com/9-best-practices-for-rest-api-testing/.

[11] Best practices JavaScript: https://www.w3schools.com/js/js_best_practices.asp.

[12] Databases:

- https://www.geeksforgeeks.org/difference-between-row-oriented-and-column-oriented-data-stores-in-dbms/
- https://www.linkedin.com/pulse/columnar-vs-row-oriented-databases-basics-2-min-read-faraz-khan/
- https://www.red-gate.com/simple-talk/databases/sql-server/performance-sql-server/comparing-multiple-rows-insert-vs-single-row-insert-with-three-data-load-methods/
- https://docs.microsoft.com/it-it/sql/relational-databases/indexes/columnstore-indexes-data-loading-guidance?view=sql-server-ver15

[13] SCRUM: https://scrumguides.org/index.html

Glossario

Advanced Analytics

è l'esame autonomo o semi-autonomo di dati o contenuti utilizzando tecniche e strumenti sofisticati, in genere al di là di quelli della tradizionale business intelligence (BI), per scoprire informazioni più approfondite, fare previsioni o generare raccomandazioni Cit. a pag. 1, 40

Agile

In ingegneria del software, è un insieme di metodi di sviluppo del software emersi a partire dai primi anni 2000 e fondati su insieme di principi comuni, direttamente o indirettamente derivati dai principi del Manifesto per lo sviluppo agile del software. Tale manifesto si può riassumere in quattro punti:

1. le persone e le interazioni sono più importanti dei processi e degli strumenti;
2. è più importante avere software funzionante che documentazione;
3. bisogna collaborare con i clienti oltre che rispettare il contratto;
4. bisogna essere pronti a rispondere ai cambiamenti oltre che aderire alla pianificazione.

Un esempio di metodo di sviluppo di tipo agile è il metodo Scrum^{G}. Cit. a pag. 21, 30, 54, 60, 62

API

API è l'acronimo di Application Programming Interface, un *software* intermediario che consente a due applicazioni di dialogare tra loro. Ogni volta che utilizzi un'app come Facebook, invii un messaggio istantaneo o controlli il meteo sul telefono, stai utilizzando un'API(video esplicativo: https://www.mulesoft.com/resources/api/what-is-an-api). Cit. a pag. I, VII, 12, 17, 29, 31, 33, 35, 39, 46, 47, 50, 53, 54, 62

AWS

Amazon Web Services, Inc. (nota con la sigla AWS) è un'azienda statunitense di proprietà del gruppo Amazon, che fornisce servizi di *cloud computing* su un'omonima piattaforma su richiesta. Questi servizi sono operativi in diverse regioni geografiche in cui Amazon stessa ha suddiviso il globo. Cit. a pag. 15, 17, 29, 42, 44

backend

Nell'architettura del software possono esserci molti livelli tra l'hardware e l'utente finale. Questa parte non è eplicitamente visibile all'utente, che in generale utilizza un software per interagire con questa parte posteriore(*backend*), che di solito gestisce l'archiviazione dei dati e la logica di *business*.). Cit. a pag. I, 7, 12, 17, 19, 20, 21, 29, 37, 40, 44, 60

Business Intelligence

Ci si riferisce sia all'insieme dei processi aziendali destinati a raccogliere ed analizzare informazioni, sia alle tecnologie e agli strumenti impiegate nel processo; la finalità ultima è quella di analizzare e poi accedere ai risultati per migliorare decisioni e prestazioni aziendali. Cit. a pag. VII, 1, 2, 3, 11, 12, 13, 26, 33, 53, 55, 59, 60

cloud

l'insieme delle risorse hardware o software presente in server remoti e distribuito in rete, contenente i dati e i programmi di un utente (ex. applicazioni cloud dove salvare file di vario genere come immagini...) Cit. a pag. VII, 4, 9, 12, 13, 17, 18, 20, 29, 30, 31, 32, 33, 35, 40, 41, 45, 49, 54

dashboard

L'idea di base di un dashboard (cruscotto) è di fornire viste a colpo d'occhio di alcuni argomenti, come lo stato di un particolare gruppo o organizzazione, una vista consolidata dei dati distribuiti, ecc. La quantità di dettagli in una dashboard varia considerevolmente a seconda allo scopo e del contesto del cruscotto. Cit. a pag. VII, 3, 7, 29, 34

data entry

In informatica *Data entry*, letteralmente inserimento di dati, è l'operazione di immissione di dati in un computer o in un generico sistema informatico. Tale operazione solitamente viene eseguita da un individuo mediante digitazione su una tastiera, ma può essere anche automatizzata mediante, ad esempio, tecnologie di lettura ottica o riconoscimento del parlato; semi-automatizzata mediante autocompletamento, la creazione di *template* e l'utilizzo di file *Comma-separated values* (CSV) per l'importazione e esportazione di dati fra programmi differenti. Cit. a pag. 13, 18, 29, 31, 49

data warehouse

In informatica, con data warehouse (DW o DWH), noto anche come Enterprise Data Warehouse (EDW), si intende un sistema utilizzato per il *reporting* e l'analisi dei dati ed è considerato un componente fondamentale della Business Intelligence^{G}. I DW sono archivi centrali di dati integrati provenienti da una o più fonti disparate. Memorizzano i dati attuali e storici in un'unica posizione che vengono utilizzati per la creazione di report analitici per i lavoratori in tutta l'azienda.. 13, 29, 39, 61

data quality

consiste nella pianificazione, implementazione e controllo delle attività che applicano tecniche di gestione della qualità dei dati, al fine di garantire che siano adatti allo scopo e soddisfino le esigenze degli utilizzatori. Cit. a pag. 4, 6

data integration

si riferisce ai processi da attuare su dati provenienti da diverse sorgenti informative per fornire all'utente una visione unificata di quei dati. Cit. a pag. 4, 6

data discovery

La *Data* discovery comporta la raccolta e la valutazione dei dati da varie fonti ed è spesso usata per capire le tendenze e i modelli nei dati. Richiede una progressione di passi che le aziende possono usare come struttura per capire i loro dati. La Data Discovery, di solito associata a Business Intelligence^G (BI), aiuta a informare le decisioni aziendali riunendo fonti di dati disparate e a compartimenti stagni per essere analizzate. Cit. a pag. 3

DevOps

è un insieme di pratiche che combina lo sviluppo software (Dev) e le operazioni IT (Ops). L'obiettivo è abbreviare il ciclo di vita dello sviluppo dei sistemi e fornire una fornitura continua con un'elevata qualità del software. DevOps è complementare allo sviluppo del software Agile^G; diversi aspetti DevOps derivano dalla metodologia Agile^G. Cit. a pag. 19, 24, 29, 42, 47, 48

endpoint

Un endpoint è il "punto di connessione" di un servizio, uno strumento o un'applicazione a cui si accede tramite una rete. Nel mondo del *software*, qualsiasi applicazione *software* in esecuzione ed in attesa di connessioni utilizza un *endpoint* come "porta d'ingresso". Cit. a pag. 31, 32, 33, 39, 46, 47, 50, 53

ETL

In informatica Extract, Transform, Load (ETL) è un'espressione in lingua inglese che si riferisce al processo di estrazione, trasformazione e caricamento dei dati in un sistema di sintesi (data warehouse, data mart, big data, ...). I dati vengono estratti da sistemi sorgenti quali database transazionali (OLTP), comuni file di testo o da altri sistemi informatici (ad esempio, sistemi ERP o CRM). Cit. a pag. 6, 8, 26

frontend

Nell'architettura del software possono esserci molti livelli tra l'hardware e l'utente finale, generalmente si intende la parte di software visibile all'utente e con cui egli può interagire —tipicamente un'interfaccia utente (UI). Cit. a pag. I, 7, 8, 12, 13, 17, 19, 20, 21, 22, 24, 28, 40, 41, 44, 54, 60

fullstack

generalmente identifica la tipologia di sviluppo software, ovvero, sia *client* che *server*. Uno sviluppatore full stack è una persona che può sviluppare software sia frontend^G che backend^G. Cit. a pag. I, 12, 13, 15

Great Place to Work

è una certificazione data ad un'azienda il che certifica se l'ambiente in cui i dipendenti credono nelle persone per cui lavorano, sono orgogliosi di quello che fanno, e stanno bene con le persone con cui lavorano. Cit. a pag. 1

IDE

Un ambiente di sviluppo integrato (IDE) è un'applicazione software che fornisce servizi completi ai programmatori di computer per lo sviluppo del software. Un IDE normalmente consiste almeno in un editor di codice sorgente, strumenti di automazione della build e un debugger. Alcuni IDE, come NetBeans ed Eclipse, contengono il compilatore, l'interprete o entrambi necessari altri invece no. Cit. a pag. 28, 33, 40, 41, 55, 63

IoT

Acronimo di Internet of things, con il quale si riferisce all'estensione di Internet al mondo degli oggetti e dei luoghi concreti.Per "cosa" o "oggetto" si può intendere più precisamente categorie quali: dispositivi, apparecchiature, impianti e sistemi, materiali e prodotti tangibili, opere e beni, macchine e attrezzature. Questi oggetti connessi che sono alla base dell'Internet delle cose si definiscono più propriamente smart object (in italiano oggetti intelligenti) e si contraddistinguono per alcune proprietà o funzionalità. Introdotto da Kevin Ashton, cofondatore e direttore esecutivo di Auto-ID Center (consorzio di ricerca con sede al MIT), durante una presentazione presso Procter Gamble nel 1999, il concetto fu in seguito sviluppato dall'agenzia di ricerca Gartner. Cit. a pag. I, VII, 4, 10, 12, 13, 17, 29, 30, 40

OLAP

acronimo dell'espressione On-Line Analytical Processing, designa un insieme di tecniche software per l'analisi interattiva e veloce di grandi quantità di dati, che è possibile esaminare in modalità piuttosto complesse. Questa è la componente tecnologica base del data warehouse^G e, ad esempio, serve alle aziende per analizzare i risultati delle vendite, l'andamento dei costi di acquisto merci, al marketing per misurare il successo di una campagna pubblicitaria, a una università per organizzare i dati di un sondaggio ed altri casi simili. Cit. a pag. 38, 39

OLTP

L'online transaction processing (OLTP) è un insieme di tecniche software utilizzate per la gestione di applicazioni orientate alle transazioni. Non esiste una definizione precisa e generale di transazione anche se in gran parte dei casi ci si riferisce alle transazioni nell'ambito dei database relazionali e alle relative metodologie e concetti che consentono lettura e scrittura concorrenti garantendo atomicità, coerenza, isolamento e non volatilità dei dati. Cit. a pag. 38

pair programming

è la programmazione a coppie, una tecnica Agile^G di sviluppo software in cui due programmatori lavorano insieme su una *workstation*. Uno, il conducente, scrive il codice mentre l'altro, l'osservatore o il navigatore, rivede ogni riga di codice mentre viene digitata. I due programmatori si scambiano frequentemente i ruoli. Durante la revisione, l'osservatore considera anche la direzione "strategica" del lavoro, proponendo idee per miglioramenti e probabili problemi futuri da affrontare. Questo ha lo scopo di liberare il conducente di concentrare tutta la sua attenzione sugli aspetti "tattici" del completamento dell'attività corrente, utilizzando l'osservatore come rete di sicurezza e guida. Cit. a pag. 19, 24, 29, 30, 42, 43, 45, 46, 47, 48

RDBMS

indica un database management system basato sul modello relazionale, introdotto da Edgar F. Codd. Oltre a questi, anche se meno diffusi a livello commerciale, esistono sistemi di gestione di basi di dati che implementano modelli dei dati alternativi a quello relazionale: gerarchico, reticolare e a oggetti. Cit. a pag. 37, 38, 39, 54

real-time

in informatica, un sistema real-time (in italiano "sistema in tempo reale") è un calcolatore in cui la correttezza del risultato delle sue computazioni dipende non solo dalla correttezza logica ma anche dalla correttezza temporale. Quest'ultima è spesso espressa come tempo massimo di risposta. Cit. a pag. 4, 10

report

In informatica, prospetto riepilogativo di una raccolta di dati disposti in colonna o in altre forme, ottenuto con il calcolatore Cit. a pag. 3, 5, 6, 7

REST

Accronimo di *Representational State Transfer*, è un tipo di architettura software per i sistemi distribuiti, basata su HTTP per la trasmissione; il funzionamento prevede una struttura degli URL ben definita (atta a identificare univocamente una risorsa o un insieme di risorse) e l'utilizzo dei verbi HTTP specifici per il recupero di informazioni (GET), per la modifica (POST, PUT, PATCH, DELETE) e per altri scopi (OPTIONS, ecc.).REST è quindi un insieme di linee guida applicabili quando necessario, che rende le API^G REST più rapide, leggere e con una maggiore scalabilità (rispetto ad altre tipologie). Cit. a pag. I, VII, 12, 17, 29, 31, 33, 35, 39, 46, 47, 50, 53, 54

runtime system

in informatica, un sistema di *runtime*, chiamato anche ambiente di runtime, implementa principalmente parti di un modello di esecuzione. Questo non deve

essere confuso con la fase del ciclo di vita di un programma, durante la quale il sistema di *runtime* è in funzione. Quando si tratta il sistema di runtime come distinto dall'ambiente di runtime (RTE), il primo può essere definito come una parte specifica del software applicativo (IDE^G) utilizzato per la programmazione, un pezzo di software che fornisce al programmatore un ambiente più conveniente per l'esecuzione dei programmi durante la loro produzione (*testing* e simili) mentre il secondo (RTE) sarebbe l'istanza stessa di un modello di esecuzione applicato al programma sviluppato che viene poi eseguito a sua volta nel suddetto sistema di *runtime*. La maggior parte dei linguaggi di programmazione ha una qualche forma di sistema di *runtime* che fornisce un ambiente in cui vengono eseguiti i programmi. Questo ambiente può affrontare una serie di problemi, tra cui la gestione della memoria dell'applicazione, il modo in cui il programma accede alle variabili, i meccanismi per il passaggio dei parametri tra le procedure, l'interfaccia con il sistema operativo e altro. Cit. a pag. 8, 20, 21

Scrum

Metodo organizzativo generalmente utilizzato nello sviluppo software che rientra fra i metodi Agile. Prevede di dividere il progetto in blocchi rapidi di lavoro (Sprint), ciascuno dei quali composto da diverse *user stories* ordinate in base all'importanza data dal *Product Owner*, alla fine di ciascuno dei quali l'obiettivo è creare un incremento del software che aggiunge valore al progetto. Esso indica come definire i dettagli del lavoro da fare nell'immediato futuro e prevede vari meeting con caratteristiche precise per creare occasioni di ispezione e controllo del lavoro svolto. Cit. a pag. 21, 54, 58

SPA

Un'applicazione a pagina singola (SPA) è un'applicazione Web o un sito Web che interagisce con l'utente riscrivendo dinamicamente la pagina Web corrente con nuovi dati dal server Web, invece del metodo predefinito di un *browser* Web che carica intere nuove pagine. L'obiettivo sono transizioni più rapide che rendano il sito Web più simile a un'app nativa. In una SPA, non si verifica mai un aggiornamento della pagina; invece, tutto il codice HTML, JavaScript e CSS necessario viene recuperato dal *browser* con un singolo caricamento della pagina oppure le risorse appropriate vengono caricate dinamicamente e aggiunte alla pagina secondo necessità, di solito in risposta alle azioni dell'utente. Cit. a pag. I, VII, 21, 22, 23, 25, 42, 48, 49, 50

TCO

è una stima finanziaria destinata ad aiutare acquirenti e proprietari a determinare i costi diretti e indiretti di un prodotto o servizio. È un concetto di contabilità di gestione che può essere utilizzato nella contabilità analitica completa o anche nell'economia ecologica dove include i costi sociali. Cit. a pag. 4

ticketing

In questo caso si intende la generazione di uno o più trouble ticket, ovvero una richiesta di assistenza, tracciata da un sistema informatico di gestione delle richieste di assistenza, che per metonimia viene indicato con lo stesso termine. In inglese questi sistemi vengono anche indicati con i termini: issue tracking system (ITS), trouble ticket system, support ticket, request management o incident ticket system. Cit. a pag. 9, 41