Politecnico di Torino

POLITECNICO DI TORINO

INGEGNERIA INFORMATICA
A.A 2024/2025
SESSIONE DI LAUREA MARZO 2025

Application Performance Monitoring e estrazione di dati per il business

RELATORI:
PROF. TORCHIANO MARCO

CANDIDATO:
TORTORE LUCA

Marzo 2025

Indice

Elenco delle figure

1 Introduzione

Spesso, quando furono scritti i primi software, questi erano progettati per scopi specifici, utilizzando hardware dedicato. Per questo motivo l'interazione e lo scambio di dati con altri programmi erano molto limitati.
L'introduzione delle reti di calcolatori e dei protocolli di comunicazione ha reso possibile l'interazione tra programmi, portando alla creazione di sistemi distribuiti. Questa novità portava a un incremento delle capacità computazionali di un sistema, ma allo stesso tempo, a un aumento della complessità nella gestione di quest'ultimo.
Con l'ascesa delle architetture basate sui servizi e del Cloud, ci fu un ulteriore ed estremo aumento della complessità del sistema software. Le infrastrutture software moderne sono caratterizzate da un'elevata complessità e interconnessione, i loro componenti devono interagire tra loro scambiandosi dati in tempo reale in modo fluido per garantire una buona esperienza utente.
In questo scenario, l'application performance monitoring (APM) è diventato un componente essenziale per il funzionamento di un applicativo software, l'APM si occupa del monitoraggio e analisi in tempo reale del funzionamento di un'applicazione, individuando inefficienze o problemi che possono causare un'esperienza utente negativa e impattando sull'utilizzo.
L'integrazione di strumenti di APM con tecnologie avanzate come l'intelligenza artificiale e l'analisi predittiva permette alle aziende di anticipare problemi, mi-

gliorare continuamente i propri servizi e mantenere un vantaggio competitivo nel mercato.

1.1 Contesto aziendale

Alten è un'azienda di consulenza internazionale con una specializzazione nell'innovazione tecnologica e nell'ingegneria. Fondata in Francia nel 1988, Alten opera in diversi settori, tra cui tecnologia dell'informazione, elettronica, sviluppo di software embedded, reti di telecomunicazioni e ingegneria.
In Italia, Alten ha stabilito una presenza significativa dal 2004, con un totale di 12 sedi. Le principali aree di competenza di Alten Italia comprendono: consulenza ICT1, consulenza tecnologica, soluzioni, telecomunicazioni e scienze della vita. Il lavoro è stato svolto all'interno del team di Application Performance interface & Application Performance Monitoring (API&APM), dove sono stato affiancato dalle mie tutor Elena Mellano e Laura Bando, che mi hanno guidato durante la realizzazione del progetto, lasciandomi comunque un margine di autonomia e garantendomi supporto quando necessario.

1.2 Obiettivi del progetto

L'azienda presso cui mi è stato proposto questo progetto mi ha ospitato all'interno del team APM, permettendomi così di comprendere l'importanza di questa pratica, spesso poco conosciuta al di fuori dell'ambito strettamente tecnico.
Nella pratica dell'APM, vengono impiegati strumenti software di monitoraggio che non si limitano solo al monitoraggio delle prestazioni del singolo programma, ma vengono integrati con sistemi che consentono l'analisi delle prestazioni sull'intera infrastruttura includendo gli host e la rete.
Uno dei strumenti principali utilizzati dall'azienda è Dynatrace, una piattaforma leader nel settore dell'APM per le infrastrutture, capace di fornire una visione completa delle performance attraverso funzionalità avanzate di monitoraggio, analisi delle dipendenze e rilevamento automatico delle anomalie. Dynatrace è uno strumento molto potente, capace di monitorare e analizzare una rete con numerosi componenti diversi tra loro, creando in automatico delle interfacce semplificate in modo da rendere comprensibili le analisi anche a personale non tecnico.

Queste sue importanti capacità, sono accoppiate a problemi in fatto di flessibilità a esigenze esterne. Un cliente dell'azienda potrebbe avere necessità di monitorare e trattare un tipo di dato particolare, in questo caso il funzionamento di Dynatrace non risulta intuitivo e rende alcune volte è addirittura impossibile eseguire l'operazione.
Il progetto propone quindi di sviluppare una serie di script che automatizzino l'estrazione di dati rilevati da Dynatrace, con l'obiettivo di:

- **Supportare l'analisi delle performance delle applicazioni**: fornire dati strutturati che consentano all'azienda di identificare rapidamente aree critiche o opportunità di miglioramento.
- **Facilitare la reportistica**: automatizzare il recupero delle informazioni chiave per generare report utili a clienti e team tecnici.
- **Abilitare la correlazione tra dati tecnici e metriche di business**: estrarre informazioni che possano essere utilizzate per analisi strategiche, come l'impatto delle prestazioni sull'esperienza cliente.

Questo progetto si pone come obiettivo principale dimostrare come l'utilizzo delle API di Dynatrace e lo sviluppo di script automatizzati possano trasformare un'attività tecnica come il monitoraggio delle prestazioni in una risorsa strategica per il business. L'elaborato approfondirà ogni fase del progetto, dall'identificazione delle esigenze aziendali, alla validazione degli script sviluppati, sottolineando i benefici ottenuti e le possibili evoluzioni future, con un approfondimento legato allo studio teorico degli strumenti utilizzati
Ho avuto l'opportunità di lavorare su dati interni all'azienda coperti da privacy, per questo motivo tutte le immagini faranno riferimento alla piattaforma demo di Dynatrace, ma saranno comunque utili ai fini della relazione.

1.3 Struttura della tesi

L'elaborato si suddivide nei seguenti capitoli:

- **Capitolo 1**: capitolo introduttivo e considerazioni generali del progetto.
- **Capitolo 2**: definizione e spiegazione dettagliata del concetto di APM, del suo utilizzo nelle aziende e degli strumenti attualmente disponibili sul mercato.

- **Capitolo 3**: caso di studio di Dynatrace, con spiegazione della sua struttura del suo funzionamento e del utilizzo dell'intelligenza artificiale Davis nel ambito dell'APM.
- **Capitolo 4**: descrizione in dettaglio della parte tecnica del progetto, della progettazione e stesura degli script, mostrando alcuni dei risultati prodotti.
- **Capitolo 5**: analisi sui risultati del mio lavoro del possibile impatto sul business aziendale.
- **Capitolo 6**: capitolo finale dell'elaborato dove sono inserite alcune delle mie considerazioni sul progetto.

2

Application Performance Monitoring

Nel mercato digitale odierno, a causa dell'estrema concorrenza, garantire che un' applicazione operi sempre al massimo delle prestazioni è essenziale. Le app non devono soltanto svolgere un servizio corretto durante ogni momento della giornata, ma anche fornire risposte rapide e risoluzione dei problemi in tempo reale. La connessione tra l'esperienza dell'utente e i numerosi servizi di back-end che supportano le sue funzioni non è sempre evidente, in particolare nel caso di applicazioni cloud distribuite. Ciò comporta una maggiore complessità strutturale che molte volte risulta in un incremento delle difficoltà nel controllo e nella gestione del software, per questo motivo si utilizza l'AMP.

2.1 Definizione e importanza dell'APM

L'Application Performance Monitoring (APM) è un termine che racchiude un insieme di strumenti e tecnologie che hanno lo scopo di monitorare e gestire le performance e la disponibilità di un software, nonché l'esperienza utente.
L'utilizzo dell'APM consente di raccogliere dati in tempo reale da diverse metriche al fine di controllare lo stato del sistema ed eventualmente segnalare e correggere problemi che potrebbero comportare un'esperienza utente negativa.

L'APM fornisce una visione chiara di come gli utenti sperimentano le applicazioni e di dove si trovano i gap di prestazioni. Applicazioni mobili, siti web e software aziendali sono alcuni esempi di interfacce front-end che l'APM può monitorare, fornendo approfondimenti sull'esperienza utente. Inoltre, l'APM include elementi di supporto come host, processi, servizi, rete e log per offrire una comprensione più approfondita delle prestazioni delle applicazioni.
Uno strumento APM consente agli sviluppatori di monitorare e risolvere i problemi all'interno dell'ambiente IT, garantendo un'esperienza utente affidabile e fluida per i consumatori finali.
La società di ricerca Gartner definisce l'APM come:

> «Application performance monitoring is a suite of monitoring software comprising digital experience monitoring (DEM), application discovery, tracing and diagnostics, and purpose-built artificial intelligence for IT operations.»

2.2 Misurare le prestazioni del software

Poichè in generale la definizione dell'APM può essere riassunta con il concetto di monitoraggio delle prestazioni di un programma, risulta quindi necessario definire gli strumenti con i quali diventa possibile effettuare questa misurazione: i KPI.
Gli Indicatori di Performance Chiave (Key Performance Indicators, KPI) rappresentano un elemento essenziale per il monitoraggio delle performance di un software. I KPI sono essenzialmente delle metriche quantificabili utilizzate per valutare l'efficacia e l'efficienza di un sistema.
Nel contesto dell'APM, i KPI permettono di identificare problemi di performance e supportano anche il processo decisionale aziendale attraverso dati concreti e tempestivi.
In un ambiente digitale in cui l'esperienza utente e la disponibilità del servizio rappresentano fattori fondamentali per il successo di un'applicazione, i KPI agiscono come punti di riferimento per valutare lo stato di salute dell'infrastruttura digitale.
L'identificazione e il monitoraggio dei KPI corretti consente di effettuare una corretta analisi ai fini dell'APM.
I KPI sono utilizzati per misurare virtualmente qualsiasi cosa, ai fini dell'APM, gli indicatori che ci interessano sono:

- KPI Tecnici: questi indicatori si concentrano sulla performance tecnica dell'applicazione e includono metriche come il tempo di risposta, la latenza, l'utilizzo delle risorse e la percentuale di errori.

- KPI Utente: misurano la qualità dell'esperienza utente attraverso metriche come la soddisfazione dell'utente, i tempi di caricamento delle pagine e i tassi di abbandono.

- KPI di Business: collegano le performance tecniche agli obiettivi strategici dell'azienda, includendo metriche come la conversion rate, il valore medio per transazione e l'impatto economico delle interruzioni.

Un altro aspetto importante nella definizione dei KPI è il collegamento tra le metriche e gli obiettivi aziendali. Ad esempio, un tempo di risposta ridotto non è importante di per sé, ma è uno strumento per aumentare la soddisfazione degli utenti e, di conseguenza, migliorare le prestazioni o ridurre i costi di supporto.
Il processo di traduzione delle metriche in insight utili per l'azienda richiede una stretta collaborazione tra team IT e stakeholders aziendali.

2.3 Componenti principali di un sistema APM

L'architettura di un sistema APM è organizzata in tre livelli principali per garantirne la flessibilità e scalabilità.
Il primo livello rappresenta la raccolta dati, si occupa del monitoraggio e dell'estrazione delle informazioni dalle risorse IT, tramite agenti e raccoglitori. Il secondo livello utilizza moduli di analisi per aggregare e processare i dati, trasformandoli in insight utili. Infine, il terzo livello offre un'interfaccia utente attraverso dashboard, report e sistemi di alert, consentendo una visione chiara e intuitiva delle performance.
Questa struttura modulare permette di adattare il sistema alle esigenze specifiche delle aziende.
Nelle successive sezioni si spiegheranno quali sono i moduli principali di un sistema APM e come interagiscono tra loro.

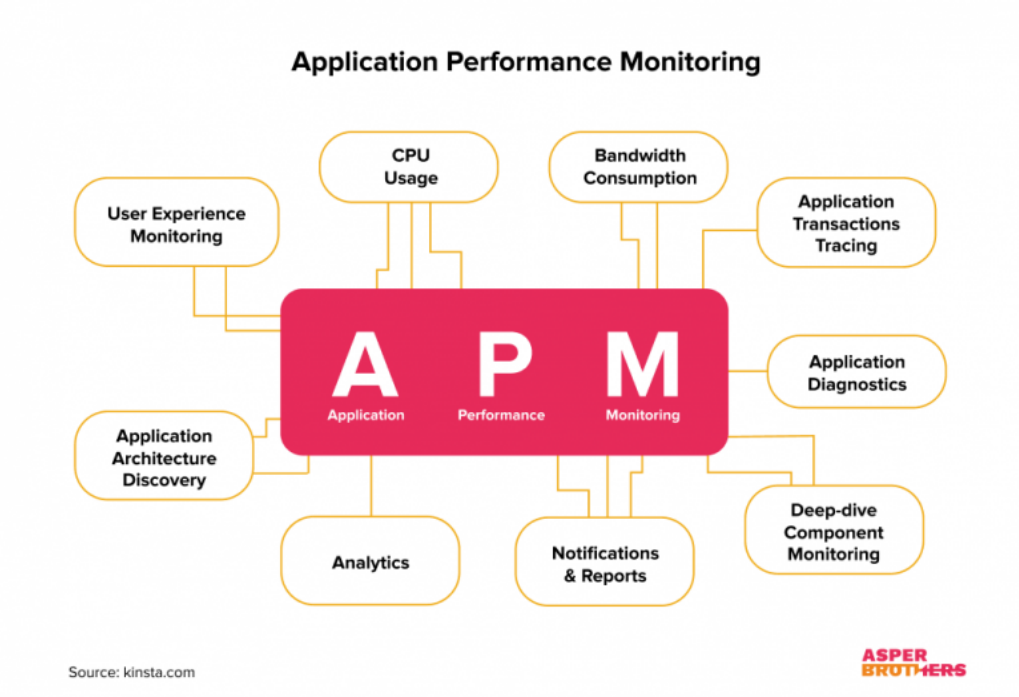

Figura 2.1: tutte le componenti di APM

2.3.1 Agenti di monitoraggio

Gli agenti di monitoraggio rappresentano uno dei pilastri di un sistema APM. Essi sono componenti software progettati per raccogliere dati in tempo reale relativi all'esecuzione delle applicazioni e delle risorse sottostanti.
Installati su server, container o dispositivi endpoint, gli agenti operano come intermediari tra il sistema monitorato e l'infrastruttura APM, fornendo informazioni dettagliate su metriche, eventi e performance.
Gli agenti di monitoraggio possono essere classificati in diverse categorie in base alla loro funzione e al livello di integrazione:

- **Agenti di sistema**: monitorano il sistema operativo e le risorse hardware utilizzate da un host. Alcuni esempi possono essere l'utilizzo della CPU e i processi attualmente in esecuzione.

- **Agenti applicativi**: monitorano la performance delle applicazioni su un host. Utilizzano degli strumenti per rilevare i tempi di risposta o eventuali errori dell'applicazione.

- **Agenti di rete**: si occupano di monitorare lo stato della rete che connette i diversi host di una infrastruttura, misurano la latenza e il throughtput, inoltre fornisco una mappatura della rete.

- **Agenti specializzati**: vengono creati appositamente per poter misurare elementi specifici come i database.

Una volta installati (nel gergo tecnico "deploy"), gli agenti iniziano a raccogliere ed inviare i dati riguardanti le KPI d'interesse al sistema APM. Inoltre, eseguono una pre-elaborazione in modo tale da non inondare lo strumento con dati inutili o ricorrenti e inviano solo i dati ritenuti rilevanti.
È quindi necessario fare delle considerazioni sull'utilizzo di questi strumenti; Nonostante la pre-elaborazione, l'installazione di troppi agenti può portare a un concreto impatto sulle performance, poichè esiste il rischio che il flusso di dati inviati al sistema centrale di APM possa causare rallentamenti. Inoltre bisogna considerare eventuali problemi di compatibilità e sicurezza.

2.3.2 Raccoglitori di dati

I raccoglitori di dati servono da intermediari tra gli agenti di monitoraggio, che raccolgono i dati in tempo reale, e il sistema centrale di APM. La loro funzione principale è garantire che i dati vengano acquisiti, organizzati e trasmessi in modo efficiente, sicuro e coerente. I raccoglitori di dati si occupano di collezionare tutte le informazioni spedite sulla rete dai diversi agenti di monitoraggio. In questo modo, possono effettuare un'aggregazione dei dati e creare un unico formato standard da spedire al sistema centrale di APM.
A seconda da dove sono posizionati all'interno dell'infrastruttura, possono essere di due tipi diversi:

- **Raccoglitori centralizzati**: questa categoria di raccoglitori riceve dati da tutti gli agenti presenti e di conseguenza aggregano i dati in unico punto. Questa configurazione consente una minore complessità.

- **Raccoglitori distribuiti**: questa tipologia è adatta alle infrastrutture decentralizzate, ricevono dati solo dagli agenti di una sottorete e devono coordinarsi con i raccoglitori di altre sottoreti per poter mandare i dati al sistema centrale. Questo assetto comporta un aumento della complessità di implementazione, ma aumenta anche la scalabilità e l'efficenza.

2.3.3 Moduli di analisi

I moduli di analisi si occupano di creare insight d'interesse utilizzando le informazioni ricevute dai raccoglitori di dati. Questi moduli hanno il compito di inter-

pretare i dati grezzi collezionati trasformandoli in informazioni contestualizzate e strutturate in modo da fornire un significato utile.
L'introduzione dell'intelligenza artificiale, avvenuta negli ultimi anni, consente un'analisi dei dati più rapida, facilitando l'identificazione sia delle anomalie che delle cause dei problemi. Inoltre, utilizzando funzioni di autoapprendimento, l'intelligenza artificiale può effettuare delle previsioni su problematiche future e dare suggerimenti in modo tale da evitarle.
I moduli di analisi hanno un ruolo importante nel contesto dell'APM, svolgendo diverse funzioni fondamentali. Analizzano metriche come tempi di risposta, throughput, utilizzo delle risorse e tassi di errore per valutare le prestazioni e la salute del sistema. Grazie a sofisticati algoritmi, rilevano comportamenti anomali del software confrontando i dati con pattern storici o soglie predefinite.
Inoltre, correlano eventi apparentemente indipendenti per identificare relazioni di causa-effetto tra i componenti, contribuendo ad una comprensione più profonda delle dinamiche del sistema. Attraverso tecniche di Root Cause Analysis (RCA), individuano rapidamente le cause principali dei problemi, migliorando la risoluzione degli incidenti.
In aggiunta, basandosi sull'analisi di trend storici, forniscono previsioni sulle performance future e suggeriscono ottimizzazioni per prevenire potenziali criticità.
I moduli di analisi sono progettati con un'architettura modulare per offrire flessibilità e scalabilità. Tra i componenti principali, troviamo il motore di analisi, che si occupa dell'elaborazione di metriche e log per generare insight utili e il database analitico che serve da archivio ottimizzato per query ad alte prestazioni, permettendo la conservazione e l'accesso rapido ai dati elaborati.
Un altro elemento chiave è **l'interfaccia API**, che facilita l'integrazione con strumenti esterni, come dashboard, sistemi di notifiche e piattaforme di automazione.
Infine, la pipeline di dati gestisce il flusso continuo di informazioni provenienti dai raccoglitori, assicurando l'elaborazione sia in tempo reale che in modalità batch.
Esistono vari tipi di analisi che utilizzano diverse tecniche per effettuare l' elaborazione dei dati:

- **Analisi statistica**: utilizza tecniche matematiche come il calcolo delle medie, deviazioni o individuazione di trend. Generalmente questo tipo di analisi non fornisce nel dettaglio indicazioni utili, ma serve per dare un contesto generale alla performance.

- **Analisi utilizzando Machine Learning**: usando algoritmi avanzati per l' intelligenza artificiale, l'analisi è in grado di adattarsi facilmente a nuovi

scenari. Questo approccio consente di individuare anche le più piccole problematiche che i metodi matematici tradizionali potrebbero non rilevare. Inoltre, utilizzando lo storico dei dati, l'analisi rende possibile fare previsioni su situazioni futuri e di conseguenza può fornire suggerimenti ai fini di evitare problemi.

- **Root Cause Analysis (RCA)**: l'RCA è un approccio che mira ad identificare l'origine del problema, piuttosto che limitarne gli effetti, utilizzando funzioni di riconoscimento delle dipendenze effettuando quindi una correlazione tra eventi e metriche.

- **Analisi in tempo reale**: utilizza il monitoraggio in tempo reale per identificare problemi critici e correggerli nel più breve tempo possibile. Questo tipo di analisi si concentra più sull'efficacia delle azioni a breve termine, non riconosce però problematiche nel lungo periodo.

Per ognuna di queste tipologie di analisi ci sono degli aspetti positivi e negativi, in un buon sistema APM si utilizzano delle combinazioni di questi approcci in modo tale da coprire le lacune di ogni tipologia di analisi.

2.3.4 Dashboard e strumenti di visualizzazione

Le dashboard forniscono un modo semplice ed efficace per rappresentare i dati calcolati durante la fase di analisi. Questi strumenti consentono di visualizzare in modo chiaro e immediato lo stato delle applicazioni, evidenziando metriche chiave, anomalie e performance in tempo reale.
Le dashboard sono un modo intuitivo per informare il personale non tecnico dello stato di salute di un'infrastruttura software, consentendo anche a chi non ne conosce il funzionamento di prendere decisioni aziendali per supportare il miglioramento del proprio software.

Le dashboard di un sistema APM forniscono strumenti versatili per il monitoraggio e l'analisi delle applicazioni; consentono la visualizzazione delle metriche chiave, come tempi di risposta, throughput, tassi di errore e utilizzo delle risorse, offrendo un quadro completo delle performance. Il monitoraggio in tempo reale permette di seguire istante per istante l'andamento del sistema, mentre la funzionalità di drill-down consente di approfondire i dati, analizzando specifiche transazioni o componenti.

Le dashboard integrano anche le funzionalità di alerting e notifiche, segnalando visivamente eventuali anomalie o il superamento di soglie critiche. La personalizzazione consente agli utenti di adattare layout, widget e filtri alle proprie esigenze, migliorando l'efficienza operativa. Inoltre, esiste una funzione di creazione dei report, in modo tale da poterli condividere con i clienti.
Per i team IT e DevOps, le dashboard offrono una visione dettagliata delle metriche tecniche, includendo funzionalità come la visualizzazione in tempo reale delle performance, l'analisi dei log e delle transazioni, e il monitoraggio dello stato di salute dei servizi e delle infrastrutture sottostanti.
Per il management e gli stakeholder aziendali, le dashboard strategiche forniscono una panoramica delle performance applicative in relazione agli obiettivi di business. Queste includono metriche di alto livello, come SLA e KPI aziendali, indicatori sull'esperienza utente, come il Net Promoter Score e i tassi di conversione, oltre a informazioni sui costi e sull'efficienza delle risorse IT.

2.4 API per accesso ai dati

Le Application Programming Interface (API) sono componenti chiave nei sistemi di APM, consentendo l'interazione tra il sistema APM e altre applicazioni o strumenti esterni. Grazie alle API, è possibile automatizzare processi, personalizzare il monitoraggio e integrare i dati raccolti con altre piattaforme IT o di business intelligence.
In un contesto APM, le API rappresentano un ponte che permette ai team tecnici di accedere ai dati in tempo reale, configurare il sistema e ottimizzare le operazioni senza dover dipendere esclusivamente dalle interfacce grafiche offerte dalla piattaforma. In questo modo si ottiene una personalizzazione della raccolta e trattamento dei dati.

Le API in un sistema APM offrono diverse funzionalità per il monitoraggio, la configurazione e l'integrazione con altri strumenti:

- **API di monitoraggio**: consentono di accedere alle metriche raccolte dagli agenti e dai raccoglitori, permettendo il recupero dei dati relativi alle performance di applicazioni, infrastrutture e servizi. Inoltre, forniscono accesso ai log per ottenere informazioni dettagliate sugli eventi registrati e supportano il monitoraggio in tempo reale attraverso flussi di dati continui.

- **API di configurazione**: permettono agli utenti di configurare il sistema APM in modo programmatico. Includono la gestione delle soglie per impostare limiti alle metriche chiave e attivare alert, la creazione e modifica automatizzata di dashboard personalizzate, e la gestione degli agenti, come registrazione, aggiornamenti o rimozione.
- **API di integrazione**: facilitano la comunicazione tra il sistema APM e altre piattaforme. Possono essere utilizzate per integrarsi con strumenti DevOps, come Jenkins, Kubernetes o Docker, per esportare dati verso piattaforme di business intelligence come Tableau, Power BI o Grafana, e per collegarsi a sistemi di alerting come Slack, PagerDuty o Microsoft Teams.

2.5 Benefici dell'APM per il business

Un sistema di APM è un asset fondamentale per una azienda moderna, con le sue funzioni migliora le prestazioni strategiche e operative. In un mercato altamente competitivo, come quello dei sistemi software, risulta cruciale garantire un'ottima esperienza utente per fidelizzare i propri clienti.
Grazie alle funzionalità di monitoraggio e analisi in tempo reale, l'APM permette di ottimizzare i tempi di risposta delle applicazioni, prevenire interruzioni attraverso una rapida rilevazione e risoluzione dei problemi e personalizzare i servizi sulla base dei dati degli utenti. Questo si traduce in clienti più soddisfatti, maggiore fidelizzazione e un incremento delle raccomandazioni positive, rafforzando la reputazione aziendale.

Dal punto di vista operativo, l'APM aiuta a ridurre i costi e migliorare l'efficienza attraverso la prevenzione proattiva dei downtime, che possono causare perdite significative in termini di fatturato e produttività. Inoltre, l'ottimizzazione delle risorse IT consente di ridurre sprechi, pianificare investimenti mirati e garantire un utilizzo efficiente delle infrastrutture. L'APM favorisce anche una migliore collaborazione tra team IT, sviluppo e business, grazie a una visibilità condivisa sui dati applicativi e ai flussi di lavoro ottimizzati.
Dal punto di vista finanziario, l'APM contribuisce alla riduzione dei costi operativi grazie all'automazione dei processi, minimizzando la necessità di interventi manuali, e grazie all'analisi dei trend di utilizzo delle risorse, evitando spese inutili o migliorie infrastrutturali non necessarie. Inoltre, l'APM aiuta a preservare il valore aziendale, minimizzando i danni reputazionali causati da interruzioni di servizio prolungate.

L'APM è anche uno strumento strategico per supportare la scalabilità del business. In momenti di crescita o trasformazione digitale, consente di monitorare nuove applicazioni fin dal loro lancio, garantendone prestazioni ottimali. La gestione dei picchi di traffico, come durante eventi promozionali o campagne di marketing, viene facilitata grazie alla visibilità in tempo reale e all'ottimizzazione delle risorse. La capacità di adattarsi a infrastrutture complesse, come ambienti multicloud o architetture basate su microservizi, rende l'APM un alleato indispensabile per sostenere l'innovazione tecnologica.
Infine, grazie alla raccolta e all'analisi dei dati, l'APM offre insight preziosi che guidano decisioni strategiche. I dati storici sulle performance permettono di identificare trend critici e opportunità di miglioramento, facilitando l'allineamento tra i reparti IT e business. Questo crea un linguaggio comune basato su metriche comprensibili e utili per entrambe le parti. Monitorando l'impatto delle iniziative IT sugli obiettivi aziendali, è quindi possibile misurare il ritorno sugli investimenti tecnologici e adattare le strategie future in modo più consapevole.
In un mercato in rapida evoluzione, l'adozione di un sistema APM diventa un vantaggio competitivo. La capacità di individuare e risolvere rapidamente i problemi consente di accelerare il time-to-market per nuove funzionalità e prodotti. La garanzia di un'esperienza utente di qualità rafforza la fiducia nel brand, mentre il supporto all'innovazione continua permette di esplorare nuove tecnologie e approcci senza compromettere la stabilità operativa. In sintesi, un sistema APM non è solo uno strumento di monitoraggio, ma una leva strategica per migliorare le performance aziendali, ridurre i costi e sostenere la crescita e l'innovazione.

3

Dynatrace: un sistema completo

3.1 Il report Gartner

Gartner rappresenta una delle aziende di maggior importanza nel panorama globale della tecnologia, distinguendosi per la sua competenza in consulenza strategica, analisi di mercato e ricerca applicata all'informatica. L'azienda è ampiamente riconosciuta come una delle fonti più autorevoli e affidabili per informazioni strategiche relative all'Information Technology (IT).
Il ruolo principale di Gartner consiste nel consigliare i propri clienti nelle decisioni di investimento tecnologico. L'azienda fornisce un'ampia gamma di servizi, tra cui attività di ricerca approfondita, consulenza strategica, benchmarking, organizzazione di eventi di settore e fornitura di notizie e approfondimenti.
Uno degli strumenti più distintivi sviluppati da Gartner è il cosiddetto Quadrante Magico (Magic Quadrant, MQ).
Questo strumento di analisi comparativa offre una valutazione obiettiva e rigorosa delle principali piattaforme e soluzioni tecnologiche disponibili sul mercato, consentendo di identificare i leader, i visionari, i player di nicchia e i challenger in specifici ambiti tecnologici. Tale metodologia si basa su parametri standardizzati e su una rigorosa raccolta di dati, garantendo una prospettiva imparziale e ben informata.

Grazie alla sua consolidata reputazione e alla metodologia analitica, Gartner svolge un ruolo cruciale nell'orientare le scelte strategiche delle organizzazioni. I rapporti e le analisi fornite dall'azienda permettono di ottenere una visione chiara delle tendenze tecnologiche emergenti, delle opportunità di mercato e delle sfide competitive. Ciò consente ai manager aziendali di adottare strategie di investimento informate, coerenti e orientate al lungo termine, minimizzando i rischi e massimizzando il ritorno sugli investimenti.
In sintesi, Gartner non si limita a essere un punto di riferimento per l'analisi di mercato, ma rappresenta un partner strategico per la trasformazione digitale e per l'innovazione tecnologica, contribuendo in maniera significativa al progresso del settore IT su scala globale.

3.1.1 Il quadrante magico di Gartner

Il Quadrante Magico di Gartner rappresenta uno degli strumenti analitici più autorevoli e utilizzati nel campo dell'IT e dei mercati tecnologici.
La rappresentazione del Magic Quadrant è organizzata graficamente in una matrice a due dimensioni, che classifica i fornitori di tecnologia in quattro categorie principali:

- **Leader**: aziende che si distinguono per la capacità di esecuzione e per una visione complessiva altamente innovativa. Questa è considerata la categoria più ambita, in quanto identifica i fornitori che combinano un'offerta tecnologica avanzata e fornitori che hanno una grande quota di mercato, operando con infrastruttura estesa e fornendo servizi eccellenti.

- **Visionari**: aziende che uniscono una comprensione avanzata delle tendenze future e una visione innovativa, ma non hanno ancora una quota di mercato sufficente a radicare una realtà ed a fornire un servizio efficente.

- **Giocatori di nicchia (Niche Players)**: fornitori che si concentrano su mercati specifici o ambiti limitati, offrendo soluzioni mirate ma meno adatte a un pubblico ampio o a esigenze globali.

- **Sfidanti (Challengers)**: attori che posseggono un'infrastruttura estesa ed efficente, ma mancano di innovazione. I Challengers rappresentano la concorrenza principale ai Leaders.

Questo schema grafico consente una facile comprensione dello stato delle aziende in un mercato in crescita rapida, fornendo una rappresentazione intuitiva del

posizionamento relativo dei diversi fornitori.
Il Magic Quadrant si basa su una serie uniforme di criteri di valutazione, che vengono applicati in modo trasparente e sistematico. Tra questi criteri figurano la capacità di esecuzione, che valuta aspetti come la qualità dei prodotti, il supporto ai clienti, la forza organizzativa, e la completezza della visione, che considera la capacità di innovare, la conoscenza del mercato e la coerenza strategica.
Grazie alla combinazione di analisi rigorosa e rappresentazione grafica, il Magic Quadrant si configura come uno strumento decisionale di fondamentale importanza per le organizzazioni. Esso consente ai responsabili aziendali e agli investitori di comprendere rapidamente quanto efficacemente i fornitori di tecnologia stiano implementando la propria visione strategica e quanto questa si avvicini alla visione di mercato delineata da Gartner. In questo modo, il MQ supporta decisioni strategiche informate, contribuendo a identificare le soluzioni più adatte per affrontare le sfide tecnologiche attuali e future.
In conclusione, il Quadrante Magico non è soltanto un mezzo di classificazione dei fornitori, ma si presenta come un punto di riferimento essenziale per le decisioni strategiche delle aziende nel contesto delle rapide trasformazioni tecnologiche globali.

Figura 3.1: Quadrante Magico Application Performance Monitoring e Osservabilità 2023

3.2 Introduzione e filosofia di Dynatrace

Dynatrace è una società tecnologica di rilevanza globale che si distingue per l'offerta di una piattaforma avanzata di APM, progettata per supportare le organizzazioni nella gestione delle complessità dei moderni ambienti IT. La piattaforma di Dynatrace si caratterizza per l'integrazione di tecnologie basate sull' intelligenza artificiale (AI) e sull'automazione, rendendola uno strumento all'avanguardia nel monitoraggio e nell'ottimizzazione delle prestazioni delle applicazioni.
La qualità e l'efficacia della soluzione proposta da Dynatrace sono state riconosciute da Gartner, che ha posizionato l'azienda come Leader nel Quadrante Magico per il settore APM. Questo riconoscimento riflette la capacità della piattaforma di soddisfare le esigenze di osservabilità dei moderni ecosistemi IT, caratterizzati da ambienti distribuiti, dinamici e altamente complessi.
Alten Italia descrive in un articolo dedicato, come la piattaforma di Dynatrace rappresenti una soluzione ottimale per soddisfare le esigenze dei propri clienti. Citando direttamente: "Pur conoscendo e utilizzando diverse piattaforme di APM, la nostra esperienza ci porta a indicare questa soluzione come quella che meglio risponde alle esigenze dei nostri Clienti." Questo riconoscimento pratico testimonia non solo l'affidabilità della piattaforma, ma anche la sua capacità di adattarsi alle sfide specifiche dei contesti aziendali.

3.3 Architettura della piattaforma

Nelle sezioni che seguono, verrà fornita un'analisi approfondita delle funzionalità offerte dalla piattaforma Dynatrace, attraverso l'esplorazione di un ambiente demo (per via della privacy aziendale non è possibile inserire dati che potrebbero risultare sensibili).
Questa analisi consentirà di comprendere nel dettaglio il funzionamento del sistema, illustrando i principali strumenti disponibili, il loro contributo nell'ambito del monitoraggio delle prestazioni applicative e dell'ottimizzazione dell' infrastruttura IT.
L'obiettivo è fornire una visione chiara e concreta dell'utilità di Dynatrace, con particolare attenzione ai vantaggi derivanti dall'utilizzo di AI e automazione per garantire una gestione proattiva delle prestazioni applicative e delle risorse tecnologiche.

3.4 OneAgent: il cuore del monitoraggio

OneAgent è il componente centrale responsabile della raccolta di tutti i dati di monitoraggio all'interno dell'ambiente supervisionato. Questo agente software richiede l'installazione di una singola istanza per ciascun host da monitorare, indipendentemente dal fatto che gli host siano distribuiti in container Docker, in architetture basate su microservizi o in infrastrutture cloud.
Grazie alla sua architettura avanzata, una sola istanza di OneAgent è in grado di gestire il monitoraggio di una vasta gamma di entità, tra cui server, applicazioni, servizi, database e molto altro. OneAgent offre un insieme completo di metriche relative alle prestazioni operative e aziendali, coprendo l'intero apparato tecnologico, dal frontend al backend. Questo include anche istanze cloud, host, stato della rete, processi e servizi.
Un elemento distintivo di OneAgent è la capacità di rilevare automaticamente tutti i processi attivi sugli host monitorati. Sulla base di questa analisi, OneAgent attiva automaticamente la strumentazione più adatta per lo specifico stack applicativo in uso.
Inoltre, OneAgent inietta dinamicamente i tag necessari per il monitoraggio dell' esperienza utente (user-experience monitoring) direttamente nel codice HTML delle pagine delle applicazioni. Questa funzionalità consente di integrare nuovi componenti e di gestirli automaticamente in tempo reale, senza necessità di intervento manuale.
Dal punto di vista tecnologico, OneAgent è composto da una serie di moduli software appositamente progettati per supportare un'ampia gamma di tecnologie già al momento dell'installazione (out-of-the-box). Ciascun modulo è specificatamente ottimizzato per interagire con determinate piattaforme, tecnologie o ambienti applicativi.
Per identificare quali moduli siano supportati per ogni piattaforma, è disponibile una matrice di supporto delle piattaforme e delle capacità di OneAgent. Allo stesso modo, è possibile consultare un elenco delle versioni supportate per ogni modulo, utile per garantire la compatibilità con le tecnologie utilizzate nell'ambiente monitorato.
In sintesi, OneAgent rappresenta uno strumento chiave per ottenere una visibilità completa e dettagliata dell'intero ecosistema IT, garantendo un monitoraggio efficace e l'ottimizzazione delle risorse attraverso automazione e intelligenza integrata.
Grazie alla sua flessibilità e alla capacità di adattarsi dinamicamente agli ambienti più complessi, OneAgent si conferma come una soluzione fondamentale per il

monitoraggio delle prestazioni applicative e delle infrastrutture moderne.

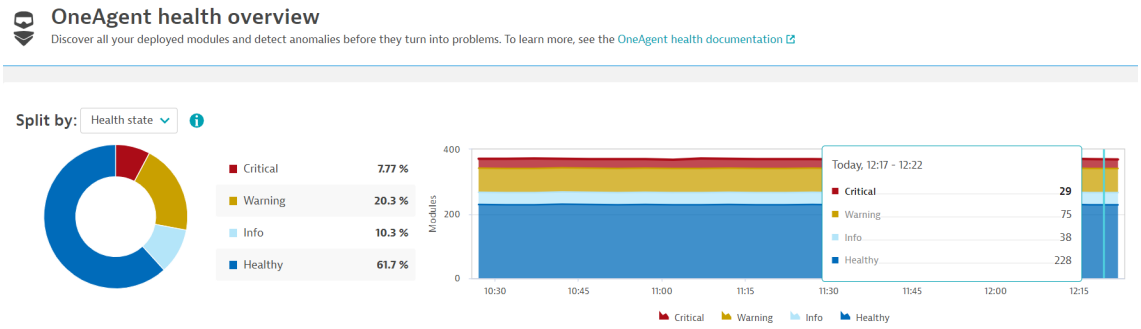

Figura 3.2: Status del monitoraggio del OneAgent

3.5 Modello di monitoraggio basato su IA

Dynatrace utilizza un motore di intelligenza artificiale avanzato chiamato Davis, per rilevare automaticamente anomalie nelle prestazioni delle applicazioni, dei servizi e dell'infrastruttura IT. Questo motore AI si distingue per la capacità di individuare e segnalare situazioni anomale, come degradi delle prestazioni, malfunzionamenti o indisponibilità del sistema. Tali problemi vengono identificati come deviazioni rispetto al comportamento di riferimento del sistema, consentendo un intervento tempestivo per mitigare i rischi operativi.
Davis implementa molte funzionalità finalizzate a migliorare l'efficacia dell'utilizzo della piattaforma.

3.5.1 Rilevamento delle Anomalie

Davis utilizza i dati di Dynatrace per monitorare continuamente le prestazioni di ogni aspetto delle applicazioni, dei servizi e dell'infrastruttura presenti nell'ambiente IT. Questo monitoraggio costante permette alla piattaforma di apprendere automaticamente tutte le metriche di riferimento (baseline) e lo stato complessivo di salute di ciascun componente, inclusi i tempi di risposta delle applicazioni e dei servizi.
L'analisi basata su baseline prende in considerazione automaticamente variabili critiche come la geolocalizzazione degli utenti, il tipo di browser utilizzato, il sistema operativo, la larghezza di banda della connessione e le azioni degli utenti. Grazie a questa intelligenza automatizzata, Dynatrace è in grado di rilevare

anomalie con un livello di granularità estremamente elevato, notificando tempestivamente eventuali problemi rilevati in tempo reale.
La funzionalità di baselining può essere personalizzata per adattarsi alle specifiche esigenze dell'azienda. È possibile regolare la sensibilità del sistema nel rilevare anomalie oppure, in casi particolari, definire soglie statiche personalizzate che sostituiscano quelle generate automaticamente.

3.5.2 Root Cause Analysis

La funzionalità di analisi di root cause analysis fornita da Davis di Dynatrace rappresenta uno strumento avanzato per l'identificazione delle origini dei problemi complessi all'interno di ambienti IT. Questo sistema analizza automaticamente tutte le informazioni catturate e ingerite dalla piattaforma, evidenziando le entità coinvolte dall' anomalia e individuando la radice del problema.
L'approccio è basato sul contesto e sfrutta le informazioni disponibili, come la topologia a livello di codice, per determinare con precisione la causa dell'anomalia. Questa capacità di analisi consente non solo di identificare rapidamente il problema principale, ma anche di fornire un quadro dettagliato delle dipendenze e delle relazioni tra le diverse componenti del sistema.
L'analisi consente di risparmiare tempo includendo automaticamente tutte le anomalie rilevanti e classificando i contributori alla causa principale in ordine di importanza. Questo approccio permette di determinare rapidamente la causa primaria del problema, riducendo al minimo i tempi di diagnosi e risoluzione.
Inoltre, il sistema riduce il carico di notifiche aggregando più anomalie connesse in un unico problema, quindi semplifica la gestione degli alert, evitando sovraccarichi informativi che potrebbero compromettere l'efficacia delle operazioni.

3.5.3 Correlazione e analisi degli eventi

Gli eventi in Dynatrace rappresentano diverse tipologie di anomalie singolari, come il superamento di una soglia da parte di una metrica, il degrado rispetto a un valore baseline. Oltre a queste anomalie, Dynatrace rileva e gestisce eventi informativi, come nuovi rilasci di software, modifiche alla configurazione e altri tipi di eventi rilevanti per il contesto operativo.
La correlazione degli eventi in Dynatrace automatizza l'analisi del vasto flusso di eventi individuali provenienti da migliaia di fonti e strumenti differenti. L'obiettivo principale della correlazione degli eventi è estrarre problemi significativi

dagli eventi in ingresso e derivarne insight operativi utili.
L'elaborazione e l'analisi degli eventi è un processo diviso in più parti:

- **Ingestione**: Gli eventi vengono acquisiti da varie fonti, inclusi agenti software, strumenti di monitoraggio e piattaforme di integrazione.
- **Normalizzazione**: Gli eventi acquisiti vengono trasformati in un formato uniforme per semplificare la loro elaborazione e analisi.
- **Creazione della topologia**: Gli eventi vengono correlati alle entità del sistema per costruire una mappa completa delle dipendenze e delle relazioni tra le diverse componenti dell'ambiente IT.
- **Aggregazione**: Gli eventi correlati vengono raggruppati per ridurne la complessità e rappresentare all'utente finale un problema unico e coerente.
- **Deduplicazione**: Eventi simili o duplicati vengono eliminati con l'obiettivo di fornire un quadro chiaro delle anomalie realmente rilevanti.

3.6 Grail database

Grail è il database progettato da Dynatrace per la gestione dei dati di osservabilità e sicurezza. Si presenta come una soluzione di archiviazione unificata per log, metriche, tracce, eventi e tutti i dati riguardanti l'APM. Tutti i dati archiviati in Grail sono interconnessi all'interno di un modello in tempo reale che riflette la topologia e le dipendenze presenti nell'ambiente monitorato.
La tecnologia di Grail è basata su un approccio schema-on-read e senza indici (indexless), progettata per garantire una scalabilità ottimale. Grazie a questa architettura, non è obbligatorio determinare in anticipo i casi d'uso dei dati di log, ad esempio durante l'ingestione. L'analisi basata su schema-on-read consente di eseguire qualsiasi attività analitica sui dati storici archiviati in Grail.
Grail offre numerosi vantaggi che lo rendono una soluzione innovativa e particolarmente efficiente per la gestione dei dati di osservabilità e sicurezza. In primo luogo, funge da sistema di archiviazione unificata, consentendo di gestire in un'unica piattaforma diverse tipologie di dati. Questo formato semplifica notevolmente la gestione delle informazioni, eliminando la necessità di ricorrere a sistemi frammentati.
Un altro aspetto rilevante è l'interfaccia unica che permette di interrogare diverse tipologie di dati attraverso un'unica piattaforma. Questa caratteristica consente di accedere facilmente alle informazioni necessarie, semplificando le operazioni

di analisi e monitoraggio.

3.6.1 Gestione dei dati di Grail

- **Ingestione**: Tutti i dati acquisiti da Dynatrace finiscono in Grail. Log ed eventi aziendali vengono inoltrati nel data lakehouse di Grail, dove gli endpoint di ingestione ricevono i dati e li incanalano come record nel sistema di elaborazione.

- **Elaborazione**: I record vengono trasformati e arricchiti con campi aggiuntivi, per poi essere destinati a specifici bucket, ossia contenitori logici per l'archiviazione.

- **Archiviazione**: I record vengono memorizzati nei bucket. Ogni bucket ha un periodo di conservazione e un tipo di record associato configurabile separatamente. Inoltre, i bucket vengono associati a tabelle del database DQL e l'accesso a tali tabelle può essere configurato per utenti o gruppi specifici.

- **Interrogazione**: Grazie al Dynatrace Query Language (DQL), è possibile interrogare tutti i tipi di dati archiviati. Le tabelle predefinite per Grail includono log ed eventi aziendali, che possono essere analizzati utilizzando comandi DQL dedicati, come il comando fetch per recuperare i record da una tabella.

3.6.2 Dynatrace Query Language

Dynatrace Query Language (DQL) è uno strumento per interfacciarsi a Grail, basato sul linguaggio Structured Query Language (SQL) che fornisce delle funzionalità aggiuntive dedicate all'analisi dei dati.
Una delle caratteristiche distintive di DQL è la sua estrema flessibilità. A differenza dei database relazionali tradizionali, come le tabelle SQL, DQL non richiede una definizione preliminare dello schema dei dati di input.
Questo lo rende ideale per processare dati provenienti da eventi arbitrari, eliminando la necessità di vincoli strutturali iniziali e garantendo una maggiore adattabilità alle diverse esigenze analitiche.
DQL consente agli utenti di esplorare in profondità i propri dati, scoprendo modelli nascosti, analizzando tendenze, individuando anomalie critiche e costruendo analisi avanzate su misura per i propri obiettivi.

La sua integrazione con Grail ne amplifica l'efficacia, offrendo un accesso diretto a un'ampia gamma di dati e assicurando un'elaborazione efficiente anche su grandi volumi di informazioni.
Per iniziare a utilizzare DQL o approfondire le sue funzionalità, è possibile consultare le risorse disponibili, come guide pratiche e documentazione tecnica, che forniscono una panoramica completa e numerosi esempi applicativi.
Grazie alla sua potenza e flessibilità, DQL rappresenta uno strumento essenziale per chiunque desideri massimizzare il valore dei dati monitorati.

3.7 OpenPipeline

OpenPipeline rappresenta una delle soluzioni più avanzate di Dynatrace per integrare l'osservabilità nei flussi DevOps, ottimizzando i processi di Continuous Integration e Continuous Delivery (CI/CD). Grazie a un framework modulare, OpenPipeline consente di automatizzare il monitoraggio, standardizzare le pratiche DevOps e migliorare l'efficienza nello sviluppo del software. Open Pipeline è distinta tra il flusso dei dati (data flow) e le fasi di elaborazione (data processing).

3.7.1 Flusso dei dati

Il flusso dei dati in OpenPipeline si articola in diverse fasi che coinvolgono raccolta, trasporto e archiviazione delle informazioni. Ogni elemento è progettato per garantire un monitoraggio continuo e senza interruzioni durante le pipeline CI/CD.
La raccolta dei dati avviene per mezzo di diversi strumenti come OneAgent o API personalizzate. Come spiegato precedentemente, queste informazioni vengono estratte da diverse sorgenti come Log, metriche, host e altro.
I dati raccolti vengono immediatamente trasportati attraverso endpoint sicuri verso Dynatrace, utilizzando API aperte. Durante questa fase, i dati vengono normalizzati per garantire una compatibilità uniforme con il sistema di monitoraggio. Inoltre, la trasmissione è ottimizzata per ridurre la latenza e garantire un'elaborazione quasi in tempo reale.
I dati vengono archiviati in Grail. Il database assicura la persistenza dei dati, rendendo disponibili le informazioni storiche per analisi retrospettive, e interconnette le varie tipologie di dati in un modello di tempo reale, collegando metriche, log

ed eventi alle relative applicazioni, processi e infrastrutture.

3.7.2 Elaborazione dei dati

Una volta raccolti e archiviati, i dati vengono elaborati attraverso un flusso strutturato che sfrutta l'intelligenza artificiale Davis per fornire insight approfonditi. I dati vengono quindi inseriti all'interno di una pipeline per poter essere elaborati. La pipeline è divisa nelle seguenti fasi:

- **Elaborazione**: fase in cui si prepano i dati per l'analisi. Si applicano dei filtri sui campi e le informazioni sensibili vengono nascoste. Inoltre si iniziano a fare alcune formattazioni per procedere ad una migliore elaborazione.

- **Estrazione delle metriche**: in questa fase si estraggono i dati riguardanti le metriche d'interesse. Queste informazioni verranno analizzate dall'intelligenza artificiale Davis per individuare modelli, schemi ricorrenti effettuando la correlazione dei dati e per fornire insight come visto nelle sezioni precedenti.

- **Estrazione dei dati**: estrazione di informazioni riguardanti i log che verranno spediti in un'altra pipeline per approfondire l'analisi. Dynatrace effettua un'analisi sui log per estrarre informazioni utili che l'intelligenza artificiale non riesce a ricostruire.

- **Permessi**: applicazione di misure che permettono di mantenere la sicurezza dei dati. Ogni tipologia di dato a seconda della sua importanza ha deiversi tipi di permessi.

- **Archiviazione**: i dati vengono salvati all'interno di Grail per poter permettere un facile accesso e mantenere un storico di tutti i dati.

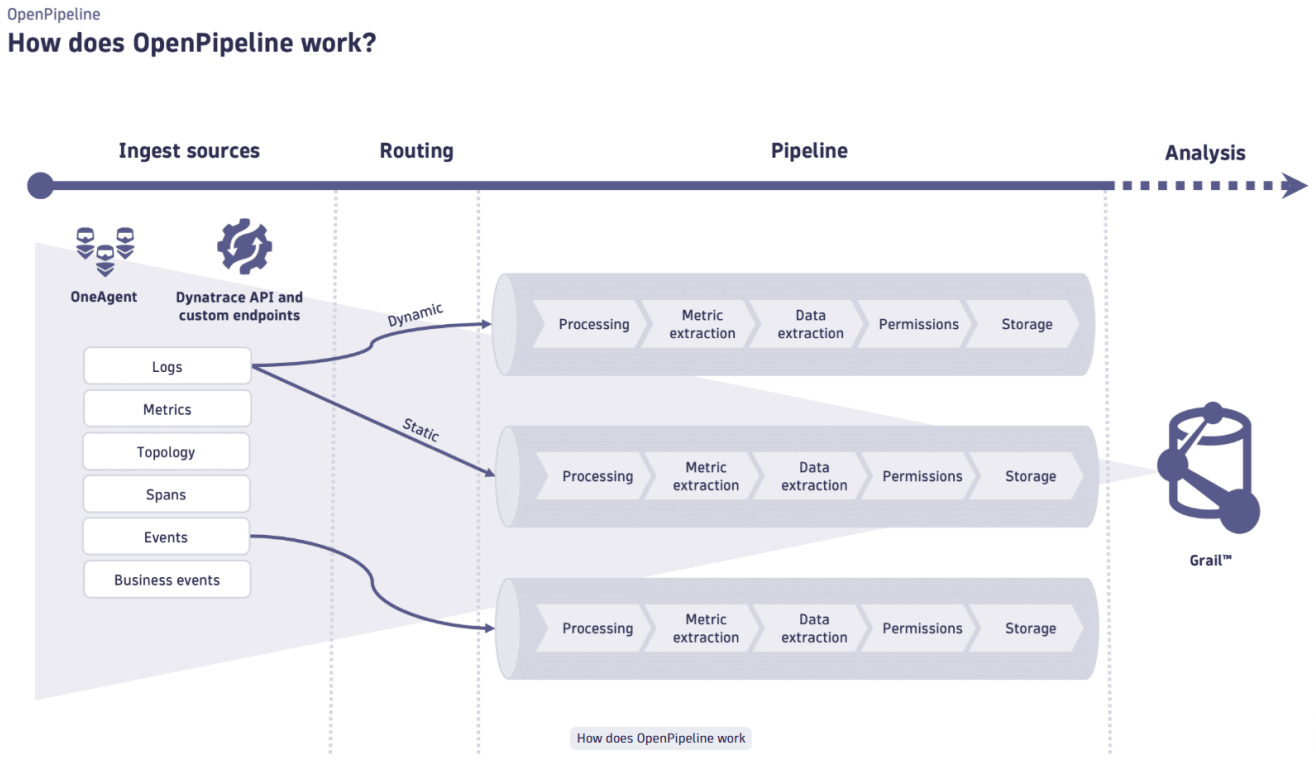

Figura 3.3: Funzionamento della racconta dati della OpenPipeline

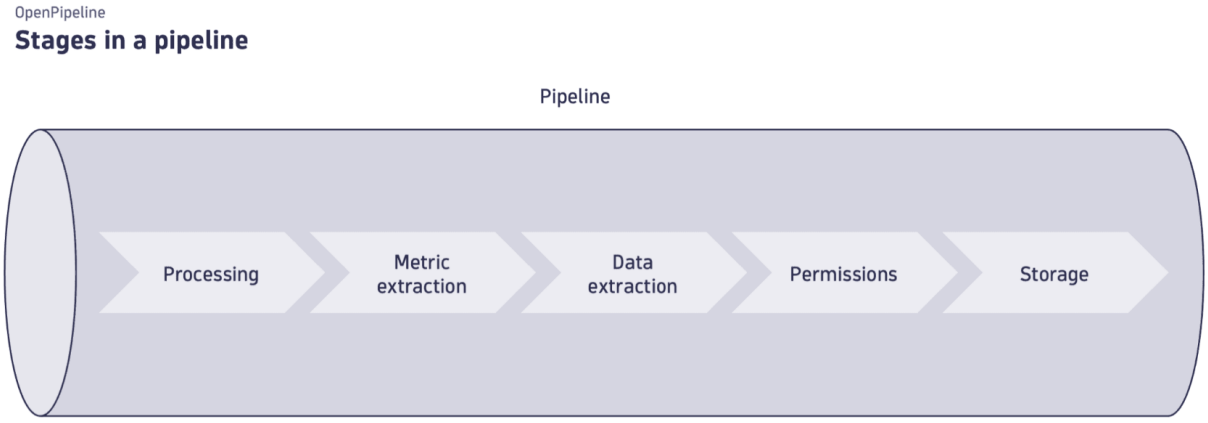

Figura 3.4: Schema delle fasi della OpenPipeline di Dynatrace

3.8 Smartscape

Smartscape è uno degli strumenti più avanzati e potenti offerti da Dynatrace per la visualizzazione topologica in tempo reale degli ambienti monitorati.
Smartscape utilizza una tecnologia di auto-discovery per fornire una rappresentazione rapida ed efficace di tutte le dipendenze topologiche presenti nell'infrastruttura, nei processi e nei servizi monitorati. L'asse verticale mostra le dipendenze full-stack attraverso tutti i livelli, mentre l'asse orizzontale visualizza le relazioni di chiamata in entrata e in uscita all'interno di ciascun livello. Con pochi clic, Smartscape consente di ottenere una visione topologica dettagliata dell'intero ambiente, fornendo maggiore controllo e comprensione delle interconnessioni e delle dipendenze.
Smartscape organizza l'ambiente monitorato in cinque livelli principali, ognuno rappresentante un tipo specifico di entità: applicazioni, servizi, processi, host e data center. Quando si abilita l'opzione "Mostra Problemi", ogni livello include una scheda dedicata con un indicatore di salute che evidenzia eventuali anomalie. Ad esempio, nel livello Host, un indicatore "1/33" indica che sono presenti 33 host monitorati, di cui uno presenta un problema. Gli altri livelli, come applicazioni, servizi, processi e data center, possono mostrare solo se tutte risultano in stato di salute.
Grazie a Smartscape, è possibile prendere decisioni strategiche sull'architettura dei servizi o sull'infrastruttura per migliorare le prestazioni delle applicazioni.
Inoltre, il sistema consente di analizzare le dipendenze sia tra livelli diversi (cross-tier) sia all'interno dello stesso livello (same-tier), come processi, host e servizi.
Questo approccio aiuta a comprendere meglio come le interazioni tra i componenti influiscano sulle prestazioni applicative. Infine, Smartscape permette di eseguire analisi approfondite per identificare e risolvere problemi specifici. Ad esempio, Dynatrace può rilevare un problema legato a una dipendenza esterna e aiutare a comprendere l'impatto di tale problema sulle prestazioni complessive delle applicazioni.

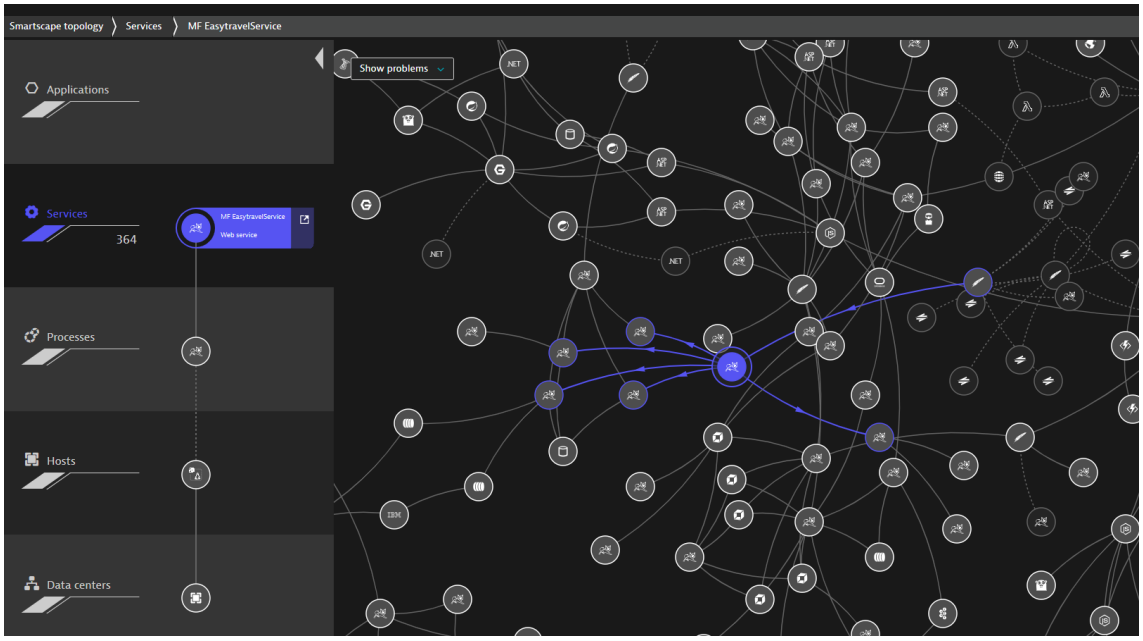

Figura 3.5: Visualizzazione della topogia legata al servizio EasyTravel

3.9 API di Dynatrace

Le Application Programming Interface (API) rappresentano un concetto fondamentale nell'informatica moderna, essendo il mezzo attraverso cui diversi software possono comunicare tra loro. In termini semplici, un'API è un insieme di regole e protocolli che definiscono come le applicazioni interagiscono, consentendo la condivisione di dati e funzionalità in modo standardizzato.
Un'API può essere vista come un'interfaccia che permette a un'applicazione di utilizzare le funzionalità o i dati di un'altra applicazione senza dover conoscere i dettagli interni del funzionamento di quest'ultima. In questo senso, l'API agisce come un "contratto" tra il fornitore di un servizio e chi lo utilizza.
Le API di Dynatrace costituiscono uno degli strumenti fondamentali per sfruttare appieno le potenzialità della piattaforma, offrendo la possibilità di automatizzare operazioni, estrarre dati e integrare Dynatrace con altri sistemi. Attraverso le API, Dynatrace consente di accedere in modo programmabile a tutte le funzionalità principali, garantendo flessibilità e scalabilità nella gestione dell'osservabilità e delle prestazioni applicative.
Per motivi di sicurezza, l'accesso alle API di Dynatrace è protetto da un meccanismo di autenticazione.
È necessario utilizzare token di accesso (API Token) generati direttamente dalla piattaforma Dynatrace. Questi token sono configurabili dall'amministratore, che può assegnare specifici permessi in base alle funzionalità richieste, come la lettu-

ra di metriche, l'estrazione di log o la gestione delle configurazioni.

```
# Esempio di token per l'accesso alle API
 dt0s01.ST2EY72KQINM7YN.G3DFPBEJYMODIKPRVMWFASS64NFH52PX6BNDV2RZM
```

Il token è diviso in 3 parti divise dal carattere "." :

- **Prefisso**: Identifica il tipo di token (se usato per API o Autenticazione), nell' esempio è "dt0s01".
- **Chiave pubblica**: Porzione di chiave visibile a tutti, è una stringa di 24 caratteri.
- **Chiave privata**: Porzione di chiave visibile solo dall'utente, è una stringa di 64 caratteri.

Nel capitolo dedicato al progetto verranno forniti dettagli sul funzionamento specifico dei vari endpoint utilizzati.

3.10 Un'alternativa: Splunk

Splunk è una piattaforma software progettata per raccogliere, indicizzare, visualizzare e analizzare dati generati da macchine in tempo reale. Si tratta di uno strumento molto potente e versatile, utilizzato principalmente per il monitoraggio, l'analisi e la gestione di grandi quantità di dati, provenienti da diverse fonti come log di sistema, metriche, eventi di rete, applicazioni e dispositivi IoT. Splunk può essere considerato un'alternativa a Dynatrace nel contesto dell'APM. Le principali particolarità di Splunk includono:

- **Indicizzazione dei dati non strutturati**: Splunk può elaborare e indicizzare dati non strutturati o semi-strutturati, trasformandoli in un formato che consente analisi rapide e ricerche flessibili. Questo è particolarmente utile per gestire dati complessi, come file di log o eventi di sistema.
- **Ricerca potente e linguaggio di query (SPL)**: Splunk utilizza il Search Processing Language (SPL), un linguaggio di query dedicato per effettuare ricerche dettagliate, analisi statistiche e visualizzazioni sui dati raccolti. SPL è altamente flessibile e permette di filtrare, correlare e trasformare i dati per estrarre informazioni utili.

- **Monitoraggio in tempo reale**: La piattaforma offre funzionalità di monitoraggio continuo, permettendo agli utenti di visualizzare dati in tempo reale. Questo è fondamentale per rilevare problemi o anomalie in sistemi complessi.
- **Machine learning integrato**: Splunk include funzionalità di apprendimento automatico che permettono di identificare pattern, prevedere anomalie e creare modelli analitici avanzati per anticipare problemi.
- **Scalabilità**: È progettato per gestire grandi volumi di dati e può essere scalato in base alle esigenze, rendendolo ideale per aziende di tutte le dimensioni, dalle piccole organizzazioni ai grandi ambienti enterprise.

Splunk viene utilizzato in molti scenari, come l'analisi dei log, la sicurezza informatica (SIEM), il monitoraggio delle applicazioni, la gestione delle infrastrutture IT e la business intelligence. Grazie alla sua capacità di fornire informazioni dettagliate e azionabili in tempo reale, è uno strumento fondamentale per le aziende che devono affrontare la crescente complessità dei dati e dei sistemi IT.

3.10.1 Splunk vs Dynatrace

Splunk e Dynatrace sono due piattaforme che operano nello stesso contesto ma sono implementate in un modo diametralmente opposto.

- **Approccio al monitoraggio e all'analisi dei dati:** Splunk si concentra sulla raccolta e sull'analisi approfondita dei log, offrendo strumenti avanzati per l'analisi retrospettiva e la gestione degli eventi. Il suo metodo si fonda sull'indicizzazione dei dati non strutturati, consentendo una ricostruzione dettagliata della sequenza degli eventi, elemento essenziale per l'analisi forense e per l'identificazione e la risoluzione degli incidenti. Al contrario, Dynatrace adotta un approccio integrato e proattivo al monitoraggio, basato sull'impiego di tecnologie di intelligenza artificiale e di funzionalità di auto-discovery. Tale metodologia permette di individuare in tempo reale eventuali anomalie, fornendo contestualmente indicazioni mirate per la risoluzione tempestiva dei problemi. Questa capacità di rilevamento preventivo risulta particolarmente vantaggiosa in ambienti dinamici e complessi, dove l'anticipazione delle criticità può tradursi in una gestione più efficiente dei sistemi informativi.
- **Interfaccia e usabilità:** Sul fronte dell'interfaccia, Dynatrace è concepito per garantire una visione immediata e intuitiva dello stato complessivo dell'in-

frastruttura. Gli utenti usufruiscono di dashboard sintetiche e facilmente navigabili, capaci di evidenziare le performance dei vari componenti e di segnalare prontamente eventuali criticità, facilitando così il monitoraggio continuo. In opposizione, Splunk richiede un maggiore grado di configurazione iniziale. Pur offrendo un'interfaccia meno immediata, la piattaforma compensa tale aspetto mediante un'elevata flessibilità: dashboard e query possono essere ampiamente personalizzate per soddisfare specifiche esigenze analitiche. Sebbene ciò comporti una curva di apprendimento più ripida, la possibilità di realizzare soluzioni analitiche sofisticate e su misura rappresenta un vantaggio significativo in contesti operativi complessi.

- **Scalabilità e prestazioni:** Entrambe le soluzioni sono progettate per operare in ambienti caratterizzati da un'elevata intensità di dati, ma differiscono per la tipologia di informazioni trattate e per le modalità di gestione della scalabilità. Dynatrace, focalizzato sul monitoraggio in tempo reale tramite metriche e tracciamenti, impiega soluzioni capaci di gestire flussi continui di informazioni, garantendo tempi di risposta rapidi e una diagnosi tempestiva delle anomalie. Splunk, invece, punta sulla capacità di indicizzare e processare grandi volumi di log ed eventi, offrendo un'architettura scalabile che può essere adeguata a requisiti specifici relativi alla capacità di elaborazione e archiviazione dei dati. Di conseguenza, la scelta tra le due piattaforme dovrà essere ponderata in base alla tipologia di dati da gestire e alle performance richieste, in relazione alle specifiche esigenze operative dell'organizzazione.

4
Progetto di tesi

4.1 Introduzione

Il progetto sviluppato si inserisce all'interno di questa cornice e si propone di sviluppare una serie di script per l'elaborazione e l'estrazione dei dati mediante le API di Dynatrace. In particolare, l'intento del lavoro è raccogliere, elaborare e aggregare informazioni relative alle User Actions e al Synthetic Monitoring, organizzandole in file strutturati per supportare analisi approfondite.
Il progetto si articola in quattro script differenti che riguardano strutture dati diverse tra loro:

- **Azioni Utente (User Actions)**

- **Monitor Sintetici**

I risultati di questo lavoro forniranno una base strutturata per effettuare analisi avanzate, con l'obiettivo di migliorare la gestione delle performance applicative e garantire un'esperienza utente ottimale. Attraverso l'uso delle API di Dynatrace e l'automazione dei processi di raccolta ed elaborazione dei dati, si intende facilitare l'identificazione di anomalie e la generazione di insight utili per i team di sviluppo e operation.

4.2 Concetti fondamentali

In questo capitolo andiamo ad approfondire e esplorare tutti i concetti utili al fine della comprensione del progetto

4.2.1 Azioni Utente

Le azioni utente, o "User Actions", rappresentano uno degli elementi centrali nel monitoraggio delle sessioni reali degli utenti(Real User Monitoring, RUM).
Una user action corrisponde a un'interazione tra un utente e l'applicazione monitorata. Queste azioni sono eventi specifici che si verificano, come cliccare su un pulsante, aprire una pagina, inviare un modulo, o interagire con un componente di un'applicazione web o mobile. Dynatrace traccia e analizza queste azioni per fornire una visione dettagliata dell'esperienza utente e del comportamento dell'applicazione.
Il monitoraggio delle azioni utente è fondamentale per comprendere le prestazioni dal punto di vista degli utenti finali, identificare eventuali problemi e ottimizzare l'esperienza digitale.
Esistono tre tipi di user action:

- **Azioni di caricamento (Load Actions)**: sono azioni che indicano il caricamento di una risorsa (come una pagina o una immagine). Queste azioni sono generate ogni volta che l'utente fa una load request al server (ad esempio un clic su un link) e si salvano numerosi parametri.

- **Richieste AJAX (XHR Actions)**: azioni che riguardano le richieste di tipo AJAX o altre richieste HTTP asincrone.

- **Azioni personalizzate (Custom Actions)**: azioni che possono essere impostate dagli utilizzatori di Dynatarce con il fine di tracciare anche casi eccezionali non previsti dalle azioni base.

Inoltre, è importante comprendere che, oltre a registrare l'azione in sè, Dynatrace salva anche i parametri e performance legate ad essa come il tempo di esecuzione o se è stata completata con successo.
Dynatrace organizza i dati nel database riguardanti le user action in diverse tabelle collegate tra loro.
I dati delle user action iniziano ad essere salvati dal momento in cui l'utente inizia la sessione. Quest'ultima contiene una lista di user action effettuate dall'utente ed esse vengono salvate all'interno del db.

La struttura del database delle User Session in Dynatrace può essere descritta come un insieme di tabelle correlate. La tabella principale è la User Session, che rappresenta una sessione completa di un utente. Ogni User Session è collegata a una serie di User Actions, che rappresentano le singole interazioni compiute dall'utente durante quella sessione. A loro volta, le User Actions possono essere associate a diversi Events, che descrivono eventi specifici accaduti durante una determinata azione, come errori o metriche di performance.
Il database interno utilizza una struttura denormalizzata per contenere i dati. La struttura denormalizzata è un modello di progettazione dei database in cui i dati vengono archiviati in un formato più "piatto" e ridondante rispetto alla normale forma relazionale. Questo approccio mira a migliorare le prestazioni delle query e a semplificare l'accesso ai dati, anche se ciò comporta una minore flessibilità e una gestione più complessa degli aggiornamenti. In una struttura denormalizzata, le informazioni possono essere duplicate in più righe o tabelle, evitando la necessità di relazioni complesse come le join.

```
1    # Esempio di query per l'accesso alle ser action
2    SELECT usersession.userSessionId, usersession.ip, application,
3        name, type, startTime,customErrorCount, hasCrash,
4        javascriptErrorCount, requestErrorCount
5    FROM useraction
6    WHERE (startTime >= TIME_FRAME_START AND endTime <= TIME_FRAME_END)
7        AND ((customErrorCount>0) OR (hasCrash=true) OR
8        (javascriptErrorCount >0) OR (requestErrorCount>0))
9    ORDER BY usersession.userSessionId ASC
```

In seguito riporto due tabelle che descrivono il significato di alcuni parametri salvati.

Termine	Descrizione
appVersion	La versione dell'applicazione utilizzata durante la sessione.
applicationType	Il tipo di applicazione (ad esempio, web, mobile).
bounce	Indica se la sessione è stata un bounce (una sessione a singola pagina).
city	La città da cui ha avuto origine la sessione.
connectionType	Il tipo di connessione di rete (ad esempio, Wi-Fi, cellulare).
device	Il tipo di dispositivo utilizzato (ad esempio, smartphone, tablet).
displayResolution	La risoluzione dello schermo del dispositivo.
duration	La durata della sessione.
endReason	La ragione per cui la sessione è terminata.
endTime	L'orario in cui la sessione è terminata.
hasCrash	Indica se si è verificato un crash durante la sessione.
hasError	Indica se si è verificato un errore durante la sessione.
internalUserId	L'ID interno dell'utente.
ip	L'indirizzo IP dell'utente.
isp	Il provider di servizi Internet utilizzato durante la sessione.
manufacturer	Il produttore del dispositivo.
networkTechnology	La tecnologia di rete utilizzata (ad esempio, 4G, 5G).
newUser	Indica se l'utente è nuovo.
numberOfRageClicks	Il numero di clic impulsivi durante la sessione.
osFamily	La famiglia del sistema operativo (ad esempio, Windows, iOS).
osVersion	La versione del sistema operativo.
screenWidth	La larghezza dello schermo del dispositivo.
startTime	L'orario in cui la sessione è iniziata.
totalErrorCount	Il numero totale di errori durante la sessione.
userActionCount	Il numero di azioni utente durante la sessione.
userExperienceScore	Il punteggio dell'esperienza utente per la sessione.
userId	L'ID dell'utente.
userSessionId	L'ID della sessione utente.
userType	Il tipo di utente (ad esempio, utente reale, utente sintetico).

Tabella 4.1: Descrizione di alcuni dei termini relativi alle sessioni utente.

Termine	Descrizione
apdexCategory	La categoria Apdex (Application Performance Index) dell'azione utente, che indica la soddisfazione dell'utente in base ai tempi di risposta.
cdnBusyTime	Il tempo trascorso in attesa delle risorse della rete di distribuzione dei contenuti (CDN).
cdnResources	Il numero di risorse caricate da una CDN.
cumulativeLayoutShift	Una metrica che misura la stabilità visiva di una pagina monitorando gli spostamenti imprevisti del layout.
customErrorCount	Il numero di errori personalizzati verificatisi durante l'azione utente.
firstPartyResources	Il numero di risorse caricate dal dominio di prima parte.
keyUserAction	Indica se l'azione utente è contrassegnata come un'azione utente chiave.
largestContentfulPaint	Il tempo impiegato per dipingere sullo schermo l'elemento di contenuto più grande.
requestErrorCount	Il numero di errori di richiesta verificatisi durante l'azione utente.
requestStart	L'istante in cui è stata avviata la richiesta per l'azione utente.
responseEnd	L'istante in cui è terminata la risposta per l'azione utente.
responseStart	L'istante in cui è iniziata la risposta per l'azione utente.
serverTime	Il tempo trascorso sul server per elaborare l'azione utente.
speedIndex	Una metrica che misura la velocità con cui il contenuto di una pagina è visibilmente popolato.
targetUrl	L'URL di destinazione dell'azione utente.
thirdPartyBusyTime	Il tempo trascorso in attesa delle risorse di terze parti.
thirdPartyResources	Il numero di risorse caricate da domini di terze parti.
type	Il tipo di azione utente (ad esempio, click, load, XHR).
PropertyCount	Il numero di proprietà personalizzate associate all'azione utente.
visuallyCompleteTime	Il tempo necessario affinché la pagina sia visivamente completa.

Tabella 4.2: Descrizione dei termini utilizzati nell'azione utente.

4.2.2 Monitor Sintetici

I monitor sintetici di Dynatrace sono strumenti utilizzati per monitorare attivamente le prestazioni e la disponibilità delle applicazioni simulando le interazioni degli utenti reali. A differenza del monitoraggio basato su sessioni reali (Real User Monitoring, RUM), i monitor sintetici creano traffico artificiale verso un'applicazione per testarne il comportamento in condizioni controllate e prevedibili.
Questi monitor permettono di verificare l'accessibilità, il tempo di risposta e la funzionalità delle applicazioni da diverse località geografiche, garantendo un controllo continuo anche quando non ci sono utenti attivi.
I monitor sintetici inviano richieste predefinite o eseguono script che simulano le azioni di un utente, come navigare su una pagina, compilare un modulo o effettuare un acquisto. I risultati di queste simulazioni includono metriche come i tempi di caricamento, gli errori HTTP, la disponibilità e il corretto funzionamento delle funzionalità.
Quindi riescono a raccogliere numerosi dati senza ricorrere al monitoraggio per gli utenti reali. È evidente che la sfida per creare un buon monitor sintetico sta nel scrivere uno script efficiente che simuli al meglio un utente reale. Esistono due tipi diversi di monitor sintetici:

- **Monitor Browser**: Il Browser Monitor è uno strumento avanzato pensato per simulare l'esperienza utente attraverso un'applicazione web, utilizzando un browser reale per eseguire azioni predefinite (clickpath) su un sito web. A differenza dell'HTTP Monitor, che si limita a inviare richieste HTTP o HTTPS, il Browser Monitor replica interazioni reali come clic, scroll, compilazione di moduli e navigazione tra le pagine. Impiega un browser come Chromium per garantire che le azioni simulate rispecchino il più possibile il comportamento di un utente umano. Inoltre, lo strumento consente di registrare uno "script di clickpath", dove vengono definiti i passi da eseguire. Il Browser Monitor fornisce metriche dettagliate, come i tempi di caricamento delle pagine, che indicano quanto tempo è necessario per caricare l'intera pagina o singoli elementi. Inoltre, permette di identificare errori nei passaggi del clickpath, cioè i punti in cui le azioni previste non vengono eseguite correttamente, e segnala eventuali problemi di rendering, come la visibilità o il malfunzionamento degli elementi della pagina. Questo tipo di monitoraggio può essere utilizzato per simulare il processo di login di un'applicazione per verificarne il corretto funzionamento, testare un'intera procedura di checkout su un sito e-commerce, dal momento in cui si selezionano i prodotti fino al pagamento, o per monitorare portali web complessi,

assicurandosi che non ci siano regressioni o interruzioni nei vari passaggi di navigazione.

- **Monitor HTTP**: L'HTTP Monitor è un tipo di monitoraggio sintetico che ha lo scopo di verificare la disponibilità e le prestazioni di un endpoint attraverso richieste HTTP o HTTPS. Viene utilizzato principalmente per monitorare API, siti web o servizi backend. Questo monitor esegue richieste verso un URL specifico e può includere intestazioni personalizzate, come quelle per l'autenticazione, o parametri di query. Supporta vari tipi di richieste, tra cui GET, POST, PUT e DELETE. Tra le metriche misurate ci sono il tempo di risposta, che indica quanto tempo ci vuole per completare la richiesta, e i codici di stato HTTP, utili per verificare la disponibilità e identificare errori come quelli della serie 4xx o 5xx. Inoltre, è in grado di esaminare i contenuti della risposta per verificare che contengano le informazioni previste, come una stringa specifica. Questo strumento può essere impiegato per monitorare la disponibilità di un'API REST, assicurarsi che un sito web risponda correttamente con un codice HTTP 200 o garantire che una pagina restituisca contenuti specifici, come un titolo o un messaggio atteso.

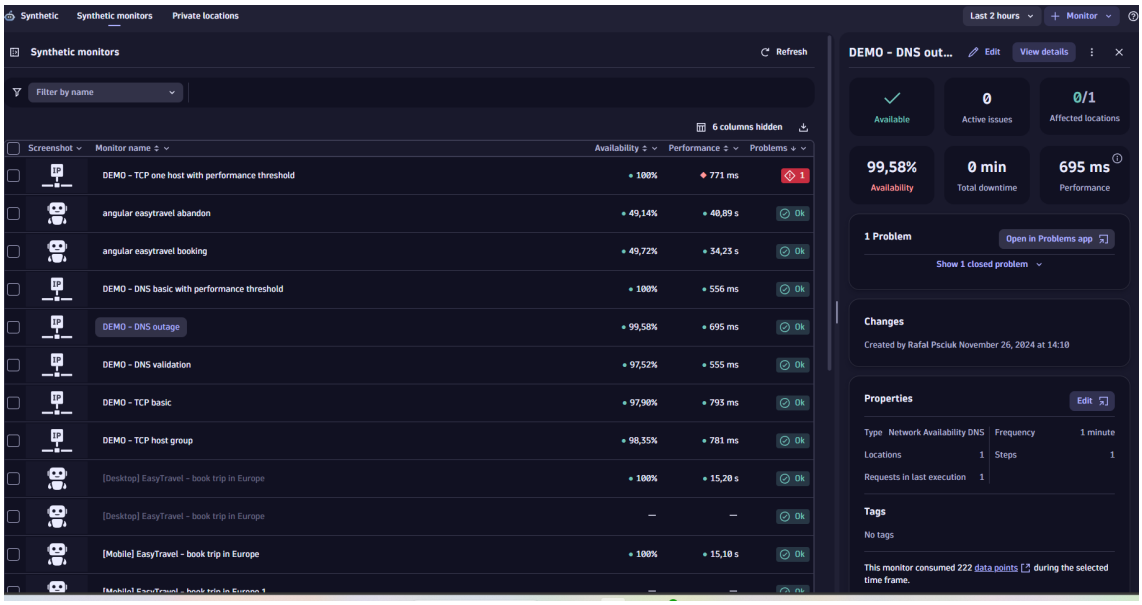

Figura 4.1: Pagina di Dynatarce dedicata ai monitor sintetici

4.3 Progettazione e struttura

La progettazione degli script ha seguito un approccio modulare e scalabile, pensato per integrarsi agevolmente in ambienti complessi. La fase iniziale ha previ-

sto una dettagliata analisi delle API fornite dalla piattaforma Dynatrace: è stato fondamentale comprendere la struttura degli endpoint, i formati di risposta e le modalità di autenticazione e interazione. Questo studio ha permesso di identificare le funzionalità chiave da implementare, come il recupero dei dati di performance, il monitoraggio in tempo reale e l'analisi degli eventi critici, definendo così un percorso chiaro per la successiva fase di sviluppo.
Il design degli script è stato orientato a garantire robustezza e flessibilità, adottando pratiche di codifica che facilitano la manutenzione e l'espansione futura.
Ogni script è stato progettato per eseguire operazioni specifiche, dalla semplice estrazione di informazioni al processamento complesso dei dati, fino alla generazione di report analitici utili per l'ottimizzazione delle performance dei sistemi monitorati.
Molto spesso Dynatrace salva tutti i dati in strutture predeterminate per poter effettuare delle analisi, ma questo approccio può risultare poco flessibile se si richiedono analisi di tipo differente.
Per questo motivo, lo scopo degli script realizzati è quello di raccogliere informazioni dalle strutture dati di dynatrace per poterle rielaborare e riorganizzarli in un formato più accessibile ai tecnici interni all'azienda.

4.4 Sviluppo del codice

Il progetto è diviso in quattro script differenti, ognuno dei quali ha l'obiettivo di raccogliere dati e informazioni per la creazione di file personalizzati con tutti i dati raccolti. Questo capitolo si occuperà di approfondire ognuno di questi script andando ad illustrarne il funzionamento e le strategie pratiche utilizzate durante lo sviluppo.

4.4.1 Aspetti comuni del codice

Nonostante gli script operino su dati diversi, è inevitabile che alcuni elementi si ripetano all'interno del codice. Quindi, per evitare le ripetizioni, in questo capitolo verranno illustrati i meccanismi comuni negli script.

Ognuno degli script è dotato di un sistema di logging per monitorare e documentare l'esecuzione dello script, facilitando così il debugging e la manutenzione. L'uso del logger permette di tenere traccia dei progressi operativi e di registrare in modo dettagliato eventuali errori o eccezioni, rendendo più semplice l'individuazione e la risoluzione di eventuali problemi.

Nonostante questa parte sia di semplice realizzazione, ho deciso di illustrarne il funzionamento perché il logger è lo strumento principale utilizzato dai tecnici per cercare di risolvere eventuali problemi.
Innumerevoli volte durante il mio stage ho visto i miei colleghi controllare i log per cercare di comprendere gli errori di alcuni script consegnati dai clienti non di loro creazione, quindi una buona implementazione riesce a far risparmiare molto tempo.
La tecnologia centrale utilizzata è la libreria standard Python "logging", che consente di configurare il livello di dettaglio dei messaggi, in questo caso impostato su DEBUG per ottenere informazioni molto granulari.
Per ogni giorno viene creato un log giornaliero che racchiude tutte le informazioni risultanti dall'esecuzione degli script.
Riporto qua sotto il tipico codice utilizzato per la creazione.

```
log_directory = "Logs/log_step1"
# Create directory if it doesn't exist
os.makedirs(log_directory, exist_ok=True)

log_file = os.path.join(log_directory, f"{today_date}
   _action_error_status.log")

# verify if the log file exist, if not create it
if not os.path.isfile(log_file):
    with open(log_file, 'w') as f:
        pass

logging.basicConfig(
    filename=log_file,
    filemode='a',
    format='%(asctime)s - %(levelname)s - %(message)s',
    level=logging.DEBUG,
)
logger = logging.getLogger()
```

Un'altro punto in comune è relativo alla gestione del tempo. Tutti i dati accessibili in Dynatrace sono vincolati a una finestra temporale: il compito consiste nell'ottenere le informazioni relative alla giornata precedente rispetto alla data di esecuzione dello script, per cui è essenziale impostare correttamente i parame-

tri temporali. Inoltre, occorre prestare attenzione ai diversi fusi orari, poiché il database di Dynatrace opera in base al fuso orario di Londra, aspetto che va considerato durante il calcolo dei dati.
Per una corretta gestione del tempo, ho fatto uso della libreria "datetime" e "timezone" per poter ottenere e modificare le informazioni relative alla zona temporale.

L'ultimo punto in comune degli script è sono le operazioni di lettura e scrittura su file.
Le api Dynatrace ritornano come risultati dei file di tipo JSON, per la generica lettura viene utilizzata la libreria standard di Python "json". In generale, all'interno degli script, viene letto il file per intero e salvato in una struttura dati locale, solo successivamente vengono effettivamente analizzati i dati. Per questo motivo, la lettura è standard per tutti gli script.
I file risultato che producono gli script sono file di tipo CSV, questo formato è stato scelto rispetto al JSON per poter contenere la dimensione del file stesso dato che alcuni di essi sono lunghi decine di milioni di righe. Anche in questo caso è stata utilizzata la libreria standard di Python chiamata "csv".

4.4.2 Script 1: azioni utente generali

Nella sezione precedente si è spiegato che le azioni effettuate da un utente sono salvate all'interno di Dynatrace. Ogni utente lascia quindi una traccia delle operazioni che effettua all'interno della sua sessione. Tra le varie informazioni raccolte ci sono i risultati di tali operazioni e noi siamo interessati a questi dati. Lo script ha l'obiettivo di estrarre dati relativi alle azioni utente da una piattaforma di monitoraggio (Dynatrace), elaborarli e salvarli in formati strutturati (JSON e CSV) per successive analisi.

Il programma in questione verte su due funzioni connesse tra loro:

- `get_user_session_data()`
- `query_user_sessions()`

La funzione `get_user_session_data()` si occupa di impostare correttamente i parametri da passare alla funzione che implementa l'endpoint. Lo scopo è quello di interrogare Dynatrace in modo tale da ottenere i dati relativi alle user action.
Tutte le API di Dynatrace restituiscono i dati con una dimensione massima pari a 5000 unità. Quando il risultato desiderato supera tale limite, è necessario

implementare un meccanismo di paginazione per gestire adeguatamente l'intero insieme di dati.
Per realizzare ciò, sfrutto il fatto che la query inviata all'endpoint restituisce i dati in ordine temporale. Attraverso un ciclo, invoco la funzione che utilizza l'endpoint per ottenere le informazioni; successivamente, in quanto non verranno recuperati tutti i dati disponibili, verifico l'ultimo dato ricevuto e ne estraggo il timestamp. A questo punto, sposto la finestra temporale in modo da recuperare le informazioni successive.
Questo semplice meccanismo permette di estrarre tutte le informazioni correttamente facendo attenzione al tempo, qua sotto lascio l'implementazione:

```
while current_timestamp_utc < endTimestamp_utc :
        current_retrived_data = query_user_sessions(query,
           current_timestamp_utc, endTimestamp_utc)
        # Check if there is data retrieved
        if current_retrived_data:
            last_entry = current_retrived_data[-1]  # Access the
                last entry
            last_timestamp = last_entry[1]  # Access the
               timestamp
            current_timestamp_utc = last_timestamp + 1
            formatted_timestamp = datetime.fromtimestamp(
               current_timestamp_utc / 1000, timezone.utc)
        if len(current_retrived_data) < 5000:    # last results
            current_timestamp_utc = endTimestamp_utc + 1
        data.extend(current_retrived_data)
```

La seconda funzione `query_user_sessions()` ha il compito di acquisire effettivamente le informazioni da Dynatrace.
Per implementare questa funzionalità, viene utilizzato un endpoint di Dynatrace che consente di applicare query DQL per recuperare i dati relativi alle sessioni utente.
Una volta impostati correttamente i parametri temporali, comprendenti i timestamp ottenuti dalla funzione precedente e la compensazione del fuso orario, è possibile creare la query necessaria per estrarre tutte le informazioni. Lascio qua sotto sia la query sia la descrizione dell'endpoint utilizzato.

```
SELECT userSessionId, startTime, duration, endTime, useraction.
   application, useraction.name, ip, useraction.responseStart,
   useraction.responseEnd
FROM usersession
WHERE startTime >= $TIME_FRAME_START AND endTime <
   $TIME_FRAME_END
ORDER BY startTime ASC
```

Endpoint `v1/userSessionQueryLanguage/table`

Parametro	Tipo	Descrizione
`query`	string	Query USQL da eseguire (es.: `SELECT country, city, COUNT(*) ...`).
`startTimestamp`	int64	Timestamp di inizio (UTC in millisecondi). Se 0, viene usato il valore predefinito (2 ore indietro).
`endTimestamp`	int64	Timestamp di fine (UTC in millisecondi). Se 0, viene usato il timestamp corrente.
`offsetUTC`	int32	Offset in minuti tra l'ora locale e UTC (es.: 120 per UTC+02:00, -300 per UTC-05:00).
`pageSize`	int32	Limite sul numero di risultati per pagina.
`pageOffset`	int32	Offset per la paginazione (es.: `pageSize`=50 e `pageOffset`=50 restituiscono le righe 51–100).
`addDeepLinkFields`	boolean	Abilita l'aggiunta di campi per deep linking. (Default: `false`).
`explain`	boolean	Fornisce informazioni aggiuntive sull'esecuzione della query (Default: `false`).

Riporto ancora un'ultima funzione per questo script meno importante, ma che si occupa di elaborare i risultati ottenuti.
La funzione `get_data_for_csv()` elabora un insieme di dati di sessioni raggruppando le informazioni per ora e per applicazione, aggregando i dati relativi

ad ogni azione. In particolare, per ogni sessione viene estratto l'orario di inizio (convertito in formato orario preciso all'ora) e, per ogni azione registrata, vengono calcolati il numero di occorrenze e il tempo di risposta totale (la differenza tra il tempo di inizio e quello di fine della risposta). Successivamente, la funzione itera sui dati raggruppati per ciascuna ora, applicazione e azione, calcolando la media del tempo di risposta (arrotondata a due decimali) e creando una stringa formattata contenente il timestamp, il nome dell'applicazione, il nome dell'azione, il conteggio delle occorrenze e il tempo medio di risposta. Il risultato finale è una lista di stringhe, ciascuna rappresentante una riga di dati pronta per essere esportata in formato CSV.

4.4.3 Script 2: gli errori delle azioni utente

L'obiettivo del secondo script è quello di raccogliere tutti i dati relativi agli errori delle user action, in particolare ci interessano gli errori di programmazione generati dalle sessioni utente.
Ricordiamo che i dati raccolti derivano dall'esperienza degli utenti sulla piattaforma. È evidente che, qualora durante una di tali esperienze si verificasse un errore che comporti la conclusione della sessione, la soddisfazione dell'utente risulti fortemente compromessa. Per questo motivo è estremamente importante collezionare tutti i dati riguardanti gli errori in modo tale da individuare i punti critici.

Lo script ha una struttura molto simile allo script precedente, dato che uso la stessa base dati.
La query DQL è stata modificata per poter raccogliere i dati richiesti.

```
1 SELECT usersession.userSessionId, usersession.ip, application,
     name, type, startTime,customErrorCount, hasCrash,
     javascriptErrorCount, requestErrorCount
2 FROM useraction
3 WHERE (startTime >= $TIME_FRAME_START AND endTime <=
     $TIME_FRAME_END) AND ((customErrorCount>0) OR (hasCrash=true)
      OR (javascriptErrorCount >0) OR (requestErrorCount>0))
4 ORDER BY usersession.userSessionId ASC
```

Mostro qua sotto come funziona la gestione del tempo usando le timeZone.

```
 # Set the timezone to Europe/Rome
rome_tz = ZoneInfo('Europe/Rome')

# Get the start of "yesterday" in Rome timezone (00:00:00)
yesterday_start_rome = datetime.now(rome_tz) - timedelta(days=2)
yesterday_start_rome = yesterday_start_rome.replace(hour=23,
   minute=59, second=9)
startTimestamp_rome = int(yesterday_start_rome.timestamp() *
   1000)

# Get the end of "yesterday" in Rome timezone (23:59:59)
yesterday_end_rome = datetime.now(rome_tz) - timedelta(days=1)
yesterday_end_rome = yesterday_end_rome.replace(hour=23, minute
   =58, second=59, microsecond=0)
endTimestamp_rome = int(yesterday_end_rome.timestamp() * 1000)
```

4.4.4 Script 3: monitor HTTP

Gli ultimi due script si concentrano sulle analisi riguardanti i monitor sintetici, in questo caso i monitor http.
I monitor http sono delle richieste atte a monitorare la disponibilità e la qualità di un servizio offerto da una API. Il fine dello script è quello di catturare i dati riguardanti il tempo di esecuzione e l'eventuale presenza di errori, andando a creare come risultato un file contenente i dati.
Al fine di ottenere i dati di interesse, è necessario impiegare una combinazione di chiamate API fornite da Dynatrace.
Con questo obiettivo, il programma è stato strutturato in diverse funzioni, ciascuna delle quali è dedicata alla raccolta dei dati necessari per la successiva aggregazione.

Il programma si articola nelle seguenti funzioni, che verranno descritte in dettaglio:

- `get_monitors_v1()`
- `get_monitor_details()`

- `get_metrics()`
- `get_data_for_csv()`

La funzione `get_monitors_v1()` si occupa di interfacciarsi con l'API di Dynatrace per recuperare l'elenco dei monitor HTTP attivi.
Essa effettua una richiesta specificando i parametri necessari (ad esempio, il tipo di monitor e lo stato di abilitazione) e, in caso di esito positivo, restituisce una struttura dati in formato JSON contenente informazioni essenziali quali l'identificativo dell'entità e il nome del monitor. In situazioni di errore, la funzione provvede a registrare i relativi messaggi di log e a restituire una struttura dati vuota.
Il formato del risultato di questa chiamata API è il seguente:

```
[
    {
        "name": "Inserisci Delega",
        "entityId": "HTTP_CHECK-XXXXXXX",
        "type": "HTTP",
        "enabled": true
    },
    {
        "name": "Open Banking",
        "entityId": "HTTP_CHECK-YYYYYYY",
        "type": "HTTP",
        "enabled": true
    },...
]
```

Si nota che il risultato ottenuto non contiene o dettagli di ogni monitor, ma solo delle semplici generalità. È compito della API successiva quello di permettere l'accesso a tutti i dati di un monitor.

Successivamente, per ogni ID ottenuto, il programma utilizza la funzione `get_monitor_details()` per poter estrarre tutti i dati riferiti ad un monitor, in particolare sono d' interesse gli step.
La funzione è progettata per ottenere informazioni dettagliate su uno specifico monitor, identificato attraverso il suo ID. Attraverso una chiamata API mirata, essa restituisce una serie di dati approfonditi relativi alla configurazione e alle proprietà del monitor, inclusi eventualmente i dettagli sui singoli step o richieste

che lo compongono.
Si ricorda che in generale i monitor sono degli script che vanno a simulare le azioni utente, quindi sono divisi in passi (step) che sono eseguiti in sequenza. Essi sono le singole fasi operative di un sintetico ed è in relazione ai singoli step che vengono raccolti i dati.
Grazie a questa funzione si riesce a raccogliere una lunga serie di informazioni riguardanti i singoli monitor, ma i dati d'interesse sono:

```
[
    {
    ...,
    "requests": [
            {
                "entityId": "HTTP_CHECK_STEP-AAAAA",
                "name": "Autorizzazione oAuth2",
                "sequenceNumber": 1
            },
            {
                "entityId": "HTTP_CHECK_STEP-BBBBB",
                "name": " send https://test",
                "sequenceNumber": 2
            }
        ],
    ...
    },
    ...
]
```

Una volta ottenuti questi dati, il programma riesce a catturare tutte le informazioni utili grazie all'utilizzo delle metriche. Come suggerisce la parola, le metriche sono delle misurazioni calcolate da Dynatrace che vengono effettuate per ogni step.

La funzione `get_metrics()` ha il compito di interrogare l'endpoint dedicato alle metriche di Dynatrace al fine di ottenere dati sulle performance di un monitor. In particolare, essa configura l'intervallo temporale di interesse (convertendo le date in millisecondi e tenendo conto della corretta zona oraria) e richiede informazioni quali la durata delle richieste HTTP e i codici di stato restituiti.
Il risultato dell'API, strutturato in formato JSON, contiene i dati metrici necessari

per valutare le performance, mentre eventuali problematiche durante il recupero dei dati vengono segnalate tramite log. Un esempio possibile di risultato può essere:

```
[
    "result": [
            {
                "metricId": "builtin:synthetic.http.duration",
                ...
                "data": [
                    {
                        "dimensions": [
                            "Get token",
                            "HTTP_CHECK_STEP-AAAAA",
                            "CSE",
                            "SYNTHETIC_LOCATION-1234"
                        ],
                        ,
                        "timestamps": [
                            1737932400000,
                            1737932460000,
                            1737932520000,
                            ...
                        ],
                        "values": [
                            5,
                            null,
                            6,
                            ...
                        ],
                    }
            }
]
```

Infine, la funzione `get_data_for_csv()` si occupa di elaborare i risultati di un'esecuzione associata a un monitor, combinando le informazioni relative allo step (come il nome sintetico e il nome dello step) con i dati metrici ottenuti. Il suo scopo è quello di estrarre e formattare i dati utili (ad esempio, il timestamp, la descrizione dell'errore, i valori di misura e un contatore degli errori) in una strut-

tura testuale idonea ad essere esportata in un file CSV. In questo modo, viene garantita una presentazione ordinata e facilmente analizzabile dei dati raccolti.
Il file creato avrà il seguente formato:

«timestamp;synthetic_name;step_name;error_desc;value;error_count»

4.4.5 Script 4: monitor browser

L'ultimo script è progettato per raccogliere i dati relativi ai monitor di tipo browser. La struttura di questo programma è simile a quella del programma precedente, pur utilizzando alcune differenze nelle chiamate API. In particolare, pur mantenendo la medesima logica di funzionamento, il programma sfrutta gli stessi endpoint delle API ma con parametri e modalità di interazione differenti, adattandosi alle peculiarità specifiche dei monitor di tipo browser.
Questo capitolo si concentrerà pertanto sull'analisi dettagliata delle API utilizzate nel contesto del monitoraggio dei browser, esaminando la loro struttura, i parametri rilevanti e la logica di accesso ai dati, tralasciando invece i risultati concreti ottenuti dall'esecuzione dello script.
Quindi riprendo l'elenco delle funzioni utilizzate dal precedente script:

- `get_monitors_v1()`
- `get_monitor_details()`
- `get_metrics()`
- `get_data_for_csv()`

La funzione `get_monitors_v1()` utilizza l'endpoint **v1/synthetic/monitors** per ottenere l'elenco dei monitor configurati in Dynatrace.
Sebbene l'endpoint preveda l'uso di diversi parametri, per il presente codice sono rilevanti in particolare i parametri "enabled" e "type".Il parametro "enabled" consente di filtrare i risultati in base allo stato operativo (attivo o disattivo), mentre il parametro "type" distingue tra i monitor browser e quelli HTTP.
In questo script, a differenza della versione precedente, si procede alla ricerca dei monitor di tipo browser.
Di seguito viene illustrata la struttura generale dell'endpoint con un esempio di utilizzo, evidenziando i parametri principali; per ragioni di spazio non sono riportati tutti i parametri, in quanto non utilizzati all'interno del programma.

```
url = f"{base_url}/v1/synthetic/monitors"
params = {
    'type': 'BROWSER',
    'enabled': 'true'
}
response = requests.get(url, headers=headers, params=params,
   verify=certifi.where())
```

Endpoint `v1/synthetic/monitors`

Parametro	Tipo	Descrizione
`managementZone`	integer ($int64$) (query)	Filtra l'insieme dei monitor risultanti includendo solo quelli che appartengono alla management zone specificata. Specificare qui l'ID della management zone.
`tag`	array[string] (query)	Filtra l'insieme dei monitor risultanti in base ai tag specificati. È possibile specificare più tag nel seguente formato: `tag=tag1&tag=tag2`. Il monitor deve corrispondere a tutti i tag specificati.
`location`	string (query)	Filtra l'insieme dei monitor risultanti includendo solo quelli assegnati a una specifica location Synthetic. Specificare qui l'ID della location.
`type`	string (query)	Filtra l'insieme dei monitor risultanti includendo solo quelli del tipo specificato: BROWSER o HTTP.
`enabled`	boolean (query)	Filtra l'insieme dei monitor risultanti includendo solo quelli abilitati (`true`) o disabilitati (`false`).

La funzione `get_monitor_details()` è progettata per estrarre i dettagli specifici di ciascun monitor utilizzando il relativo identificativo.
A tal fine, essa sfrutta l'endpoint **/v1/synthetic/monitors/monitorId**, il quale richiede soltanto in input l'ID del monitor per restituire le informazioni dettagliate ad esso associate.

Endpoint `v1/synthetic/monitors/{monitorId}`

Parametro	Tipo	Descrizione
`monitorId`	string (path)	ID the monitor sintetico richiesto.

La funzione `get_metrics()` è responsabile dell'estrazione dei dati relativi alle esecuzioni di ciascuno step dei monitor. Per ottenere queste informazioni, viene utilizzato l'endpoint **v2/metrics/query**, il quale prevede l'impiego di cinque parametri principali.
Il parametro metricSelector definisce le metriche da interrogare, nel nostro caso relative alla durata delle richieste e agli errori. Il parametro entitySelector riceve in input l'ID dello step da analizzare. I parametri from e to stabiliscono i limiti temporali entro cui eseguire la query. Infine, il parametro resolution indica il numero di data points, determinando la granularità dei dati restituiti. Grazie a questa struttura, la funzione raccoglie in maniera dettagliata e organizzata le informazioni sulle performance e sugli eventuali errori riscontrati durante l'esecuzione dei vari step.

```
1 params = {
2     'metricSelector': 'builtin:synthetic.browser.event.
        actionDuration.load:names:parents, builtin:synthetic.
        browser.event.actionDuration.xhr:names:parents, builtin:
        synthetic.browser.event.errorCodes:names:parents',
3     'entitySelector':  f'entityId("{monitorId}")',
4     'from' : startTimestamp_rome,
5     'to' : endTimestamp_rome,
6     'resolution': '1m'     # devo mettere ogni minuto
7 }
8 response = requests.get(url, headers=headers, params=params,
    verify=certifi.where())
```

Endpoint `v2/metrics/query`

Parametro	Tipo	Descrizione
`metricSelector`	string (query)	Seleziona le metriche per la query in base alle loro chiavi. È possibile selezionare fino a 10 metriche per query. È possibile specificare più chiavi metriche, separate da virgole. Se la chiave della metrica contiene simboli, deve essere racchiusa tra virgolette (`"`) e i caratteri speciali devono essere preceduti dal carattere di escape ( ). Operatori di trasformazione aggiuntivi possono essere impostati utilizzando i due punti (:).
`resolution`	string (query)	La risoluzione desiderata dei punti dati. Opzioni: numero di punti dati (predefinito: 120) oppure intervallo di tempo tra i punti dati con unità valide: `m` (minuti), `h` (ore), `d` (giorni), `w` (settimane), `M` (mesi), `q` (trimestri), `y` (anni).
`from`	string (query)	L'inizio dell'intervallo di tempo richiesto. Formato: timestamp in millisecondi UTC. Predefinito: `now-2h`.
`to`	string (query)	La fine dell'intervallo di tempo richiesto. Formato: timestamp in millisecondi UTC. Predefinito: timestamp corrente.
`entitySelector`	string (query)	Specifica l'ambito delle entità per la query. Vengono inclusi solo i punti dati provenienti dalle entità corrispondenti.

5

Risultati

Il presente capitolo è dedicato all'analisi dei risultati ottenuti durante lo sviluppo del progetto, con particolare attenzione a due aspetti fondamentali: la quantificazione del codice prodotto e l'interpretazione dei dati raccolti.
Nella prima parte, verrà fornita una valutazione strutturata del codice implementato, evidenziandone la complessità, la modularità e l'efficienza. Saranno esaminati aspetti quantitativi, quali il numero di righe di codice, il grado di documentazione e la distribuzione delle funzionalità all'interno delle diverse componenti del progetto. Questa analisi permetterà di valutare la qualità del software sviluppato in termini di manutenibilità e scalabilità.
Successivamente, verranno illustrati i risultati derivanti dall'estrazione e dall'analisi dei dati raccolti tramite la piattaforma Dynatrace. Saranno presentate le principali evidenze emerse dall'elaborazione delle informazioni, con particolare riferimento ai trend individuati, alle metriche prestazionali e alle possibili implicazioni per l'ottimizzazione delle applicazioni monitorate.
L'obiettivo di questo capitolo è dunque fornire una visione chiara e dettagliata dell'impatto del lavoro svolto, evidenziando sia la solidità dell'implementazione tecnica sia il valore informativo dei dati ottenuti.

5.1 Qualità del codice

Come illustrato nel capitolo precedente, il progetto ha portato allo sviluppo di quattro script distinti, realizzati interamente in Python e per un totale approssimativo di 1.400 righe di codice suddivise su quattro script differenti.
L'implementazione si basa su un'architettura modulare che ha richiesto la scrittura di circa 30 funzioni, ognuna progettata per soddisfare requisiti specifici e contribuire al raggiungimento degli obiettivi. Tale scelta progettuale ha permesso di suddividere il problema complesso in componenti più gestibili, facilitando l'eventuale estensione e la manutenzione futura del software.

È importante notare che, all'interno del progetto, alcune funzioni risultano duplicate, in particolar modo quelle riguardanti la gestione dei file. Questa scelta è stata deliberata al fine di garantire l'indipendenza operativa di ciascuno degli script, eliminando ogni potenziale dipendenza da file esterni, ad eccezione delle librerie standard di Python. Tale approccio ha consentito di ottenere una maggiore portabilità e modularità, in quanto ogni script risulta completamente autonomo e in grado di operare in ambienti differenti senza necessità di riconfigurazioni aggiuntive.
Inoltre, sebbene alcune funzioni presentino similitudini nel loro scopo, le sostanziali differenze nei requisiti di esecuzione hanno reso necessaria una loro riscrittura, garantendo così che ogni modulo operi in modo ottimale e risponda in maniera specifica alle diverse esigenze computazionali.

L'approccio modulare adottato, unitamente ad una strutturazione accurata del codice e ad un elevato livello di documentazione interna, ha notevolmente semplificato la manutenzione del software.
L'uso sistematico dei commenti, abbinato all'integrazione di un sistema di logging dettagliato, ha facilitato le operazioni di debug e l'analisi degli errori, rendendo il codice accessibile anche a tecnici non particolarmente esperti nel dominio specifico del progetto. Tale metodologia non solo assicura che il software possa essere aggiornato e adattato in maniera efficace, ma garantisce anche la sostenibilità e l'evoluzione futura del progetto.
L'adozione di queste best practice rappresenta, infatti, un elemento chiave per rispondere tempestivamente a eventuali nuove esigenze funzionali o a cambiamenti nei requisiti di sistema, consolidando la resilienza e la flessibilità dell'intera architettura software.

Gli script sviluppati presentano una notevole complessità esecutiva, in gran parte dovuta alla necessità di implementare cicli annidati per garantire l'uso corretto delle API.
I tempi di esecuzione sono direttamente proporzionali alla quantità di dati raccolti da Dynatrace: un aumento dei dati comporta infatti un maggior numero di operazioni di paginazione, che a loro volta richiedono ulteriori chiamate alle API. Gli script, infatti, sono progettati per raccogliere dati relativi a una determinata categoria, prendendo come riferimento la giornata precedente. È evidente che, in presenza di una piattaforma a capacità limitata, i tempi di esecuzione risulteranno contenuti, mentre su una piattaforma estesa si osserverà un considerevole prolungamento dei tempi di elaborazione.

5.2 Risultati ottenuti

Come già anticipato precedentemente, il progetto è stato testato su un caso reale, in cui è stato implementato presso un cliente aziendale. Questo cliente dispone di una vasta infrastruttura IT, la quale ha generato una quantità notevole di dati, risultando in una sfida significativa sia in termini di volume che di complessità computazionale.

In particolare, durante le operazioni di test, gli script 2 e 4 hanno evidenziato prestazioni notevolmente efficienti, con tempi di esecuzione inferiori a un minuto. Lo script 3, invece, ha registrato tempi compresi tra i 2 e i 3 minuti, mentre lo script 1, il più impegnativo dal punto di vista computazionale, ha richiesto circa 10 minuti per completare l'intero ciclo di operazioni. È utile approfondire questi dati, analizzando con maggiore precisione i tempi di esecuzione di quest'ultimo scenario.
Lo script 1 è progettato per occuparsi della raccolta e dell'elaborazione integrale dei dati relativi alle azioni degli utenti, senza applicare alcuna forma di scrematura preliminare.
L'implementazione sfrutta numerose chiamate alle API di Dynatrace, operazioni che, nel loro numero elevato, comportano la creazione e la gestione di un file contenente all'incirca 30 milioni di righe. Tale volume di dati evidenzia la complessità dell'operazione, in quanto la gestione, l'organizzazione e la successiva elaborazione di un dataset così vasto richiedono un considerevole impegno computazionale. Un elemento rilevante da considerare è il tempo impiegato nell'attesa delle risposte alle chiamate API: la frequenza elevata di tali richieste, indi-

spensabili per il corretto reperimento dei dati, contribuisce significativamente alla durata complessiva delle operazioni.
Per questo motivo è importante sottolineare che questi script non sono stati concepiti per essere eseguiti in tempo reale, bensì per operare in modalità batch durante le ore notturne, al fine di non interferire con le prestazioni operative della piattaforma Dynatrace.

L'obiettivo principale del progetto consiste nella generazione di file di dati facilmente accessibili e interpretabili da parte di tecnici specializzati, per agevolare successive analisi. In tale ottica, la gestione diretta di un file contenente milioni di righe risulterebbe eccessivamente complessa, rendendo necessaria l'adozione di tecniche di compressione e ottimizzazione.
Nel caso specifico dello script 1, il processo di elaborazione consente di ridurre il file iniziale, contenente circa 30 milioni di righe, ad un dataset più gestibile di circa 800.000 righe. Pur rimanendo un volume consistente, tale riduzione facilita l'accesso e l'analisi dei dati, migliorando notevolmente la fruibilità del prodotto finale.
È altresì rilevante evidenziare come la quantità dei dati raccolti sia strettamente correlata all'utilizzo dell'infrastruttura da parte degli utenti. Ad esempio, nei giorni caratterizzati da minori interazioni, come la domenica, il volume dei dati elaborati è sensibilmente inferiore rispetto ai giorni feriali, con differenze di ordine di grandezza che incidono in modo determinante sui tempi di esecuzione degli script. Questa variabilità sottolinea l'importanza di un'architettura scalabile e di strategie di gestione dei dati che possano adattarsi dinamicamente alle diverse condizioni operative.

L'esecuzione degli script ha portato alla creazione di diversi file riguardanti i dati d'interesse raccolti. A seconda del giorno di riferimento dei dati, si ottengono dei file finali diversi. In particolare, nel caso degli script 1 e 2 che fanno riferimento alle azioni utente, perché fanno riferimento direttamente al comportamento di utenti reali.
In particolare, lo script 1 ha prodotto un ampio file che raccoglie in maniera esaustiva l'elenco delle diverse azioni compiute da tutti gli utenti nel corso della giornata precedente. Oltre alla semplice registrazione delle azioni, il file include anche il tempo impiegato per ciascuna di esse. Queste informazioni consentono di individuare in maniera analitica le azioni più onerose dal punto di vista computazionale e di rilevare i periodi in cui il sistema è maggiormente trafficato. Dall'analisi dei dati emerge, infatti, che durante le ore notturne l'infrastruttura risulta

meno utilizzata e, di conseguenza, più efficiente, mentre al mattino e nel primo pomeriggio si registra un aumento significativo del traffico, comportando tempi di risposta maggiori per gli utenti, i quali possono portare a un peggioramento dell'esperienza utente.

Parallelamente, lo script 2 è stato sviluppato per raccogliere e analizzare i dati relativi agli errori delle user action. Attraverso questo strumento, è possibile identificare le azioni che presentano maggiori criticità.
Nel caso in esame, si è rilevata la presenza di errori di tipo "javascript_error", dovuti a problematiche inerenti la programmazione lato client; tuttavia, la maggior parte degli errori riscontrati è riconducibile a problematiche di comunicazione con il server, evidenziate dalla frequente comparsa di "request_error". Analogamente a quanto osservato per lo script 1, anche in questo caso si nota una maggiore incidenza di tali errori nelle fasce orarie in cui l'infrastruttura è maggiormente sollecitata, suggerendo che l'aumentato carico di lavoro influisce negativamente sulla stabilità delle comunicazioni.

Gli script 3 e 4, invece, si focalizzano sui dati relativi ai monitor sintetici di tipo browser e HTTP. Entrambi gli script raccolgono informazioni riguardanti i tempi di esecuzione delle operazioni di monitoraggio e l'eventuale presenza di errori. Al fine di facilitare una successiva fase di visualizzazione ed elaborazione dei dati, per ciascuno di questi due script si è optato per la creazione di due file distinti: il primo file contiene un elenco dettagliato di tutti i dati raccolti, con i relativi valori, mentre il secondo file prevede un'aggregazione dei dati mediante il calcolo della media degli step ripetuti. Questa doppia organizzazione permette di analizzare le informazioni sia a livello granulare che aggregato, offrendo una visione completa delle prestazioni del sistema.
Dall'analisi dei risultati si evince che i monitor sintetici, in media, registrano un numero inferiore di errori rispetto alle azioni utente; tuttavia, è importante considerare che il numero complessivo di esecuzioni dei monitor è significativamente inferiore, con il rischio che la rappresentatività del testing possa risultare non completa. In ogni caso, l'errore più ricorrente in questo contesto è stato un "internal_server_error", indicativo di problematiche legate alla gestione del server.

6
Conclusioni

Lo scopo principale del progetto è stato lo sviluppo di una serie di script finalizzati a migliorare la leggibilità dei dati da parte del personale tecnico.
Tali script sono stati progettati non solo per facilitare l'interpretazione immediata delle informazioni, ma anche per formattare i dati in modo tale da renderli prontamente utilizzabili in successive analisi e processi decisionali.

In particolare, gli script hanno contribuito a standardizzare il formato dei dati, semplificando così l'integrazione con altri strumenti di analisi e garantendo una maggiore coerenza nell'elaborazione delle informazioni. Questo approccio ha consentito di ottimizzare il flusso di lavoro e di ridurre il tempo necessario per la preparazione e la verifica dei dati stessi.

Il progetto sviluppato potrebbe già essere adottato dall'azienda, in quanto integra tutte le componenti necessarie per operare in un ambiente reale. In particolare, il sistema è dotato di strumenti fondamentali quali il meccanismo di logging, indispensabile per le attività di debug, e la gestione dei token d'accesso, che garantisce un adeguato livello di sicurezza e controllo. Questi elementi, combinati, consentono una rapida implementazione e un funzionamento efficace del progetto all'interno dei processi aziendali, rendendolo una soluzione pronta all'uso.

In conclusione, sono soddisfatto del lavoro svolto. Sebbene ci siano ancora margini di miglioramento, il risultato ottenuto offre una buona qualità del servizio richiesto. Questo progetto mi ha insegnato molte cose durante il suo sviluppo, contribuendo in maniera significativa alla mia crescita professionale.

Ringraziamenti

Con la conclusione di questo progetto si chiude un lungo percorso accademico, caratterizzato da sfide, imprevisti, ma anche da momenti di sollievo e soddisfazione. Desidero esprimere il mio sincero ringraziamento a tutte le persone che hanno reso possibile questo cammino, quelle che, anche se per un breve periodo, mi hanno accompagnato lungo questa strada.

Un sentito ringraziamento va alla mia famiglia, che mi ha sostenuto non solo a livello finanziario, ma soprattutto morale, specialmente nei momenti di incertezza, quando il futuro sembrava incerto e le difficoltà apparivano insormontabili.

Un ringraziamento speciale a Filippo e Giulia, le due persone che mi sono state più vicine durante questo percorso. Filippo, con il suo supporto razionale, è stato al mio fianco per innumerevoli anni, un amico di lunga data che rappresenta un legame raro e prezioso, che poche persone hanno l'onore di avere, un vero amico. Giulia, con il suo supporto emotivo, mi ha sostenuto, soprattutto nell'ultimo periodo, fortificando la mia determinazione e offrendo conforto nei momenti di difficoltà, riempiendo le mie giornate di affetto e serenità, grazie alla sua felicità maledettamente contagiosa.

Un pensiero particolare va al mio vecchio coinquilino Francesco, con cui ho condiviso le giornate in quel piccolo alloggio torinese: grazie a lui, anche le giornate più grigie sono diventate più colorate. Un grazie anche ad Anna, che mi ha insegnato il valore di avere un'amica che sa offrire una prospettiva femminile unica e arricchente sulle questioni della vita e anche sulle semplici chiacchierate.

La lista delle persone che meritano il mio ringraziamento è lunga, e purtroppo, per motivi di spazio, non posso citarle tutte. Un pensiero va a tutti i miei amici, che da quelli delle scuole superiori a quelli incontrati al Politecnico, sono stati vicini e hanno reso il mio lungo percorso universitario più leggero e piacevole.

Bibliografia

[1] Alten. *Alten Website.* Disponbile a: https://www.alten.it/.

[2] Che cos'è un APM (Application Performance Monitor)? Disponbile a: https://managedserver.it/che-cose-un-apm-application-performance-monitor/.

[3] Application Performance Monitoring definizione. Disponbile a: https://factorial.it/blog/application-performance-monitoring-definizione/.

[4] IBM. *Application Performance Management.* Disponbile a: https://www.ibm.com/topics/application-performance-management.

[5] Wikipedia. *Application Performance Management.* Disponbile a: https://en.wikipedia.org/wiki/Application_performance_management.

[6] Dynatrace. *What is APM?* Disponbile a: https://www.dynatrace.com/news/blog/what-is-apm/.

[7] Alten Italia. *Alten APM - Dynatrace.* Disponbile a: https://www.alten.it/2023/10/05/alten-apm-dynatrace/ (Published: 5 ottobre 2023).

[8] Dynatrace. *Dynatrace Documentation.* Disponbile a: https://docs.dynatrace.com/docs/.