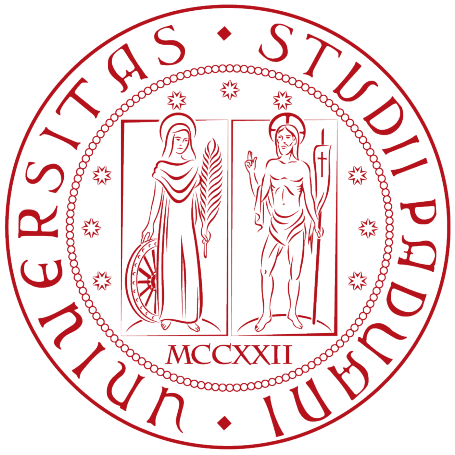
Università degli Studi di Padova

Dipartimento di Matematica "Tullio Levi-Civita"

Corso di Laurea in Informatica



Sviluppo di una piattaforma di monitoraggio del traffico utente mediante APM e IA

Tesi di laurea

Relatore
Prof. Marco Zanella

Laureando
Marco Cola
Matricola 2079237

Anno Accademico 2024-2025

Marco Cola: *Sviluppo di una piattaforma di monitoraggio del traffico utente mediante APM e IA,* Tesi di laurea, © Dicembre 2025.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit.

— Oscar Wilde

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di 320 ore, dal laureando Marco Cola presso l'azienda Kirey S.r.l.
L'obiettivo principale dello stage è stato lo sviluppo di una piattaforma per il monitoraggio e l'analisi dei dati di navigazione relativi ad una *web application*, con particolare attenzione alla raccolta, indicizzazione ed elaborazione dei *log*.
Il progetto si è articolato in quattro fasi principali:

- Una prima fase di preparazione dell'ambiente di lavoro, comprendente l'individuazione, l'installazione e la configurazione delle componenti *software* necessarie, con successiva verifica del corretto funzionamento e della connettività tra i moduli;
- Una seconda fase di implementazione dell'estrazione dei dati, realizzata attraverso la configurazione di agenti di raccolta, la creazione di *pipeline* di *log* tramite *Logstash* e l'invio dei dati a *Elasticsearch*, con validazione del processo di acquisizione e indicizzazione;
- Una terza fase di elaborazione e rappresentazione grafica dei dati, che ha previsto lo sviluppo di *script* per la generazione di traffico utente e monitoraggio, l'analisi e aggregazione dei dati su *Elasticsearch* e la realizzazione di dashboard avanzate in *Kibana* con metriche di performance, accesso e flussi utente. Sono state inoltre configurate regole di *alerting* e notifiche per la rilevazione in tempo reale di anomalie mediante *Machine Learning*;
- Una fase finale di documentazione tecnica del progetto, contenente la descrizione delle tecnologie e dei prodotti utilizzati, la rappresentazione dei flussi logici dell'applicazione, nonché un'analisi dei pro e contro di ciascuna componente e delle principali criticità riscontrate.

Lo stage ha permesso di acquisire competenze trasversali nell'ambito dell'osservabilità delle applicazioni *web*, approfondendo l'utilizzo dello *stack Elastic* (*Elasticsearch, Logstash, Kibana*) e l'integrazione con strumenti di monitoraggio automatizzato, con un approccio volto alla creazione di soluzioni scalabili, robuste e orientate al miglioramento continuo delle prestazioni.

"Quando incontri un uomo con la spada, estrai la tua spada:
non recitare poesie a chi non è poeta"

— proverbio Ch'an

Ringraziamenti

Innanzitutto, vorrei esprimere la mia gratitudine al Prof. Marco Zanella, relatore della mia tesi, per l'aiuto e il sostegno fornitomi durante la stesura del lavoro.

Desidero ringraziare con affetto i miei genitori per il sostegno, il grande aiuto e per essermi stati vicini in ogni momento durante gli anni di studio.

Padova, Dicembre 2025 Marco Cola

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

1.1 L'azienda

Kirey Group è uno dei *system integrator* europei più dinamici e in crescita, specializzato nell'accompagnare le imprese nei percorsi di trasformazione digitale e di adozione di modelli *data-driven*. Con sede principale in Italia e una presenza consolidata in diversi Paesi europei ed extraeuropei, il Gruppo conta oltre 1500 dipendenti ed opera in dieci Paesi.

La missione di Kirey è rendere l'innovazione accessibile, trasformando il potenziale tecnologico in valore economico e in nuovi modelli di *business*. L'azienda si distingue per un approccio che unisce affidabilità tecnica, innovazione, competenza centrata sul lavoro delle persone e sinergia cross-funzionale, elementi che costituiscono i valori fondanti del marchio.

Il manifesto del gruppo sintetizza questa filosofia nel concetto "*Data Made Human*", ovvero la volontà di tradurre la complessità dei dati in soluzioni comprensibili, intuitive e ad alto impatto, mettendo sempre la persona al centro della tecnologia.

Figura 1.1: Figura 1.1 - Logo di Kirey S.r.l.

1.1.1 Storia e servizi

La storia del gruppo affonda le radici negli anni Settanta e, attraverso fusioni, acquisizioni e nuove fondazioni, ha portato alla nascita di Kirey Group nel 2016. Negli anni successivi l'azienda ha accelerato la propria espansione internazionale integrando nuove realtà, consolidando così competenze e capacità operative in diversi settori e mercati.

Il portafoglio di servizi è ampio e integrato, con *Data & AI* come filo conduttore e aree principali che comprendono:

- *Cloud & Infrastructure*, con soluzioni ibride e *on-premise*, sicurezza in ambienti *cloud*, migrazione e monitoraggio;
- *Software Development*, che spazia dallo sviluppo *agile* e *mobile* alla *system integration*, con particolare attenzione alla qualità e all'automazione dei *test*;
- *Cybersecurity*, con servizi di consulenza, *audit*, architetture sicure, *managed services* e sistemi antifrode;
- *Data & AI*, che include *data integration*, *data governance*, *analytics*, *machine learning*, *synthetic data*, *forecasting* e soluzioni Environmental, Social and Governance (ESG)[g].

Kirey Group pone grande attenzione alla sostenibilità, alla trasparenza e all'integrità, adottando pratiche responsabili nei confronti di clienti, partner, dipendenti e *stakeholder*. L'azienda è inoltre attivamente impegnata in progetti sociali, promuove la diversità e l'inclusione, e investe nello sviluppo delle competenze tecnologiche e professionali dei propri collaboratori.

Oggi il gruppo conta oltre 1370 casi di business realizzati, 10 *Innovation Center* attivi, un fatturato di circa 126 milioni di euro e più di 1500 collaboratori distribuiti in 10 paesi.

1.2 L'idea

L'idea alla base dello stage consiste nello sviluppo di una piattaforma per il monitoraggio intelligente del traffico utente di un *e-commerce*. L'obiettivo principale è quello di sfruttare algoritmi di Intelligenza Artificiale e di *Machine Learning* per individuare e segnalare automaticamente eventuali anomalie nei dati raccolti.

La piattaforma sarà in grado di analizzare i flussi in tempo reale, rilevando accessi sospetti, rallentamenti e potenziali minacce, così da consentire interventi tempestivi e garantire sia la sicurezza sia le prestazioni ottimali del sistema.

Il progetto è stato organizzato in quattro fasi principali:

- Una prima fase di formazione e preparazione dell'ambiente di lavoro, utile a familiarizzare con le tecnologie e i prodotti utilizzati, che comprende l'individuazione, l'installazione e la configurazione delle componenti necessarie, con successiva verifica della connettività tra i moduli;
- Una fase di analisi e progettazione, in cui saranno definite le specifiche funzionali e la soluzione tecnica tramite la configurazione degli agenti di raccolta, la realizzazione di *pipeline* di *log* e la loro indicizzazione in *Elasticsearch*;
- Una fase di realizzazione e test della piattaforma, con sviluppo di *script* per il monitoraggio sintetico, analisi su *Elasticsearch*, creazione di *dashboard* in *Kibana* e configurazione di regole di *alerting*;
- Infine la stesura della documentazione tecnica e funzionale, contenente la descrizione delle tecnologie adottate, dei flussi logici implementati e delle criticità riscontrate.

Per lo sviluppo saranno impiegati linguaggi come *Python* e *Java*, sistemi *Linux* e i prodotti della suite *Elastic Stack*, strumenti particolarmente adatti per l'elaborazione e il monitoraggio di grandi volumi di dati in tempo reale.

1.3 Organizzazione del testo

Il secondo capitolo approfondisce il progetto di stage, descrivendo gli obiettivi e le attività svolte, la metodologia di lavoro adottata e un'analisi preventiva dei principali rischi e relative strategie di mitigazione;

Il terzo capitolo è dedicato all'analisi dei requisiti della piattaforma e alle motivazioni che ne hanno guidato le scelte progettuali, fornendo al contempo una mappatura completa e strutturata dei requisiti individuati per il sistema;

Il quarto capitolo approfondisce le tecnologie e i principi teorici alla base della soluzione proposta, analizzando i principali strumenti e approcci per il monitoraggio delle prestazioni applicative;

Il quinto capitolo approfondisce ...

Nel sesto capitolo viene descritta ...

1.3.1 Convenzioni tipografiche

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

- gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
- per la prima occorrenza dei termini riportati nel glossario viene utilizzata la seguente nomenclatura: *parola*[g];
- i termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*.

Capitolo 2

Descrizione dello stage

Il capitolo approfondisce il progetto di stage, descrivendo gli obiettivi e le attività svolte, la metodologia di lavoro adottata e un'analisi preventiva dei principali rischi e relative strategie di mitigazione.

2.1 Introduzione al progetto

Lo stage è stato svolto presso l'azienda Kirey Group S.r.l., realtà consolidata nell'ambito della fornitura di prodotti e servizi informatici, con clienti internazionali e una forte specializzazione nei settori *Banking*, *Insurance*, *Oil&Gas* e Pubblica Amministrazione. L'attività si è inserita nel contesto del team Application Performance Monitoring (APM)[g]e ha avuto come obiettivo principale la realizzazione e il collaudo di una piattaforma per il monitoraggio delle performance di una *web application*.
Il progetto è stato sviluppato interamente in ambiente *Linux* mediante *Windows Subsystem for Linux*, utilizzando la *suite Elastic Stack* e i suoi principali componenti: *Elasticsearch* per la gestione dei dati, *Kibana* per la visualizzazione, *Logstash* per l'ingestione e la trasformazione dei log, *Beats* e *Fleet Server* per la raccolta distribuita delle metriche e *APM Server/Agent* per il tracciamento delle *performance* applicative. Sono stati realizzati sviluppi in *Python* e *Java* per l'estensione delle funzionalità e l'integrazione di algoritmi di Artificial Intelligence (AI)[g]e *Machine Learning*, con l'obiettivo di rilevare automaticamente anomalie e problematiche di prestazione.
La finalità complessiva è quella di fornire un sistema scalabile, proattivo e ben documentato, capace di garantire prestazioni ottimali e un monitoraggio continuo della *web application*.

2.2 Pianificazione

Tutte le attività sono state condotte in affiancamento ad un tutor aziendale che ha curato sia la parte di formazione che di indirizzamento delle attività. A tal fine sono stati svolti dei momenti di confronto settimanali per la valutazione dello stato di avanzamento delle attività e momenti quotidiani di confronto sulle problematiche riscontrate.
Le attività proposte sono state collocate all'interno di un progetto più ampio portato avanti in Kirey da un team di persone eterogeneo.

Al termine dello stage sono stati presentati i risultati ottenuti a tutto il team. L'infrastruttura tecnologica e le piattaforme su cui girerà l'applicazione sono state messe a disposizione da Kirey.

2.3 Analisi preventiva dei rischi

Durante la fase di analisi iniziale sono stati individuati alcuni possibili rischi a cui si potrà andare incontro. Si è quindi proceduto a elaborare delle possibili soluzioni per far fronte a tali rischi.

1. Inesperienza nella suite Elastic

Descrizione: La limitata esperienza iniziale nell'utilizzo della *Elastic Stack* (*Elasticsearch*, *Kibana*, *Logstash*, *APM*) potrebbe comportare difficoltà nella configurazione e nell'integrazione delle componenti, rallentando lo sviluppo del progetto.

Impatto: Alto.

Probabilità: Alta.

Soluzione: Organizzazione di momenti di confronto con il *tutor* aziendale e studio personale della documentazione, al fine di acquisire le competenze necessarie.

2. Integrazione tra componenti Elastic

Descrizione: La comunicazione tra i diversi moduli della *suite Elastic* (*Elasticsearch*, *Kibana*, *Logstash*, *APM*) potrebbe presentare problemi di configurazione, causando ritardi o malfunzionamenti nell'acquisizione dei dati.

Impatto: Medio.

Probabilità: Media.

Soluzione: Esecuzione di test di connettività e validazione progressiva delle *pipeline*, con il supporto del *tutor* aziendale per la risoluzione dei problemi.

3. Qualità e coerenza dei dati raccolti

Descrizione: I dati acquisiti dagli agenti potrebbero risultare incompleti, duplicati o non coerenti, compromettendo le analisi e le *dashboard*.

Impatto: Alto.

Probabilità: Media.

Soluzione: Definizione di regole di filtraggio e validazione all'interno delle *pipeline Logstash* ed esecuzione di test di integrità sugli indici *Elasticsearch*.

4. Scalabilità e carico del sistema

Descrizione: L'aumento del volume dei dati e delle richieste potrebbe impattare sulle prestazioni della piattaforma, riducendo l'efficienza del monitoraggio.

Impatto: Medio.

Probabilità: Media.

Soluzione: Implementazione di strategie di *scaling* orizzontale e utilizzo di metriche di *Kibana* per monitorare l'impatto del carico in tempo reale.

5. Implementazione di algoritmi di Machine Learning

Descrizione: L'integrazione di modelli di *Machine Learning* per il rilevamento delle anomalie potrebbe richiedere competenze specifiche e tempi di sviluppo più lunghi del previsto.

Impatto: Alto.

Probabilità: Media.

Soluzione: Formazione preliminare su tecniche di *Machine Learning* e utilizzo di librerie e *framework* consolidati per accelerare lo sviluppo.

6. Problemi di configurazione delle pipeline Logstash

Descrizione: Errori di configurazione nelle *pipeline* di *Logstash* potrebbero causare la perdita, la duplicazione o la trasformazione errata dei dati raccolti, compromettendo l'affidabilità delle analisi.

Impatto: Alto.

Probabilità: Media.

Soluzione: Adozione di un approccio con *test* di validazione a ogni modifica delle *pipeline* e utilizzo di ambienti di prova per verificare la correttezza del flusso dei dati prima della messa in produzione.

7. Accesso limitato a funzionalità premium di Elastic

Descrizione: Durante le prime settimane di lavoro in locale, la mancata possibilità di operare su *Elastic Cloud* e di accedere a funzionalità premium come il *Machine Learning* potrebbe limitare l'analisi dei dati e rallentare la validazione di alcune funzionalità previste dal progetto.

Impatto: Medio.

Probabilità: Bassa.

Soluzione: Esecuzione preventiva delle attività in locale con dati di test, studio della documentazione sulle funzionalità premium e pianificazione di un passaggio successivo a *Elastic Cloud* non appena disponibile, con supporto del tutor aziendale.

2.4 Requisiti e obiettivi

Gli obiettivi del progetto sono stati classificati in base alla loro priorità secondo le seguenti notazioni:

- **Ob** (Requisiti Obbligatori) - requisiti essenziali e imprescindibili per il successo del progetto, vincolanti in quanto obiettivo primario richiesto dal committente;
- **D** (Requisiti Desiderabili) - requisiti importanti ma non critici, la cui assenza non compromette il progetto, non vincolanti o strettamente necessari, ma dal riconoscibile valore aggiunto;

- **Op** (Requisiti Opzionali) - requisiti desiderabili ma non essenziali, rappresentanti valore aggiunto non strettamente competitivo.

Durante il periodo di stage si prevede il raggiungimento dei seguenti obiettivi:

- **Preparazione dell'ambiente per l'implementazione della soluzione:**
  - **Ob1.1:** Individuazione delle componenti ed eventuali librerie da utilizzare;
  - **Ob1.2:** Installazione e configurazione delle componenti;
  - **Ob1.3:** Verifica del corretto funzionamento dell'ambiente e test di connettività tra componenti.
- **Implementazione estrazione dati dalla web application:**
  - **Ob2.1:** Configurazione *agent* (*Beats/APM*) per la raccolta dati della navigazione;
  - **Ob2.2:** Implementazione *pipeline* di *log* tramite *Logstash* per filtraggio e inoltro dati in *Elasticsearch*;
  - **Ob2.3:** Verifica della corretta acquisizione dei dati e loro indicizzazione in *Elasticsearch*.
- **Implementazione elaborazione dati e rappresentazione grafica dei dati:**
  - **Ob3.1:** Sviluppo *script Python/Java* per il monitoraggio sintetico (*Selenium*) e generazione di traffico *log*;
  - **Ob3.2:** Analisi e aggregazione dati in *Elasticsearch*, con *query* e visualizzazioni preliminari;
  - **Ob3.3:** Creazione dashboard avanzate su *Kibana* con metriche di *performance*, accesso e flussi utente;
  - **Op3.4:** Configurazione regole di *alerting* e notifiche per anomalie rilevate in tempo reale.
- **Documentazione dettagliata:**
  - **Ob4.1:** Descrizione delle tecnologie e prodotti utilizzati;
  - **Ob4.2:** Descrizione dei flussi logici del progetto e delle funzionalità dell'applicazione;
  - **D4.3:** Pro/contro di ogni componente e criticità nell'applicazione.

Nel capitolo successivo verrà approfondita l'analisi dei requisiti identificati in questa sezione, al fine di definirne in modo più strutturato la natura e le caratteristiche.

Capitolo 3

Analisi dei requisiti

Il capitolo è dedicato all'analisi dei requisiti della piattaforma, con l'obiettivo di fornire una visione completa e dettagliata delle funzionalità e delle caratteristiche attese dal sistema. Verranno illustrate le esigenze degli utenti e del contesto operativo, evidenziando le specifiche tecniche e le funzionalità che hanno guidato le scelte progettuali.

3.1 Requisiti del sistema

A seguito di un'attenta attività di analisi del progetto e degli obiettivi tecnici e funzionali prefissati, sono state redatte le tabelle di tracciamento che riassumono in modo strutturato i requisiti individuati.
Durante questa fase sono state identificate differenti tipologie di requisiti, distinte sia in base alla loro categoria (funzionale, non funzionale, qualitativo o di vincolo), sia in base alla loro priorità di implementazione (obbligatorio, desiderabile o opzionale). Per garantire una tracciabilità chiara e univoca, a ciascun requisito è stato assegnato un codice identificativo composto da lettere che ne descrivono la tipologia e l'importanza, secondo la seguente convenzione:

R = Requisito

F = Funzionale

N = Non funzionale

Q = Qualitativo

V = Vincolo

O = Obbligatorio

D = Desiderabile

Z = Opzionale

Ogni requisito analizzato sarà identificato univocamente dalla seguente notazione:

R[Priorità][Categoria]-[Numero]

Dove:

- **Priorità** indica la priorità di implementazione, che può essere O = Obbligatorio, D = Desiderabile, Z = Opzionale;
- **Categoria** la categoria di appartenenza, che può essere F = Funzionale, N = Non funzionale, Q = Qualitativo, V = Vincolo;
- **Numero** un numero progressivo che identifica in modo univoco il requisito all'interno della sua categoria.

Nelle tabelle 3.1, 3.2, 3.3 e 3.4 sono riportati in modo sintetico tutti i requisiti emersi dall'analisi, classificati in base alla loro priorità e accompagnati da una breve descrizione della relativa funzionalità o vincolo tecnico.

3.1.1 Requisiti funzionali

I requisiti funzionali descrivono cosa deve fare il sistema. Sono le funzionalità concrete che la soluzione deve offrire per raggiungere gli obiettivi del progetto.

Tabella 3.1: Tabella del tracciamento dei requisiti funzionali

Codice	Descrizione	Classificazione
ROF-1	Il sistema deve permettere la raccolta automatica di metriche e *log* relativi alla *web application* tramite agenti *OpenTelemetry* o *Elastic APM*.	Obbligatorio
ROF-2	Il sistema deve inviare i dati raccolti agli *endpoint* *Elasticsearch* per l'analisi e l'indicizzazione.	Obbligatorio
ROF-3	Il sistema deve consentire la creazione di *pipeline* di *log* tramite *Logstash* per filtraggio, trasformazione e inoltro dei dati in *Elasticsearch*.	Obbligatorio
ROF-4	Il sistema deve prevedere la configurazione di *Elastic Agents* (*Beats*/*APM*) per la raccolta dati della navigazione.	Obbligatorio
ROF-5	Il sistema deve generare *dashboard* avanzate e visualizzazioni in *Kibana*, con metriche di *performance*, accesso e flussi utente.	Obbligatorio
ROF-6	Il sistema deve permettere la verifica della corretta acquisizione dei dati e la loro indicizzazione in *Elasticsearch*.	Obbligatorio
ROF-7	Il sistema deve prevedere lo sviluppo e l'esecuzione di *script* automatizzati in *Python* o *Java* per la simulazione del traffico utente (*Synthetic Monitoring*).	Obbligatorio
ROF-8	Deve essere possibile filtrare e ricercare i *log* per *host*, servizio, livello di severità o periodo temporale.	Obbligatorio
RDF-9	Il sistema dovrebbe prevedere la configurazione di regole di *alerting* e notifiche in tempo reale per anomalie rilevate.	Desiderabile
RDF-10	Il sistema dovrebbe integrare algoritmi di *Machine Learning* per l'individuazione automatica di anomalie.	Desiderabile
RDF-11	Il sistema dovrebbe consentire l'esportazione delle *dashboard* o dei risultati delle *query* in formato PDF[g] o CSV[g].	Desiderabile
RZF-12	Il sistema può prevedere un modulo aggiuntivo per la generazione automatica di report periodici delle metriche raccolte.	Opzionale
RZF-13	Il sistema può consentire l'importazione automatica delle configurazioni APM da ambienti di *test* o *staging*.	Opzionale

3.1.2 Requisiti non funzionali

I requisiti non funzionali definiscono come il sistema deve comportarsi, cioè le sue proprietà di qualità interna. Non aggiungono nuove funzioni, ma impongono vincoli di prestazioni, sicurezza, disponibilità, scalabilità, affidabilità e manutenibilità.

Tabella 3.2: Tabella del tracciamento dei requisiti non funzionali

Codice	Descrizione	Classificazione
RON-1	Il sistema deve essere scalabile e consentire l'aggiunta di nuove fonti di dati o agenti senza compromettere la stabilità.	- Obbligatorio
RON-2	Il sistema deve garantire l'affidabilità nella trasmissione e nella conservazione dei dati raccolti.	- Obbligatorio
RON-3	La piattaforma deve assicurare un tempo di latenza accettabile nella visualizzazione dei dati ($<$ 5 secondi per l'aggiornamento delle *dashboard*).	- Obbligatorio
RDN-4	Il sistema dovrebbe garantire la possibilità di eseguire *test* di carico e stress per valutare la stabilità dell'ambiente.	- Desiderabile
RDN-5	Il sistema dovrebbe supportare l'autenticazione per la gestione degli accessi a *Kibana*.	- Desiderabile

3.1.3 Requisiti qualitativi

I requisiti qualitativi specificano le proprietà qualitative che influenzano l'esperienza d'uso e la manutenibilità. Si concentrano su aspetti percepibili, come semplicità, chiarezza, flessibilità o estendibilità.

Tabella 3.3: Tabella del tracciamento dei requisiti qualitativi

Codice	Descrizione	Use Case
ROQ-1	L'interfaccia di *Kibana* deve offrire una rappresentazione chiara e intuitiva delle metriche principali.	- Obbligatorio
ROQ-2	I dati devono essere visualizzabili in forma aggregata e filtrabile in base a intervalli temporali e categorie di evento.	- Obbligatorio
ROQ-3	Le *dashboard* devono presentare una chiara distinzione cromatica tra metriche positive, neutre e anomale.	- Obbligatorio
RDQ-4	Le *dashboard* dovrebbero essere personalizzabili dall'utente secondo criteri di interesse (*performance*, accessi, flussi).	- Desiderabile
RZQ-5	Il sistema può includere un *layout dark/light mode* o temi grafici personalizzati per una migliore leggibilità.	- Opzionale

3.1.4 Requisiti di vincolo

Impongono limitazioni o condizioni esterne al progetto: ambienti, tecnologie, compatibilità, strumenti, standard aziendali o legali.

Tabella 3.4: Tabella del tracciamento dei requisiti di vincolo

Codice	Descrizione	Use Case
ROV-1	Il sistema deve utilizzare i prodotti della suite *Elastic Stack* (*Elasticsearch, Logstash, Kibana, Beats, APM Server/Agent*).	- Obbligatorio
ROV-2	L'ambiente operativo deve essere *Linux* (*Red Hat* o distribuzioni equivalenti).	- Obbligatorio
ROV-3	Tutti i componenti *software* devono essere compatibili con la versione di *Linux* installata sull'ambiente aziendale (es. *Ubuntu 22.04 LTS* o *Red Hat 9*).	- Obbligatorio
ROV-4	Le componenti devono rispettare le seguenti versioni minime: • *Python* >= 3.10 • *Java* >= 16 • *Node.js* >= 17 • *Logstash* >= 8.10 • *Kibana* >= 8.10 • *Beats* >= 8.10 • *APM Server* >= 8.10 • *Elasticsearch* >= 8.10 • *OpenTelemetry* >= 1.0	- Obbligatorio
ROV-5	La *web application* deve essere compatibile con i principali *browser* (*Chrome* >= 120, *Firefox* >= 115, *Edge* >= 120, *Apple Safari* >= 15).	- Obbligatorio
ROV-6	Tutte le configurazioni devono essere eseguite in un ambiente *Docker* o su infrastruttura fornita da Kirey Group.	- Obbligatorio
RDV-7	La documentazione tecnica deve essere redatta in formato *Markdown*, *LaTeX* o PDF, seguendo gli standard aziendali.	- Desiderabile
RDV-8	Il sistema dovrebbe consentire la distribuzione automatizzata tramite *Docker Compose*.	- Desiderabile

3.2 Riepilogo dei requisiti

Tabella 3.5: Riepilogo dei requisiti

Tipologia	Obbligatorio	Desiderabile	Opzionale	Totale
Funzionali	8	3	2	13
Non funzionali	3	2	0	5
Qualitativi	3	1	1	5
Di Vincolo	6	2	0	8
Totale				31

3.3 Rappresentazione architetturale preliminare

La rappresentazione architetturale descrive la struttura della soluzione proposta, evidenziando le principali componenti coinvolte nel sistema di APM e le loro relazioni. Essa costituisce il collegamento tra la fase di analisi dei requisiti e la successiva implementazione pratica. I dettagli tecnici e i diagrammi della soluzione verranno approfonditi nel capitolo dedicato alla progettazione architetturale.

3.3.1 Struttura generale della soluzione di monitoraggio

La soluzione di APM proposta si fonda su un'architettura ibrida che combina le funzionalità di *OpenTelemetry* con la potenza analitica dello *Stack Elastic*, fornendo un sistema di osservabilità completo.
L'applicazione di riferimento *PetClinic*, viene monitorata tramite l'*OpenTelemetry Java Agent*, avviato contestualmente all'applicazione mediante comandi *-Dotel* nello script di avvio. Questo agente intercetta automaticamente le chiamate HTTP, le transazioni e le operazioni sul database, generando metriche e tracce relative al comportamento del *backend*.
I dati raccolti vengono inviati all'*APM Server*, gestito da *Fleet* all'interno dell'*Elastic Agent*. L'*APM Server* riceve, elabora e normalizza le informazioni provenienti dall'agente *OpenTelemetry*, rendendole compatibili con *Elasticsearch*.
In contemporanea, il monitoraggio del *frontend* è gestito dal *RUM Agent JavaScript*, che traccia le interazioni dell'utente, i tempi di caricamento delle pagine e gli eventi di errore nel *browser*.
Tutti i dati vengono centralizzati in *Elasticsearch*, dove sono indicizzati e resi disponibili per la consultazione tramite *Kibana*.

3.3.2 Integrazione dei componenti principali

L'architettura della soluzione di monitoraggio si compone dei seguenti moduli principali:

- **Elasticsearch:** motore di ricerca e analisi distribuito che gestisce la memorizzazione, l'indicizzazione e l'interrogazione dei dati provenienti dagli agenti di monitoraggio in tempo reale;
- **Kibana:** interfaccia di visualizzazione e analisi integrata con *Elasticsearch*, consente di visualizzare metriche e tracce applicative, creare *dashboard* personalizzate e configurare regole di *alerting* per il monitoraggio del sistema;
- **Fleet:** componente centralizzato per la gestione e la configurazione degli agenti *Elastic*. Attraverso il *Fleet Server*, coordina la comunicazione con gli *Elastic Agents*;
- **Elastic Agent:** agente unificato che integra funzionalità di raccolta dati, sicurezza e monitoraggio. Gestito tramite *Fleet*, può essere configurato per includere moduli specifici come *APM Server* e *Beats* per la raccolta di metriche di sistema e *log*;
- **APM Server:** incluso all'interno dell'*Elastic Agent*, riceve i dati di telemetria provenienti dagli agenti applicativi. Supporta lo standard *OpenTelemetry*, fungendo da *endpoint* compatibile per la ricezione dei dati raccolti dal *Java Agent*;

- **OpenTelemetry Java Agent:** agente di strumentazione automatica avviato con l'applicazione *PetClinic*, utilizzato per raccogliere metriche e tracce delle transazioni lato *backend* (chiamate HTTP, query al *database*, eccezioni, tempi di risposta). I dati generati vengono inviati all'*APM Server* attraverso il protocollo OpenTelemetry Protocol (OTLP)[g];
- **RUM Agent JavaScript:** agente di monitoraggio RUM eseguito nel *frontend* dell'applicazione, che raccoglie dati sulle interazioni utente, sui tempi di caricamento delle pagine e sulle prestazioni percepite nel *browser*;

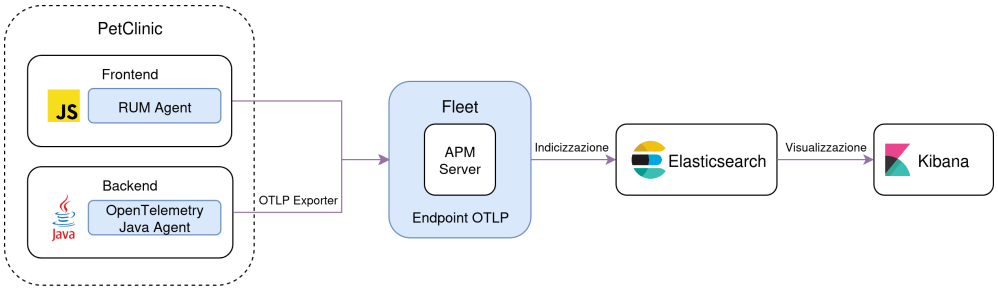


Figura 3.1: Figura 3.1 - Integrazione tra componenti del sistema di monitoraggio

3.3.3 Applicazione di riferimento: Spring PetClinic

Per la sperimentazione del sistema di APM è stata utilizzata come applicazione di riferimento *Spring PetClinic*, una *web application open source* sviluppata in *Java* e basata sul *framework Spring Boot*. La scelta di *PetClinic* è motivata dalla sua architettura chiara e ben documentata, composta da un livello di presentazione (*controller* e *template*), un livello logico (servizi) e un livello di persistenza (*repository* e *database* MySQL[g]). Tale struttura consente di osservare comportamenti applicativi realistici, come chiamate HTTP[g], interazioni con il *database* e logiche di *business*, fornendo un contesto ideale per testare le funzionalità di monitoraggio offerte da *Elastic APM* e *OpenTelemetry*.

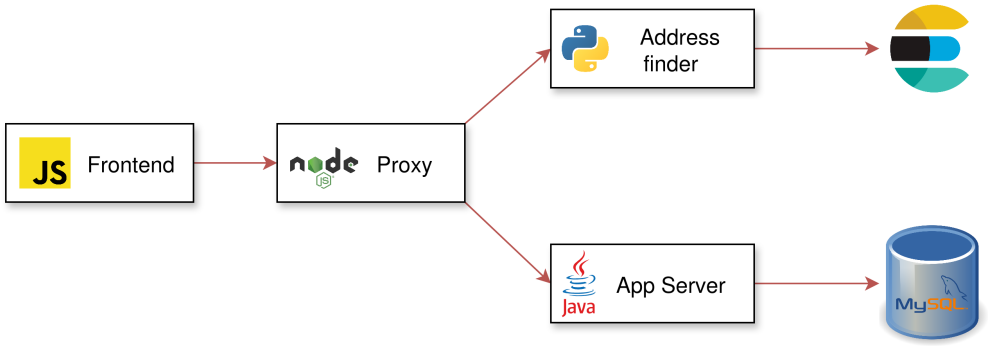


Figura 3.2: Figura 3.2 - Architettura di Spring PetClinic

L'applicazione *PetClinic* verrà quindi utilizzata come caso di studio per la validazione della soluzione di monitoraggio proposta, analizzando successivamente il modo in cui i diversi moduli architetturali interagiscono con essa.

3.4 User Stories e scenari di utilizzo

Le seguenti *user stories* descrivono brevemente i principali scenari di utilizzo del sistema di APM, evidenziando gli obiettivi e le esigenze degli utenti tecnici aziendali che operano sulla piattaforma di monitoraggio. L'unico attore considerato è l'*Observability Engineer*, figura responsabile della configurazione, analisi e manutenzione del sistema all'interno dell'azienda.

Tabella 3.6: User Stories

ID	User Story	Obiettivo
US1	Come *Observability Engineer*, voglio analizzare i tempi medi di risposta degli *endpoint* dell'applicazione per individuare colli di bottiglia o degradi di *performance*.	Ottimizzare le prestazioni applicative.
US2	Come *Observability Engineer*, voglio monitorare l'utilizzo di CPU, memoria e connessioni per valutare la stabilità del sistema.	Garantire la corretta gestione delle risorse.
US3	Come *Observability Engineer*, voglio rilevare e classificare errori e anomalie dai *log* per identificare rapidamente le cause dei malfunzionamenti.	Migliorare l'affidabilità del sistema.
US4	Come *Observability Engineer*, voglio configurare regole di *alerting* in *Kibana* per essere notificato in caso di anomalie.	Ridurre i tempi di risposta agli incidenti.
US5	Come *Observability Engineer*, voglio visualizzare le metriche aggregate in *dashboard* personalizzate.	Facilitare l'analisi e la comunicazione dei risultati.
US6	Come *Observability Engineer*, voglio visualizzare in *Kibana* le metriche di prestazione del *backend* (tempi medi di risposta, *throughput*, tasso di errore, durata media delle transazioni) per valutare lo stato di salute dell'applicazione.	Monitorare le prestazioni e la stabilità del servizio.
US7	Come *Observability Engineer*, voglio visualizzare metriche di utilizzo delle risorse di sistema (CPU, memoria, disco, rete) per individuare potenziali colli di bottiglia.	Ottimizzare l'efficienza del sistema.
US8	Come *Observability Engineer*, voglio analizzare in *Kibana* le metriche relative al comportamento degli utenti (*page load time*, errori *JavaScript*, interazioni lente) raccolte dal *RUM Agent*.	Migliorare l'esperienza utente.
US9	Come *Observability Engineer*, voglio correlare in un'unica *dashboard* le metriche di infrastruttura, *application performance* e *user experience*, per ottenere una visione completa del sistema.	Favorire un'analisi integrata e diretta delle anomalie.
US10	Come *Observability Engineer*, voglio integrare l'*OpenTelemetry Java Agent* e il *RUM Agent JavaScript* per raccogliere rispettivamente metriche di *backend* e *frontend*.	Avere una visione completa del comportamento dell'applicazione.

US11	Come *Observability Engineer*, voglio configurare e gestire gli agenti tramite *Fleet* per garantire una distribuzione e un aggiornamento centralizzati.	Semplificare la manutenzione e ridurre gli errori di configurazione.
US12	Come *Observability Engineer*, voglio eseguire *test* sintetici periodici per monitorare la disponibilità e i tempi di risposta delle pagine.	Garantire la continuità operativa.
US13	Come *Observability Engineer*, voglio esportare le *dashboard* in formato PDF o CSV per la condivisione all'interno del team.	Documentare e condividere le analisi in modo efficace.
US14	Come *Observability Engineer*, voglio ricevere notifiche automatiche via email quando vengono superate determinate soglie di prestazione.	Automatizzare la gestione degli incidenti e migliorare il tempo di reazione.
US15	Come *Observability Engineer*, voglio configurare in *Kibana job* di *Machine Learning* per l'individuazione automatica di anomalie.	Rilevare comportamenti anomali senza intervento manuale.
US16	Come *Observability Engineer*, voglio visualizzare in *Kibana* i risultati dei *job* di *Machine Learning* (anomalie e previsioni) integrati nelle *dashboard*.	Integrare l'analisi automatica con la visualizzazione interattiva dei dati.

Capitolo 4

Tecnologie e principi teorici

Il capitolo presenta il quadro teorico e tecnologico di riferimento del progetto, descrivendo i principali approcci e strumenti per il monitoraggio delle prestazioni applicative. Vengono illustrate le tecnologie analizzate e le motivazioni che hanno guidato la scelta della soluzione, in relazione ai requisiti e agli obiettivi del sistema di osservabilità sviluppato.

4.1 APM e Observability

L'osservabilità di un sistema si basa sull'analisi congiunta di tre pilastri fondamentali: metriche, log e tracce.

- **Logs**: un log è un record testuale di un evento ad un orario specifico, come un tentativo di accesso, un errore di sistema o una transazione completata. I log forniscono dettagli di contesto che aiutano a diagnosticare problemi specifici;
- **Metriche**: una metrica è una misurazione numerica di un aspetto specifico delle prestazioni del sistema, come l'utilizzo della CPU, la latenza delle richieste o il numero di utenti attivi;
- **Tracce**: una traccia rappresenta il percorso di una singola richiesta attraverso i vari componenti di un sistema distribuito, consentendo di visualizzare il flusso delle operazioni e identificare i colli di bottiglia.

Un sistema APM moderno integra questi aspetti per fornire una visione completa dello stato e del comportamento applicativo. Nel contesto del monitoraggio di applicazioni *web* come *PetClinic*, l'osservabilità consente di analizzare i dati di performance raccolti dal sistema di APM per identificare e risolvere problemi, ottimizzare le prestazioni e migliorare l'esperienza utente complessiva.

4.2 Approcci e tecnologie per il monitoraggio

Nel mondo dell'*Application Performance Monitoring* esistono diversi approcci e tecnologie per raccogliere, analizzare e visualizzare i dati di telemetria.
Il monitoraggio delle prestazioni può essere realizzato mediante diverse strategie, che si distinguono per tipo di dati raccolti, modalità di raccolta e grado di integrazione con l'applicazione.

4.2.1 Approcci principali

Gli approcci al monitoraggio delle applicazioni possono essere classificati in base alla modalità di raccolta dei dati e al livello di osservabilità fornito.
Tra i principali si distinguono:

- **Monitoraggio basato sugli agenti (Agent-based monitoring):** prevede l'integrazione di componenti software, detti *agent*, all'interno dell'applicazione o dell'infrastruttura. Questi raccolgono metriche, log e tracce in tempo reale, offrendo una visione complessiva del sistema. È il modello adottato da strumenti come *Elastic APM*, *Datadog* e *New Relic*;
- **Monitoraggio senza agenti (Agentless monitoring):** in questo caso la raccolta dei dati avviene tramite l'analisi di log o metriche esposte da servizi esterni, senza modificare il codice dell'applicazione. Sebbene riduca l'invasività, questo approccio offre un livello di dettaglio inferiore rispetto ai sistemi *agent-based*;
- **Distributed tracing:** metodo che consente di tracciare l'intero ciclo di vita di una richiesta distribuita tra più servizi o microservizi, associando a ciascun evento un identificativo univoco;
- **Synthetic monitoring:** utilizza richieste simulate e test automatizzati per verificare la disponibilità e le prestazioni delle applicazioni da diverse località geografiche. È utile per individuare problemi prima che impattino sugli utenti reali dell'applicazione;
- **Real User Monitoring (RUM):** misura le prestazioni dal punto di vista dell'utente reale, analizzando tempi di caricamento, interazioni e metriche di esperienza. Combinato con il monitoraggio sintetico, fornisce una visione completa della user experience indicata con il termine End User Monitoring (EUM)[g].

Nel corso del monitoraggio di *PetClinic*, l'approccio adottato integra principalmente il monitoraggio basato sugli agenti, il *distributed tracing* e il *Real User Monitoring*, al fine di ottenere una visione dettagliata e completa delle prestazioni dell'applicazione.

4.2.2 Tecnologie e piattaforme note

Negli ultimi anni si sono affermate diverse piattaforme e *framework* dedicati al monitoraggio e all'osservabilità, che adottano architetture e modelli di raccolta dati differenti.
Tra le più rilevanti si trovano:

- **Elastic Stack (ELK):** una delle soluzioni *open source* più diffuse, integra *Elasticsearch*, *Logstash* e *Kibana*, estesa con *Elastic APM* per il tracciamento delle prestazioni. Offre un ecosistema unificato per metriche, log e tracce;
- **OpenTelemetry:** standard *open source* promosso dalla *Cloud Native Computing Foundation (CNCF)* per la raccolta e l'esportazione di dati di telemetria. Fornisce Software Development Kit (SDK)[g]e agenti per numerosi linguaggi di programmazione, garantendo interoperabilità tra sistemi di osservabilità differenti;
- **Prometheus e Grafana:** soluzione *open source* focalizzata sulle metriche. *Prometheus* raccoglie e memorizza dati temporali, mentre *Grafana* li visualizza in *dashboard* personalizzabili. È molto usata in ambienti *cloud-native* e *Kubernetes*;

- **Datadog, New Relic, Dynatrace:** piattaforme commerciali che offrono funzionalità avanzate di APM, monitoraggio dell'infrastruttura e analisi basata su *machine learning.*;

Le soluzioni open source, come *Elastic Stack* e *OpenTelemetry*, offrono maggiore flessibilità e possibilità di personalizzazione, rendendole particolarmente adatte per ambienti di sviluppo e sperimentazione.
Nel contesto del progetto di monitoraggio *PetClinic*, l'attenzione si è concentrata sull'integrazione tra l'*Elastic Stack* e *OpenTelemetry*, con l'obiettivo di realizzare una soluzione di monitoraggio estendibile e compatibile con l'infrastruttura aziendale.

4.3 Tecnologie e strumenti utilizzati

Di seguito viene data una panoramica delle tecnologie e degli strumenti utilizzati.

4.3.1 Elastic Stack

Elasticsearch

Elasticsearch è un motore di ricerca e analisi distribuito basato su *Apache Lucene*, progettato per gestire grandi volumi di dati in tempo reale. Fornisce funzionalità avanzate di ricerca *full-text*, analisi dei dati e aggregazioni, rendendolo ideale per applicazioni di monitoraggio, analisi dei *log* e *business intelligence*. Elasticsearch supporta un'architettura scalabile e flessibile, consentendo di distribuire i dati su più nodi e cluster per garantire alta disponibilità e prestazioni elevate.

Figura 4.1: Figura 4.1 - Elasticsearch

Kibana

Kibana è uno strumento di visualizzazione e analisi dei dati open source, progettato per lavorare in stretta integrazione con Elasticsearch. Fornisce un'interfaccia utente intuitiva che consente agli utenti di creare *dashboard* interattive, visualizzare grafici, mappe e tabelle, e analizzare i dati in tempo reale. Kibana supporta una vasta gamma di visualizzazioni personalizzabili e offre funzionalità avanzate come il filtraggio dei dati, la ricerca *full-text* e l'esplorazione delle relazioni tra i dati, rendendolo uno strumento potente per l'analisi dei *log*, il monitoraggio delle prestazioni e la *business intelligence*.

Figura 4.2: Figura 4.2 - Kibana

Elastic Agent e Fleet

Elastic Agent è un agente unificato sviluppato da Elastic che consente di raccogliere, monitorare e proteggere i dati da diverse fonti in modo semplice ed efficiente. Integrando funzionalità di raccolta dati, sicurezza e monitoraggio, Elastic Agent semplifica la gestione degli agenti e riduce la complessità operativa. Fleet è una funzionalità di gestione centralizzata all'interno di Kibana che consente di distribuire, configurare e monitorare gli Elastic Agent in modo scalabile e automatizzato. Attraverso Fleet, gli utenti possono gestire facilmente le policy di raccolta dati, aggiornare gli agenti e monitorare lo stato della loro infrastruttura da un'unica interfaccia.

OpenTelemetry

Figura 4.3: Figura 4.3 - OpenTelemetry

Elastic APM e APM Server

Figura 4.4: Figura 4.4 - Elastic APM

RUM Agent e Synthetic Monitoring

MCP

4.3.2 Linguaggi di programmazione

Python

Figura 4.5: Figura 4.5 - Python

JavaScript

Figura 4.6: Figura 4.6 - JavaScript

Java

Figura 4.7: Figura 4.7 - Java

4.3.3 Framework e librerie

Node.js

Figura 4.8: Figura 4.8 - Node.js

Spring Boot

Figura 4.9: Figura 4.9 - Spring

Selenium

Figura 4.10: Figura 4.10 - Selenium

4.3.4 Strumenti di sviluppo

VSCode

Figura 4.11: Figura 4.11 - VSCode

4.3.5 Database

MySQL

Figura 4.12: Figura 4.12 - MySQL

4.4 Criteri di scelta della soluzione

Le principali motivazioni che hanno guidato la scelta della soluzione di monitoraggio basata su *Elastic Stack* e *OpenTelemetry* sono:

- **Open source e flessibilità:** entrambe le tecnologie sono open source, consentendo una maggiore personalizzazione e adattabilità alle esigenze specifiche del progetto;
- **Scalabilità:** la soluzione è progettata per gestire grandi volumi di dati in ambienti distribuiti, garantendo prestazioni elevate anche con l'aumento del carico di lavoro;
- **Ecosistema completo:** *Elastic Stack* offre un ecosistema completo per la raccolta, l'analisi e la visualizzazione dei dati, semplificando la gestione del sistema di monitoraggio;
- **Requisiti aziendali:** la scelta è stata influenzata dalla necessità di integrare il sistema di monitoraggio con l'infrastruttura esistente dell'azienda, che già utilizza componenti dell'*Elastic Stack*;
- **Comunità attiva:** entrambe le tecnologie vantano una comunità di sviluppatori attiva e in crescita, che contribuisce al miglioramento continuo delle piattaforme e fornisce supporto agli utenti.

4.5 Integrazione con l'ambiente esistente

L'integrazione della soluzione di monitoraggio è stata progettata tenendo conto delle specifiche dell'ambiente tecnico aziendale, basato su infrastrutture Linux e containerizzazione tramite Docker.
Tutti i componenti dell'*Elastic Stack* sono stati distribuiti in un ambiente isolato, mantenendo la compatibilità con le versioni approvate dall'azienda.
La comunicazione tra l'applicazione *web PetClinic* e il sistema di osservabilità avviene tramite l'agente *OpenTelemetry Java*, che esporta i dati di telemetria verso l'*Elastic APM Server* gestito da *Fleet*. L'approccio adottato permette un'integrazione trasparente con l'ambiente esistente, garantendo la scalabilità del sistema e la possibilità di estendere la soluzione ad altri servizi o applicazioni monitorate nel futuro.

Capitolo 5

Verifica e validazione

Capitolo 6

Conclusioni

6.1 Consuntivo finale

6.2 Raggiungimento degli obiettivi

6.3 Conoscenze acquisite

6.4 Valutazione personale

Appendice A

Appendice A

Citazione

Autore della citazione

Acronimi e abbreviazioni

AI Artificial Intelligence. 4, 32

APM Application Performance Monitoring. 4, 10, 15, 17, 18, 32

CPU Central Processing Unit. 32

CSV Comma-Separated Values. 10, 19, 32

ESG Environmental, Social and Governance. 2, 32

EUM End User Monitoring. 21, 33

HTTP HyperText Transfer Protocol. 33

MySQL My Structured Query Language. 33

OTLP OpenTelemetry Protocol. 16, 33

PDF Portable Document Format. 10, 13, 19, 33

RUM Real User Monitoring. 33

SDK Software Development Kit. 21, 34

UML Unified Modeling Language. 34

Glossario

AI *Artificial Intelligence* (ing. Intelligenza Artificiale) è un ramo dell'informatica che si occupa della creazione di sistemi in grado di svolgere compiti che normalmente richiederebbero l'intelligenza umana, come il riconoscimento vocale, la visione artificiale, l'apprendimento automatico e la risoluzione di problemi complessi. L'obiettivo principale dell'AI è sviluppare algoritmi e modelli che permettano alle macchine di apprendere dai dati, adattarsi a nuove situazioni e prendere decisioni autonome, migliorando così l'efficienza e l'efficacia in vari settori, tra cui la medicina, la finanza, l'industria e i servizi. 31

APM *Application Performance Monitoring* (ing. Monitoraggio delle Prestazioni delle Applicazioni) è un insieme di pratiche e strumenti utilizzati per monitorare, misurare e gestire le prestazioni e la disponibilità delle applicazioni software. L'obiettivo principale dell'APM è garantire che le applicazioni funzionino in modo ottimale, offrendo un'esperienza utente fluida e senza interruzioni. Ciò include il monitoraggio di vari parametri come tempi di risposta, tassi di errore, utilizzo delle risorse e throughput, nonché l'identificazione e la risoluzione di problemi che potrebbero influire sulle prestazioni dell'applicazione. 20, 22, 31

CPU *Central Processing Unit* (ing. Unità Centrale di Elaborazione), comunemente nota come processore, è il componente principale di un computer responsabile dell'esecuzione delle istruzioni dei programmi e del controllo delle operazioni del sistema. La CPU interpreta e processa i dati, eseguendo operazioni aritmetiche, logiche e di controllo, fungendo da cervello del computer. Le prestazioni della CPU sono influenzate da vari fattori, tra cui la velocità di clock, il numero di core e l'architettura del processore. 18, 20, 31

CSV *Comma-Separated Values* (ing. Valori Separati da Virgola) è un formato di file di testo utilizzato per rappresentare dati tabulari, in cui ogni riga del file corrisponde a un record e i valori all'interno di ogni record sono separati da virgole. Il formato CSV è ampiamente utilizzato per l'importazione e l'esportazione di dati tra diversi programmi, come fogli di calcolo, database e applicazioni di analisi dei dati, grazie alla sua semplicità e compatibilità con molti software. 31

ESG *Environmental, Social, Governance*, (ing. Ambientale, Sociale e di Governance) è un acronimo che indica i criteri utilizzati per valutare la sostenibilità e la responsabilità di un'azienda. L'aspetto ambientale riguarda pratiche come riduzione delle emissioni, uso delle risorse e tutela del clima; quello sociale include rapporti con dipendenti, clienti e comunità, promuovendo inclusione e condizioni di lavoro eque; infine, la governance si riferisce ai meccanismi di gestione, trasparenza, etica e correttezza nei processi decisionali. 31

EUM *End User Monitoring* (ing. Monitoraggio dell'Utente Finale) è una tecnica di monitoraggio delle prestazioni delle applicazioni che si concentra sull'analisi dell'esperienza degli utenti finali mentre interagiscono con un'applicazione o un sistema. L'EUM raccoglie dati in tempo reale sulle prestazioni percepite dagli utenti, come i tempi di risposta, la disponibilità del servizio e gli errori riscontrati, fornendo informazioni preziose per migliorare l'usabilità e l'efficienza dell'applicazione. 31

HTTP *HyperText Transfer Protocol* (ing. Protocollo di Trasferimento Ipertestuale) è un protocollo di comunicazione utilizzato per la trasmissione di dati su Internet, in particolare per il trasferimento di pagine web tra server e client (come i browser web). HTTP definisce le regole e le convenzioni per la richiesta e la risposta di risorse, come documenti HTML, immagini, video e altri contenuti multimediali. Il protocollo è basato su un modello client-server, in cui il client invia una richiesta al server, che elabora la richiesta e restituisce una risposta contenente i dati richiesti. 15–17, 31

MySQL *My Structured Query Language* è un sistema di gestione di database relazionali (RDBMS) open source basato sul linguaggio SQL (Structured Query Language). MySQL è ampiamente utilizzato per la gestione e l'organizzazione di grandi quantità di dati in applicazioni web, software aziendali e sistemi di gestione dei contenuti. Offre funzionalità avanzate come transazioni, replica, partizionamento e supporto per vari motori di archiviazione, rendendolo una scelta popolare per sviluppatori e amministratori di database in tutto il mondo. 17, 31

OTLP *OpenTelemetry Protocol* (ing. Protocollo OpenTelemetry) è un protocollo di comunicazione standardizzato utilizzato per la trasmissione di dati di telemetria, come tracce, metriche e log, tra componenti di sistemi di osservabilità basati su OpenTelemetry. Il protocollo supporta vari formati di trasporto, facilitando l'integrazione con una vasta gamma di tecnologie e infrastrutture. 31

PDF *Portable Document Format* (ing. Formato di Documento Portatile) è un formato di file sviluppato da Adobe Systems per rappresentare documenti in modo indipendente dall'hardware, dal software e dal sistema operativo utilizzati per crearli o visualizzarli. I file PDF possono contenere testo, immagini, grafica vettoriale e persino elementi interattivi come moduli compilabili e collegamenti ipertestuali. Il formato PDF è ampiamente utilizzato per la condivisione di documenti, in quanto preserva il layout e la formattazione originale, garantendo che il documento venga visualizzato correttamente su qualsiasi dispositivo o piattaforma. 31

RUM *Real User Monitoring* (ing. Monitoraggio degli Utenti Reali) è una tecnica di monitoraggio delle prestazioni delle applicazioni web che si concentra sull'analisi del comportamento e dell'esperienza degli utenti reali mentre interagiscono con un sito web o un'applicazione. A differenza delle soluzioni di monitoraggio sintetico, che simulano il traffico utente attraverso script predefiniti, il RUM raccoglie dati direttamente dai browser degli utenti, fornendo informazioni dettagliate sulle prestazioni percepite, i tempi di caricamento delle pagine, gli errori riscontrati e altri aspetti critici dell'esperienza utente. 16, 31

SDK *Software Development Kit* (ing. Kit di Sviluppo Software) è un insieme di strumenti, librerie, documentazione e esempi di codice forniti agli sviluppatori per facilitare la creazione di applicazioni software specifiche per una piattaforma, un framework o un sistema operativo. 31

UML in ingegneria del software *UML, Unified Modeling Language* (ing. linguaggio di modellazione unificato) è un linguaggio di modellazione e specifica basato sul paradigma object-oriented. L'*UML* svolge un'importantissima funzione di "lingua franca" nella comunità della progettazione e programmazione a oggetti. Gran parte della letteratura di settore usa tale linguaggio per descrivere soluzioni analitiche e progettuali in modo sintetico e comprensibile a un vasto pubblico. 31

Bibliografia

Riferimenti bibliografici

James P. Womack, Daniel T. Jones. *Lean Thinking, Second Editon*. Simon & Schuster, Inc., 2010.

Siti web consultati

Manifesto Agile. URL: http://agilemanifesto.org/iso/it/.