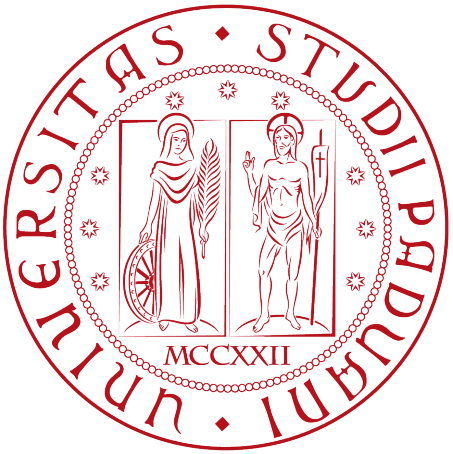
Università degli Studi di Padova

Dipartimento di Matematica

Corso di Laurea in Informatica



Containerizzazione di Malware Dashboard

Relazione finale di stage

Relatore
Prof. Tullio Vardanega

Laureando
Andrei Petrov

Anno Accademico 2016-2017

Sommario

//TODO

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

L'Azienda

1.1 IKS

IKS *(Information Knowledge. Supply)* è un'azienda padovana fondata, dall'attuale Amministratore Delegato Paolo Pittarello, nel 1999.

Nell'insieme, IKS unisce figure di alto profilo con lo scopo di proporre soluzioni innovative alle richieste di mercato dell'Information and Communication Technology (ICT) sia italiano che estero. Le soluzioni offerte interessano in particolare gli ambiti della sicurezza, dell'infrastruttura e della governance Information Technology (IT).

L'azienda è in continua ricerca tecnologica. Investendo sulla formazione del proprio personale, IKS si impegna di portare solo valore aggiunto al business dei propri clienti. Inoltre, l'Azienda crede fortemente nell'innovazione come strumento verso un ambiente digitale comune, Agile (metodologia) e totalmente disponibile.

Il quartier generale aziendale è a Padova. Inoltre, IKS possiede uffici anche nelle seguenti città: Roma, Milano e Trento.

A partire dallo scorso anno (2016), IKS SRL, Kirey SRL, Insirio SPA e System Evolution SRL hanno fondato il Gruppo Kirey. L'obiettivo comune delle quattro aziende è l'unione delle competenze complementari e garantire un portfolio completo di soluzioni ai clienti attuali e futuri.

La creazione del Gruppo Kirey è stata guidata dalla Synergo SGR., società di *private equity*. Il presidente del nuovo Gruppo commerciale è Vittorio Lusvarghi.

A seguito della creazione del Gruppo, IKS SRL e le restanti tre realtà aziendali hanno conservato la propria struttura di governance e management, con il fine di garantire la propria continuità gestionale.

1.2 Profilo dell'azienda

1.2.1 Servizi e prodotti offerti

Nel corso degli anni, IKS si è fatta notare per gli enormi contributi innovativi nell'ambito della sicurezza informatica. Tuttavia, essa non è limitata a questo ambito. Infatti, gli altri ambiti di applicazione sono: infrastruttura e governance IT.

Di seguito vengono presentati i servizi offerti da IKS per ciascun ambito di applicazione:

* **IT Security**

  - **Risk analysis e vulnerability assessment**
    È importante garantire la sicurezza dell'infrastruttura informatica nel suo complesso. A questo scopo, IKS offre un servizio orientato alla ricerca di eventuali vulnerabilità e analisi dei rischi a esse collegate;

Figura 1.1: Visione a processo della gestione del rischio.Immagine tratta da: http://bit.ly/2rh3V0A.

  - **Audit management**
    Le aziende di continuo sono sottoposte a controlli di vario genere; il loro scopo è l'accertamento della regolarità delle aziende con: certificazioni, normative, bilanci ed ecc. IKS offre un servizio di supporto per le aziende con il fine di agevolare le attività di *auditing* ed eventualmente per migliorare i loro processi interni;

Figura 1.2: Flusso di lavoro durante lo svolgimento di un *audit*. Immagine tratta da: http://bit.ly/2rdFhfv.

  - **Difesa perimetrale**
    Sempre in ambito della sicurezza è importante prendere le giuste misure per garantire a priori uno specifico livello di sicurezza e limitare a zero le intrusioni dall'esterno di un'infrastruttura IT aziendale. A questo scopo, IKS offre un'insieme di soluzioni orientare al monitoraggio degli accessi a sistemi aziendali, dei permessi sulle operazioni che un utente può effettuare, e molto altro;

* **IT Infrastructure**

Figura 1.3: Visione grafica del concetto di difesa perimetrale. Immagine tratta da: http://bit.ly/2s834O2.

– **Business continuity**
In ambito bancario, le infrastrutture informatiche sono molto complesse. La manutenzione delle infrastrutture informatiche non è semplice. La sfida più difficile è garantire che questi sistemi siano operativi al 100%. Una simile percentuale nella pratica è impossibile. IKS con il proprio gruppo di esperti sono alla continua ricerca di soluzioni per incrementare la percentuale di continuità operativa di questi sistemi. Infatti, le soluzioni offerte dall'azienda sono orientate nel concreto all'infrastruttura del cliente richiedente supporto;

Figura 1.4: Visione del ciclo di vita del processo di *business continuity*. Immagine tratta da: http://bit.ly/2qvCmgP.

– **Virtualization technology**
Ogni prodotto software di business per portare valore aggiunto deve essere eseguito. Eseguire un prodotto software per server fisico richiede la disponibilità di un cospicuo numero di server. A questo scopo la tecnologia di virtualizzazione permette la creazione di server virtuali che eseguono programmi e questi vengono eseguiti da server fisici. I benefici di una simile infrastruttura è l'ottimizzazione delle risorse di calcolo, agilità di gestione e sicurezza. Alcune delle soluzioni di virtualizzazione offerte da IKS sono: VMWare, RHEV ed ecc. Un'evoluzione della tecnologia di virtualizzazione è

il *Cloud*. In questo ambito, IKS propone soluzioni di migrazione e supporto verso il Cloud dell'infrastruttura IT classica di un'azienda;

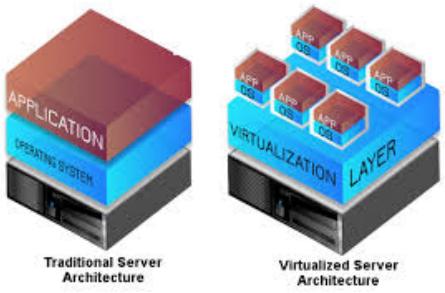


Figura 1.5: Vista a confronto: ambiente server bare metal e virtualizzato. Immagine tratta da: http://bit.ly/2qvtLLk.

∗ **IT Governance**

– **Service management**
Un servizio informatico di business, a causa della sua criticità, richiede costante attenzione. Il monitoraggio del servizio informatico presenta la necessità di enormi investimenti economici. IKS offre piani di gestione per soddisfare anche i più esigenti clienti;

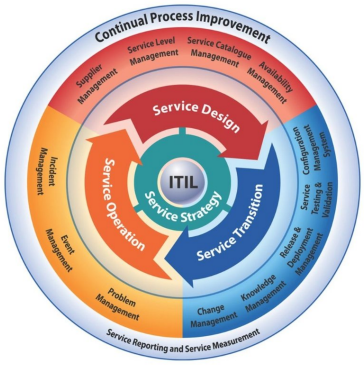


Figura 1.6: Visione della gestione di servizio in prospettiva del Framework ITIL. Immagine tratta da: http://bit.ly/2qvNryk.

– **Application and performance monitoring**
Ogni prodotto software ha il proprio specifico ciclo di vita. Concluso il ciclo di sviluppo, il prodotto è rilasciato in produzione. La seconda parte del ciclo di vita di un prodotto software è la manutenzione. Il monitoraggio di un applicativo è importante per avere una costante visione dello stato del prodotto e prevenire eventuali esigenze di manutenzione generica oppure di basso profilo a livello di codice sorgente. In questo dominio, grazie a partnership strategiche, IKS offre soluzioni mirate a garantire la miglior possibile esperienza di monitoraggio applicativo;

  - **System and networking management**
    Gestire sistemi e reti informatiche è un compito complesso. L'utilizzo di strumenti adeguati permette di semplificare il lavoro e garantisce un stato consistente del sistema nel tempo. Le soluzioni che IKS offre sono orientate alla flessibilità e facilità d'uso dei prodotti offerti in questo contesto;

* **Innovation & Project**

  - **Architetture applicative distribuite**
    I sistemi informatici diventano sempre più di natura distribuita. IKS offre in questo ambito soluzioni architetturali orientate a microservizi, utilizzando le ultime tecnologie orientate alla containerizzazione e orchestrazione di container;
  - **Sviluppo di applicazioni cloud native**
    È comune sentire parlare di cloud. Le classiche architetture applicative non riescono a beneficiare della flessibilità del cloud, perché in organizzazione e struttura non sono scalabili e sono difficilmente modularizzabili. Applicazioni che vengono gestite nel complesso come un'unica unità prendono il nome di monolite. La diretta conseguenza di una simile organizzazione è il carattere statico e poco flessibile dell'applicazione. Paradigmi nuovi, per la messa in esercizio di applicazioni, mancano l'intigrazione con architetture software tradizionali. Per questo motivO, le applicazioni devono essere sviluppate fin dal principio con un'architettura orientata al Cloud. Una buona guida, di sviluppo di applicazioni orientate al Cloud, è la segeunte: *Twelve Factor-Factor App.*, Heroku, servizio PaaS per applicazioni cloud native, promuove continuamente l'importanza dei 12 principi alla base della filosofia *cloud native*. In questa direzione IKS propone servizi di sviluppo di applicazioni di business orientate all'affidabilità, resilienza, scalabilità orizzontale ed ecc.

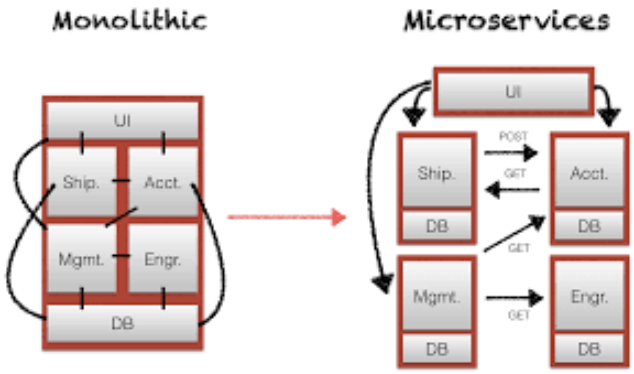


Figura 1.7: Visione architetturale a monolite e microservizi a confronto. Immagine tratta da: http://bit.ly/2rh1niY.

La clientela tipica di IKS sono le aziende operanti nei seguenti ambiti: pubblica amministrazione, bancario, assicurativo e servizi.

Una lista dettagliata delle referenze può essere consultata sul sito di IKS (https://www.iks.it/referenze.html).

1.2.2 Struttura organizzativa

Ad oggi, IKS conta più di 100 dipendenti. La sua organizzazione interna è riassunta nel diagramma in **Figura 1.8**.

In seguito, descrivo le unità operative che costituiscono il nucleo decisionale dell'azienda. Queste unità sono:

* **Direzione**
  Definisce gli orientamenti e le politiche aziendali, gli obiettivi per la qualità, riesamina periodicamente il sistema di qualità e gestisce il piano di formazione dei dipendenti in funzione alle esigenze e motivazioni personali;

* **Direzione Commerciale**
  Definisce le politiche commerciali, gli obiettivi e le risorse necessarie. Promuove i servizi e prodotti dell'azienda. Gestisce i clienti, i fornitori e le offerte contrattuali;

* **Direzione tecnica o Operation**
  Supporta la Direzione Commerciale nella valutazione commerciale di prodotti e/o offerte dal punti di vista tecnico. Gestisce a livello tecnico i progetti e servizi. Pianifica le risorse necessarie per i prodotti/servizi. Verifica lo stato del prodotto/servizio offerto;

* **Amministrazione & Finanza**
  Gestisce la documentazione di progetto, su coordinamento della direzione commerciale e tecnica. Gestisce l'archiviazione della documentazione;

* **Acquisti**
  Su coordinamento della Direzione, gestisce i fornitori di prodotti e servizi. Gestisce il processo di acquisizione di nuovi prodotti o servizi. Il processo di acquisizione è guidato dalle necessità interne aziendali oppure da quelle dei clienti;

* **Assicurazione Qualità**
  È a stretto contatto solo con la Direzione. Gestisce il piano di qualità, coordina le attività di ispezione, misura e stima il livello della qualità aziendale;

* **Business Unit (BU)**
  Gestisce i progetti o servizi concordati con il Cliente. Rendiconta direttamente alla Direzione Tecnica e gestisce l'emissione delle fatture verso il Cliente.

1.2.3 Processi aziendali

IKS, a partire dal 2003, è certificata UNI EN ISO 9001. Questo certifica che l'azienda cura molto la qualità del proprio lavoro. Infatti, il miglioramento continuo permette all'azienda di rimanere competitiva e consolidare la propria posizione di leader sul mercato del ICT italiano. Riporto, di seguito, alcuni obiettivi di qualità dell'azienda:

* Mantenere e aumentare il livello di soddisfazione del Cliente;

* Operare in modo efficiente ed efficace per soddisfare i requisiti contrattuali, norme e regolamenti;

* Monitorare i propri processi per: garantire azioni correttive tempestivamente e permettere un comportamento pro attivo, anticipare i bisogni e predire le risorse aziendali necessarie prima dell'effettivo bisogno;

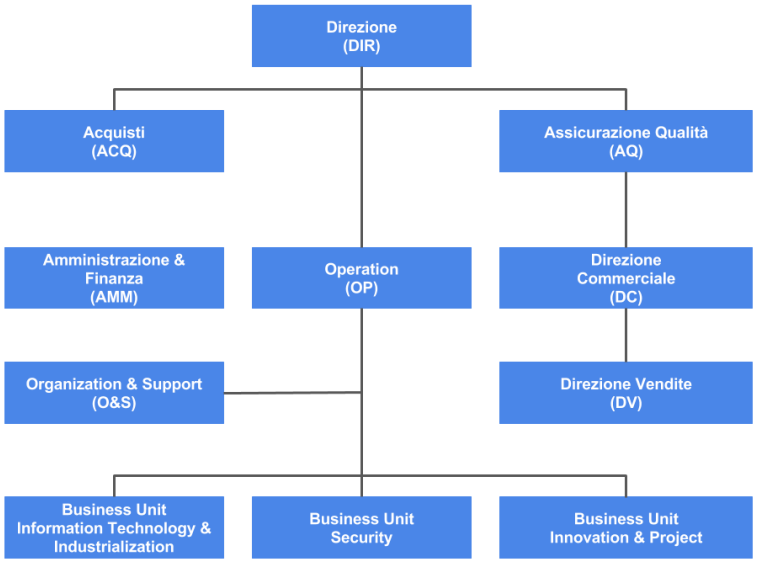


Figura 1.8: Organigramma aziendale

* Assicurare una adeguata formazione al Personale.

Il Cliente copre un ruolo importante nella quotidinità di IKS. Infatti, l'azienda cerca di coinvolgere i propri clienti il più possibile., questo è necessario per comprendere meglio i bisogni attuali del cliente e cogliere esigenze future. In azienda, il passo successivo alla formalizzazione del bisogno del Cliente segue un'attività di analisi dei requisiti. L'obiettivo dell'attività è la dettagliata comprensione del contesto applicativo, quali sono le parti interagenti e quali possono essere i rischi, durante l'attività di progetto, per implementare i bisogni del Cliente.

A progetto concluso, il Cliente valuta criticamente la soluzione presentata. La valutazione, eventualmente, coinvolge un reclamo. Questo è rivolto alla Direzione dell'Azienda.

IKS organizza il proprio lavoro per processi: primari, direttivi e di supporto. Ciascuna categoria di processo definisce delle responsabilità e compiti. Per esempio, i processi organizzativi interessano le attività per: definire la politica e strategia aziendale, pianificare e allocare le risorse, riesaminare la gestione del sistema di qualità. Invece, i processi primari ricoprono attività che garantiscono un diretto ricavo economico per l'azienda e danno un valore aggiunto al prodotto o servizio fornito. Esempi di attività in questa categoria sono: proporre offerte commerciali ai clienti, progettare e sviluppare prodotti software, erogare servizi IT. L'ultima categoria di processi sono i processi di supporto. Le consone attività giornaliere riguardano: gestire le risorse umane, l'infrastruttura e gli ambienti di lavoro., monitorare e analizzare la qualità aziendale.

In **Figura 1.9** segue una presentazione dello schema organizzativo utilizzato dall'Azienda, durante il ciclo di vita di un progetto.

Periodicamente, il responsabile della qualità, incaricato della Direzione, attua attività d'ispezione. L'obiettivo dell'attività è il controllo del livello di qualità fornita

dai dipendenti aziendali. A posteriori, segue un'attività di analisi e misura dei livelli di qualità organizzati per BU, servizio e prodotto offerto da IKS.

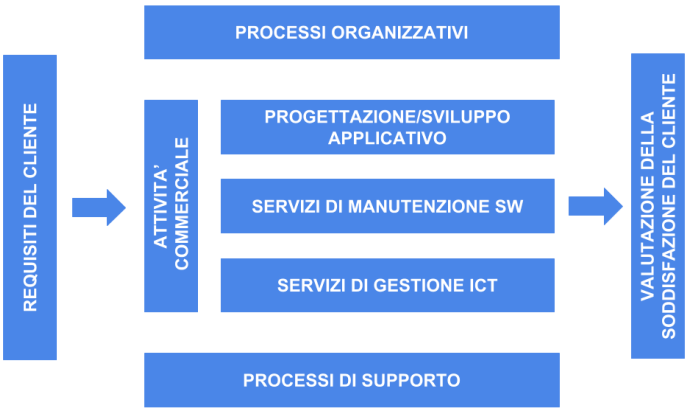


Figura 1.9: Rappresentazione grafica del coinvolgimento del Cliente e i corrispettivi livelli degli interventi del gruppo commerciale, tecnico, direzionale e di supporto nella gestione di un'offerta di progetto.

1.3 Rapporto con l'innovazione

L'innovazione è il processo di gestione dell'intero ciclo di vita di un'idea. L'obiettivo è: portare un miglioramento di processo aziendale, di prodotto e/o di servizio. Le conseguenze dirette del miglioramento sono: valore aggiunto, per l'azienda, in termini di rientro economico e soddisfare un bisogno, per il Cliente, in modo efficace ed efficiente.

L'approccio innovativo induce l'utilizzo dell'informazione, della creatività e dello spirito d'iniziativa per raccogliere maggior valore aggiunto dalle risorse a disposizione. L'azienda utilizza l'innovazione per soddisfare in modo pro attivo le richieste del Cliente. Questo principio è pienamente il linea con la strategia di qualità aziendale: *client first*.

La modalità di innovazione di IKS è un approccio incrementale. Inizialmente l'azienda cerca di soddisfare i bisogni principali e raggiungere il prima possibile gli obiettivi minimi pre-fissati. In seguito, l'azienda migliora la propria offerta mediante incrementi continuativi di dettaglio.

Per supportare l'innovazione, IKS ha creato una cultura aziendale che permette ai propri dipendenti di scambiarsi idee, sperimentare, imparare in gruppo e mettere in atto la propria creatività. Non manca la comunicazione con i propri responsabili. Questi sono i primi a motivare di continuo le risorse umane a loro disposizione. Il dialogo dipendente-responsabile non è verticale. La cultura aziendale in questa direzione è molto drastica: favorire uno scambio di idee in modo che esso sia equo, semplice e non orientato alle gerarchie aziendali.

In questo contesto, per l'intera durata del mio periodo di stage e dopo un primo momento di ambientamento, io ho beneficiato molto del clima aziendale. Infatti, non è mancato il libero confronto con il tutor aziendale, il quale ha mostrato disponibilità e

apertura al mio spirito d'iniziativa. Sempre mio tutor aziendale ha supportato me in ogni scelta decisionale che io abbia motivato e ritenuto significativa per il beneficio del mio progetto.

Le idee sono una parte del processo di gestione dell'innovazione: la realtà è molto più complessa. IKS non possiede un effettivo processo di gestione a livello aziendale. Questo viene gestito da un gruppo di persone con competenze trasversali e a livelli organizzativi differenti.

Figura 1.10: Il legame attivo tra innovazione e il processo del cambiamento e gestione della conoscenza. Immagine tratta da: http://bit.ly/2qty3XD.

Capitolo 2

L'azienda e gli stage

L'azienda investe costantemente sui propri dipendenti per soddisfare i bisogni tecnologici del mercato con le giuste competenze. I dipendenti dell'azienda dedicano parte della propria giornata lavorativa in approfondimenti personali e attività di formazione.

A supporto dell'attività di laboratorio, il team IT interno ha installato un ambiente virtuale. Disporre in azienda di un simile *environment* permette ai dipendenti di sperimentare con: tecnologie, integrazione di sistemi ed ecc. Inoltre, l'ambiente virtuale offre un'infrastruttura dedicata per i progetti di stage. I temi comuni di sperimentazione sono: il *Cloud*, *Machine Learning* e *Analytics* ed ecc.

Di recente, l'Azienda ha concluso una partnership strategica con AWS (*Amazon Web Services*). Quest'accordo di collaborazione, permetterà a IKS di migrare verso il Cloud la propria infrastruttura informatica e beneficiare delle peculiarità del Cloud, per esempio: elasticità, alta affidabilità, flessibilità di gestione, semplice accesso remoto ed ecc.

Ogni anno, l'azienda partecipa a StageIT: un evento completamente dedicato agli studenti universitari dei Corsi di Laurea in Scienze ed Ingegneria Informatica. Infatti, StageIT permette allo studente di mettersi in contatto diretto con le aziende e queste ultime di promuovere i propri progetti di stage. L'evento prevede, inoltre, un concorso per premiare il miglior progetto di stage, svolto nell'edizione dell'anno precedente. Il vincitore del concorso, scelto dagli studenti, ottiene come premio un buono d'acquisto del valore di 500 Euro.

2.1 Il valore aggiunto di uno stagista

IKS è un partecipante attivo a StageIT; annualmente l'azienda propone fino a 6 progetti di stage. Gli argomenti degli stage non sono verticali su un'unica tematica, ma coinvolgono temi come:

* Sviluppo di applicazioni basate su web, Cloud, mobile o migrazione su Cloud/mobile di applicazioni tradizionali;
* Progettazione di ambienti, metodologie e strumenti di sviluppo software.

Lo stagista è una risorsa importante per IKS; l'azienda vede lo stagista come portatore di idee nuove e contribuitore nel consolidare il valore aggiunto aziendale. In

principio, l'azienda impiega lo stagista su progetti di sperimentazione. Questi ultimi hanno come obiettivo l'analisi di fattibilità e lo studio dell'integrazione delle soluzioni nell'offerta commerciale dell'azienda.

Per l'intera durata dello stage, lo stagista opera in un ambiente vero simile alla realtà aziendale. Il tutor esterno, oltre a guidare nel lavoro lo stagista in relazione a un PdL (Piano di Lavoro), osserva le attività dello stagista. Le osservazioni contribuiscono alla valutazione finale dell'attività di stage del tirocinante.

L'Azienda, con il contributo degli stagisti, allinea se stessa con i temi di ricerca universitari e con le tendenze tecnologiche del momento sul mercato internazionale.

2.2 Alcuni temi di stage

2.2.1 AIOps e Machine Learning

Il progetto di stage tratta l'integrazione del *Machine Learning* con strumenti di Application Performance Monitoring. L'obiettivo dello stage è sperimentare questo nuovo approccio di monitoraggio delle applicazioni, integrando diverse soluzioni in questo ambito e fornire uno studio del prodotto finale. Una conseguenza importante di questo progetto è lo sviluppo di un pensiero critico per affrontare le più difficili sfide del monitoraggio di applicazioni e infrastrutture.

Il presente progetto si colloca nell'ambito del *application and performance monitoring* che è un servizio offerto dall'azienda al supporto della governance IT.

2.2.2 DevOps Automazione

L'automazione è fondamento di ogni realtà aziendale contemporanea. Infatti, il numero di macchine da gestire spesso non è piccolo. Per semplificare i compiti di gestione si devono utilizzare strumenti di configurazione e automazione. Queste tecnologie permettono di automatizzare tutte le operazioni manuali che un sistemista spesso compie durante le attività di manutenzione giornaliere. L'obiettivo di questo progetto è l'integrazione di alcuni strumenti che semplificano il Patching dei server e sperimentare con nuove tecnologie del settore. Il presente progetto si colloca nell'ambito del *system management*.

2.2.3 Sviluppo moduli evolutivi in ambito antifrode

IKS ha grande esperienza in ambito della sicurezza informatica bancaria. Uno dei prodotti risultati di questa esperienza è SMASH. L'obiettivo dello stage è estendere il prodotto con qualche funzionalità di monitoraggio di azioni sospette. Oltre allo sviluppo di moduli evolutivi lo stagista ha la possibilità di apprendere delle competenze forti nell'ambito della sicurezza informatica. La presente proposta di stage è un progetto inter business unit dell'azienda. Esso si colloca nell'ambito dello sviluppo di prodotti software e della sicurezza informatica nel settore bancario.

2.3 Il progetto proposto

2.3.1 Motivazioni

E' sempre più comune, nelle realtà aziendali, l'approccio agile. Quest'approccio promuove la comunicazione e concentra l'attenzione di tutti i stakeholder sul valore

finale di prodotto e/o strategia da raggiungere.

Se gli sviluppatori hanno come obiettivo primario lo sviluppo di un prodotto software allora i professionisti dell'IT hanno come priorità la garanzia di servizio e manutenzione periodica del prodotto software realizzato.

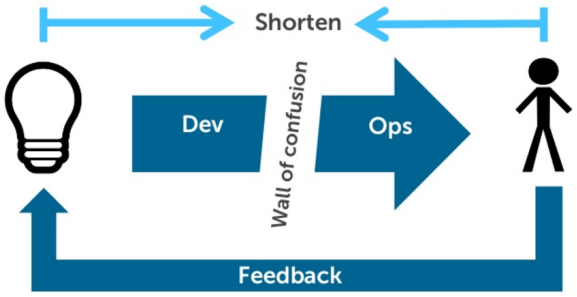


Figura 2.1: Sia gli sviluppatori che i professionisti IT sono portatori di valore: un feedback che coinvolge ambe le parti è essenziale. Immagine tratta da: http://bit.ly/2rM9lBQ.

Tra i due gruppi esiste un muro di incomprensione. La mancanza di comunicazione ed interazione rafforza questo fenomeno. Le problematiche, che avvengono dopo il rilascio del prodotto software, sono responsabilità dei professionisti IT. Di conseguenza, le loro attività sono orientate alla risoluzione dei problemi. Una simile organizzazione della distribuzione delle responsabilità è normale in contesti di realtà aziendali con rilasci sporadici.

Un'azienda informatica, che deve affrontare un numero elevato di rilasci giornalieri, necessita di un approccio diverso. A questo scopo il movimento culturale, chiamato DevOps, è orientato all'unione degli sviluppatori e sistemisti. L'unione promuove un cambio di mentalità, creazione di nuove competenze e sviluppo di nuovi strumenti che diminuiscano la distanza tra le due realtà.

Il DevOps ha conseguenze più profonde del semplice cambio culturale. Esso introduce un cambiamento interno orientato alla modifica del modello di qualità. I benefici di questo cambiamento sono i seguenti: maggiore innovazione, qualità di prodotto, processo e agilità nel cogliere i bisogni di mercato del momento.

Una tipica rappresentazione del ciclo di vita DevOps segue in figura.

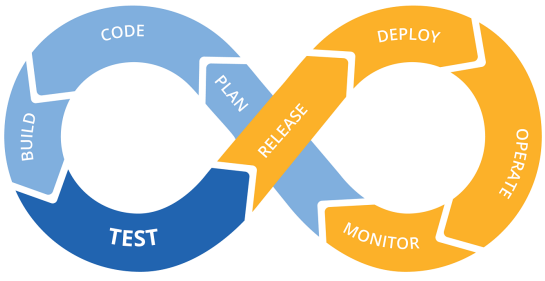


Figura 2.2: Il DevOps abilita l'automazione del processo di rilascio del software e i cambi dell'infrastruttura IT. Immagine tratta da: http://bit.ly/2rsw9nm.

L'abilità di cambiare in modo agile è un grande beneficio per le aziende: il modo di lavorare diventa più flessibile e i dipendenti mostrano un maggior coinvolgimento nella quotidianità aziendale.

Lato infrastrutturale, invece, la gestione diventa: disciplinata, sistematica e standardizzata. Con un approccio standardizzato e ben strumentalizzato è sempre più difficile individuare server nomadi. Può succedere che, in caso di operazioni sofisticate e mirate alla manutenzione, un server scompaia dall'orizzonte di visibilità. In assenza di opportuni strumenti è difficile individuare l'accaduto.

Sebbene il DevOps possieda uno scopo più ampio, il *continuous delivery* è un approccio che promuove l'automazione di tutti i processi coinvolti durante il rilascio di un prodotto software. Questo permette di abbreviare i tempi, aumentare il numero dei rilasci e migliorare la gestione del cambiamento.

Essere veloci nel *delivery* di un prodotto software non è sufficiente. E' importante prevedere una *pipeline* di *deployment* che coinvolga ogni *stage* del ciclo di *operation*. Automatizzare il *deployment* implica minor intervento manuale, minor numero di errori e conseguentemente maggiore formalità nelle attività complessive coinvolte.

Un *application container* permette di confezionare le applicazioni e dipendenze esterne in unità singole. Implementare il *packaging* in questo modo le applicazioni facilita lo scambio di artefatti tra tutti i gruppi coinvolti nel ciclo di vita del prodotto software. Di conseguenza; sia il professionista IT che lo sviluppatore instaurano un protocollo di comunicazione e scambio di esperienze basato su qualcosa di concreto, stabile e funzionante. Il classico detto "funziona sul mio computer" perde di significato.

Al momento la comunità open source offre molte tecnologie di containerizzazione. Quella più stabile e famosa nel comunità Linux è LXC (*Linux Kernel Container*); questa tecnologia aggiunge il supporto a livello del kernel dei container applicativi per i sistemi operativi Linux. Docker, invece, è un'altra soluzione di containerizzazione. Le prime versioni di Docker interagivano con LXC; le versioni recenti di Docker hanno rimosso la dipendenza software verso LXC e ha implementato una soluzione di containerizzazione proprietaria.

La containerizzazione è una tecnologia molto attraente. Visti i vantaggi tecnici, la containerizzazione porta a un livello superiore il modello concettuale rappresentante un'applicazione. L'architettura dei prodotti software, dal punto di vista dei container, risulta essere più componibile. In un ambiente dinamico, caratterizzato dall'automazione, verifiche e *deployment* automatici, le classiche architetture software non sono pensate per beneficiare di questa flessibilità. Lo stesso non vale per i microservizi.

I microservizi rappresentano uno stile architetturale in sintonia con la filosofia Unix: ogni microservizio implementa una sola funzionalità - basso accoppiamento.

La figura presenta graficamente diversi microservizi. Ciascuno dei quali ha una responsabilità ben definita. Per garantire un basso accoppiamento tra i microservizi, il sistema deve utilizzare un servizio infrastrutturale e di supporto, chiamato *Service Discovery*, utilizzato come un DNS (*Domain Name System*). In questo modo, i microservizi continuano ad essere autonomi, mentre la gestione della comunicazione inter microservizio è separata dall'evoluzione dei singoli microservizi. In aggiunta, i microservizi beneficiano della *space transparency*. Se un microservizio X, in esecuzione su una macchina A, migra per eseguire su una macchina B, allora un microservizio Y, che vuole comunicare con X, deve contattare il *Service Discovery* per ottenere l'indirizzo di X. L'effetto ottenibile è un alto tasso di mobilità dei servizi.

E' usuale incapsulare un microservizio in una capsula - il container software. Per la proprietà di isolamento: nello stesso ambiente possono coesistere due o più copie dello stesso microservizio, riducendo quasi a zero l'interferenza di un microservizio

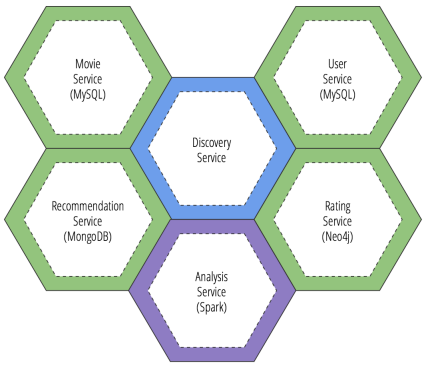


Figura 2.3: Esempio di un sistema a microservizi. Immagine tratta da: http://bit.ly/2qNXKxj.

su un'altro. Con la scalabilità orizzontale i microservizi beneficiano dell'incremento in robustezza e alta affidabilità. Per implementare il *routing* delle richieste verso i microservizi, i professionisti IT configurano un microservizio di *load balancing* a livello applicativo del modello OSI (*Open System Interconnection*).

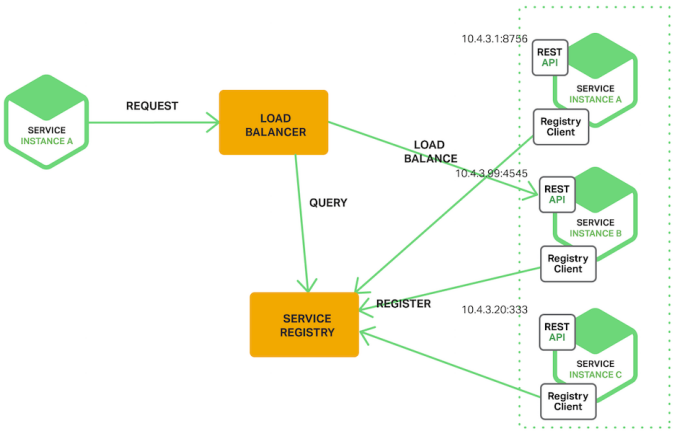


Figura 2.4: I microservizi permettono di scalare orizzontalmente per reggere ai più esigenti carichi di lavoro. Immagine tratta da: http://bit.ly/2qI50LR.

I microservizi semplificano l'analisi, progettazione e implementazione dell'applicazione complessiva, scomponendo il prodotto complessivo in sotto applicazioni indipendenti; inoltre, i microservizi complicano l'applicazione a causa di: un elevato numero di componenti da gestire, difficoltà di versionare i microservizi in modo consistente e indipendente, politiche complesse di monitoraggio ed ecc. Per la tracciabilità delle richieste inter servizio è usuale utilizzare strumenti specializzati nel monitoraggio. Esempi di applicazioni, a questo scopo, sono: Appdynamics, Dynatrace, Instana ed ecc.

I container e microservizi aprono nuove sfide sia per gli sviluppatori che per i

sistemisti. E queste sfide caratterizzano il contratto di collaborazione tra i due gruppi.

2.3.2 Obiettivi aziendali

Un team interno di IKS ha sviluppato una soluzione di Executive e Malware Dashboard basata sullo stack applicativo: Elasticsearch, Logstash e Kibana.

La mia attività di stage ha avuto la containerizzazione della soluzione precedentemente implementata come obiettivo principale.

Di seguito presento gli obiettivi principali della mia attività di stage:

* Containerizzare le componenti applicative, costituenti la soluzione implementata dal team interno di IKS;
* Garantire la non regressione al livello funzionale della soluzione;
* Predisporre un ambiente containerizzato con e senza orchestratore di container.

In conclusione, la containerizzazione garantisce che la soluzione di executive e malware dashboard sia portabile su ambienti diversi, dal portatile del sviluppatore al Cloud.

2.3.3 Obiettivi personali

Come attività preliminare alla ricerca di un progetto di stage per la Laurea ho attuato uno studio individuale di mercato. Lo scopo era capire: tendenze tecnologiche, architetturali e metodologiche. Se da un lato le mie ricerche hanno cercato di cogliere le novità del momento, dall'altro a livello personale queste erano mirate alla ricerca di un contesto in cui potermi applicare e maturare.

Con il presente progetto gli obiettivi personali erano:

* Apprendere conoscenze e competenze in ambito di:
  - Virtualizzazione basata sulla tecnologia a container;
  - Sistemi distribuiti;
  - Amministrazione di sistema Linux;
* Acquisire esperienza pratica nella gestione delle reti di calcolatori in ambito dei sistemi, nello specifico le reti definite in modo programmatico per le tecnologie orientate alla containerizzazione;
* Acquisire esperienza nell'analisi, progettazione e implementazione di sistemi orientati ai microservizi;
* Famigliarizzare con la piattaforma Kubernetes e i principi del Cloud.

2.4 Piano di lavoro

L'azienda ha pianificato un piano di lavoro (PdL) per un totale di 300 ore complessive. Ho consegnato il documento del PdL all'Ufficio degli Stage presso l'Ateneo dell'Università di Padova; una seconda copia del PdL ho consegnato al tutor interno; l'ultima copia, invece, controfirmata dall'ufficio stage dell'Università ho consegnato all'azienda.

Descrivo brevemente di seguito il contenuto del PdL inerente all'insieme delle attività svolte:

* Fase 1 - Formazione (56 ore)
  - Docker: la tecnologia per la containerizzazione;
  - Kubernetes: la tecnologia per l'orchestrazione;
  - Elasticsearch, Logstash e Kibana (ELK): lo stack applicativo;
  - Verifiche delle competenze acquisite;
* Fase 2 - Analisi e progettazione (56 ore)
  - Analisi delle funzionalità della soluzione non containerizzata di dashboard;
  - Analisi delle modalità di containerizzazione delle componenti;
  - Analisi delle modalità di *deployment*;
  - Progettazione delle modalità di verifica della non regressione;
  - Progettazione architetturale della soluzione;
  - Progettazione della modalità di *deployment*;
  - Documentazione;
* Fase 3 - Implementazione (188 ore)
  - Installazione e configurazione dell'orchestratore;
  - Implementazione della soluzione in un contesto con e senza orchestratore;
  - Verifica di non regressione;
  - Documentazione.

2.5 Vincoli

2.5.1 Vincoli temporali

Lo stage ha avuto una durata di 8 settimane, per un complessivo di 310 ore di lavoro. Ho lavorato a tempo pieno con il seguente orario: 9.00-18.00. Con la pausa pranzo di 1 ora dalle 12.30 alle 13.30. Come pianificato nel PdL, ho seguito le attività in ordine seguendo le fasi della pianificazione. Qualche volta ho alterato l'ordine delle attività per adattare le esigenze e ridurre gli effetti del cambio di contesto. Infatti, ogni fase ha coinvolto attività mirate al raggiungimento di specifici obiettivi. Per maggior dettaglio sul contenuto del PdL riferire la sezione Piano di Lavoro.

2.5.2 Vincoli tecnologici

Fin dal primo giorno di lavoro l'azienda mi ha fornito un portatile dedicato per l'itero periodo di stage. Inoltre, mi è stato vietato di collegare alla rete aziendale qualsiasi dispositivo personale. Inoltre, il portatile di lavoro non poteva essere portato a casa. Per comunicare internamente sono stati utilizzati strumenti di messaggistica istantanea, come Skype, per comunicazioni informali e la posta elettronica.

Oltre a questo vincolo, a livello tecnologico sono state fissate le seguenti tecnologie:

* CentOS7: il sistema operativo installato sulle macchine di laboratorio. CentOS7 è la versione open source di RHEL7 (Red Hat Enterprise Linux versione 7);

* Docker: lo strumento che permette in modo estremamente facile la creazione, il *deployment* e l'esecuzione di applicazioni utilizzando la tecnologia a container. In questo modo l'utente focalizza l'attenzione su questioni diverse dall'installazione e configurazione dell'applicazione. L'architettura di Docker segue in figura.

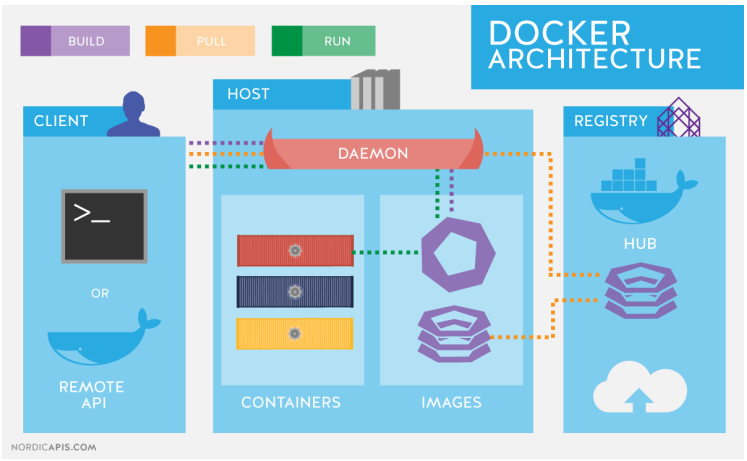


Figura 2.5: Le parti costituenti la piattaforma Docker sono: il demone, il client e il registry Docker. Immagine tratta da: http://bit.ly/2rmkt7g.

Le componenti architetturali costituenti la piattaforma Docker sono il: demone, client e registry. L'architettura di alto livello è un architettura client/server. Il server di Docker è il demone che ha la responsabilità di gestione dei container sulla macchina locale. Mentre il client si interfaccia tramite un'interfaccia REST al demone e permette di interagire in modo agile con i container, creare reti virtuali, gestire i dati che devono essere condivisi tra i container e il file system locale della macchina. Infine, il registry di Docker è una repository che può essere pubblica o privata e ha la responsabilità di abilitare la condivisione di immagini utili alla creazione dei container. Come modello mentale, in relazione con il paradigma ad oggetti, è possibile paragonare le immagini Docker a classi che devono essere istanziate per la creazione di oggetti, ovvero container.

Per favorire il libero scambio di immagini Docker, l'azienda Docker Inc ha messo a disposizione degli utenti un hub: spazio web per la condivisione delle immagini Docker. Nella modalità privata di una repository, Docker Inc rende disponibile il servizio di *scanning* delle vulnerabilità delle immagini.

Quando l'utente esegue il comando run tramite la CLI di Docker, il demone controlla che l'immagine da istanziare, per la creazione del container, sia presente sul *file system* locale. In caso affermativo, il demone Docker crea e mette in esecuzione il container, altrimenti esso scarica prima l'immagine dal registry e al termine, di questa attività, istanzia il container;

* Kubernetes: è un sistema open source per automatizzare il *deployment*, la scalabilità e gestione di applicazioni containerizzate. La tecnologia è il risultato

di 15 anni di esperienza in Google con la tecnologia a container. Kubernetes garantisce la portabilità delle applicazioni e l'indipendenza dall'ambiente fisico di esecuzione.

Kubernetes è un sistema distribuito per l'orchestrazione di container applicativi. Il modello architetturale dell'orchestratore è master/slave.

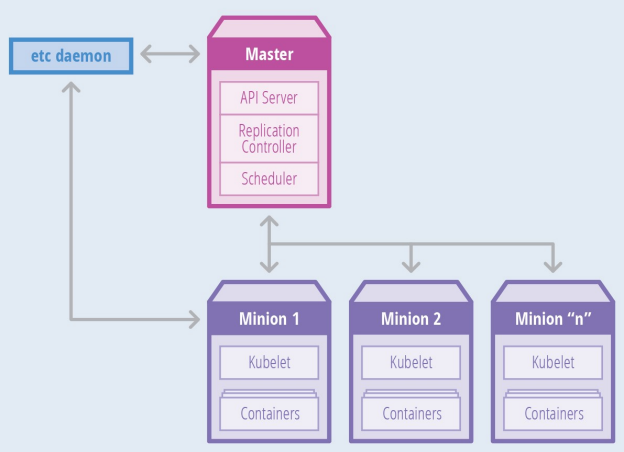


Figura 2.6: Le componenti architetturali di Kubernetes sono: API server, Scheduler, Replication Contoller, Kubelet, Kube proxy, Database Etcd. Immagine tratta da: http://bit.ly/2s4eKUR.

Il master è un pannello di controllo del sistema K8s (Kubernetes). Lui gestisce il *workload* dei container. Inoltre, esegue le componenti critiche del sistema: il database chiave valore ad alta consistenza **etcd**, il gestore delle repliche per la scalabilità orizzontale - **replication controller** e lo **scheduler**.

La componente slave esegue il carico di lavoro. Essa comunica solo con il master e salva le informazioni di servizio, tramite il API server, nel database etcd.

Ogni componente master e slave eseguono un agente locale chiamato Kubelet. L'agente collega le varie componenti e interagisce con il demone Docker.

Infine, il Kube Proxy è la componente che gestisce il traffico di rete dell'intera infrastruttura.

Kubernetes, essendo un sistema fin dall'inizio pensato per essere componibile si può integrare bene con soluzioni di terzi parti, come per esempio: diverse soluzioni per lo storage, diversi plugin per la rete ed ecc;

* Elasticsearch, Logstash e Kibana: Le tre componenti sono comunemente conosciute con l'acronimo ELK. Esse vengono utilizzate insieme come una soluzione open source in progetti che hanno forti esigenze di ricerca e analisi di dati. Elasctisearch è il cuore dello stack applicativo. Esso è un database NoSQL e distribuito implementato in Java. Orientato all'immagazzinamento di dati non strutturati, Elasticsearch permette di effettuare ricerche complesse impiegando millisecondi contro i secondi necessari utilizzando un classico DB SQL.

  Il meccanismo di gestione dei dati di Elasticsearch è complicato. In figura è rappresentato un cluster di due nodi. Elasticsearch permette di organizzare l'insieme

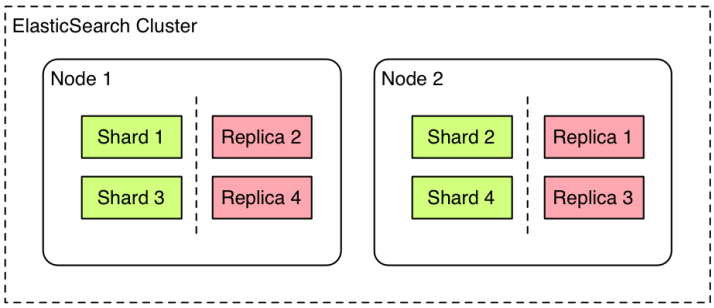


Figura 2.7: Vista di alto livello del meccanismo di partizione dei dati immagazzinati in Elasticsearch. Immagine tratta da: http://bit.ly/2r0HTQL.

complessivo di dati in sottoinsiemi (*shard*) e aggiungere un numero y di copie (*replica*) per ogni sottoinsieme primario. Una simile organizzazione permette di garantire un'alta affidabilità di dati a livello applicativo. Inoltre, in funzione ai casi d'uso Elasticsearch è altamente personalizzabile. Per incrementare le performance di lettura, l'utente incrementa il numero delle repliche. Per incrementare le performance di scrittura, l'utente diminuisce il numero di sottoinsiemi primari dei dati. Kibana, invece, è la componente dello stack che offre la funzionalità di visualizzazione dei dati presenti in Elasticsearch. Una peculiare caratteristica di Kibana è l'interfaccia di creazione dei cruscotti. Kibana sfrutta l'integrazione nativa con Elasticsearch per esprimere ricerche molto complesse e renderizzarle a video tramite effetti grafici accattivanti. Infine, Logstash è la componente di estrazione, trasformazione e caricamento dei dati dalla sorgente in Elasticsearch. Con Logstash risulta semplice filtrare l'informazione utile per l'analisi dei dati ed eliminare il rumore di fondo. La comunità open source di Logstash ha sviluppato un insieme molto ricco di strumenti che estendono il suo insieme di funzionalità. Per esempio, tramite uno plugin esterno è possibile programmare Logstash a interagire con le API (Application Program Interface) del social network Twitter, per cercare, scaricare e filtrare l'informazione che soddisfa uno specifico criterio di ricerca. Dal punto di vista architetturale la soluzione ELK è flessibile e permette di scalare facilmente in orizzontale, proporzionalmente al carico di lavoro. A seguire allego una rappresentazione grafica delle dipendenze inter componenti a livello architetturale. Infatti, Kibana legge da Elasticsearch (ES) mentre Logstash scrive in ES.

Figura 2.8: Vista di alto livello del legame sussistente inter componente. Il verso della freccia nell'immagine indica una dipendenza.

Inizialmente sono state fissate anche le rispettive versioni delle componenti sopra citate. Tuttavia, nel corso dello stage ho realizzato che questo può bloccare l'evoluzione dell'infrastruttura e comportare qualche problema in futuro. A questo scopo ho

predisposto un ambiente tollerante agli aggiornamenti. Durante la personalizzazione dell'ambiente mi sono ispirato al principio *self driven infrastructure* di CoreOS. In questo modo gli aggiornamenti delle componenti possono essere effettuati in modo completamente trasparente.

Infine, come vincolo per le immagini dei container Docker mi hanno imposto di utilizzare solo le immagini ufficiali provenienti dal hub di Docker. Questo vincolo è dovuto alla presenza di vulnerabilità di sicurezza nelle immagini di terzi parti.

Capitolo 3

Lo svolgimento dello stage

3.1 Metodo di lavoro

Durante il periodo di stage, io ho avuto modo di interfacciarmi direttamente con tutto il team aziendale della BU Operation.

3.2 Attività di formazione

3.3 Analisi dei requisiti

3.3.1 Requisiti

3.3.1.1 Funzionali

3.3.1.2 Non funzionali

3.4 Progettazione e realizzazione

3.4.1 Scelte progettuali

3.4.2 Visione architetturale a microservizi

3.4.3 Codifica

3.4.4 Test

3.4.4.1 Test di carico

3.4.4.2 Test di durata

Capitolo 4

Valutazioni retrospettive

TODO: Aggiungere sintesi al capitolo

4.1 Obiettivi raggiunti

4.2 Problematiche riscontrate

4.3 Bilancio formativo

4.3.1 Il prima

4.3.2 Il dopo

4.4 Valutazione critica del Corso di Laurea

Riferimenti

Bibliografia

Baier, Jonathan. *Getting Started With Kubernetes*. PACKT, 2015.

Clinton Gormley, Zachary Tong. *Elasticsearch: The Definitive Guide*. O'Reilly Media, 2015.

Rafal Kuć, Marek Rogoziński. *Elasticsearch Server*. PACKT, 2016.

Turnbull, James. *The Docker Book: Containerization is the new virtualization*. James Turnbull, 2016.

Vohra, Deepak. *Kubernetes Microservices With Docker*. Apress, 2016.

Sitografia

Documentazione Docker. URL: https://docs.docker.com/.

Documentazione Elasticsearch. URL: https://www.elastic.co/guide/en/elasticsearch/reference/2.4/index.html.

Documentazione Kibana. URL: https://www.elastic.co/guide/en/kibana/4.6/index.html.

Documentazione Kubernetes. URL: https://kubernetes.io/docs/home/.

Documentazione Linux. URL: http://www.tldp.org/.

Documentazione Logstash. URL: https://www.elastic.co/guide/en/logstash/2.3/index.html.

Documentazione Nginx. URL: https://nginx.org/en/docs/.

Martin Fawler: Micorservices. URL: https://martinfowler.com/articles/microservices.html.

Pattern architetturale a microservizi. URL: http://microservices.io/index.html.

Wikipedia: Cloud computing. URL: https://en.wikipedia.org/wiki/Cloud_computing.

Wikipedia: Microservices. URL: https://en.wikipedia.org/wiki/Microservices.