SER&P
Software Engineering Research & Practices

I-MOLE

Gap Analysis

OR1.3

Spin Off dell'Università degli Studi di Bari
Via Orabona, 4 c/o Dip. di informatica 70126 - Bari
Tel: +39.080.5442048 | Fax: +39.080.5442536
www.serandp.com

INDICE

INTRODUZIONE 5

SELEZIONE DELLA PIATTAFORMA E OTTIMIZZAZIONE DELLA INFRASTRUTTURA INFORMATICA 9

1.1 INTRODUZIONE E DEFINIZIONI 9
1.2 SELEZIONE DELLA PIATTAFORMA INFORMATICA 12
1.3 OTTIMIZZAZIONE DELLA INFRASTRUTTURA INFORMATICA 14
1.4 MATRICI DI SELEZIONE 16
1.4.1 Matrice Architettura/Hypervisor 16
1.4.2 Matrice Sistema Operativo/Hypervisor 16
1.4.3 Matrice Cloud Manager/Hypervisor 17

IOT: PROTOCOLLI E STRUMENTI DI GESTIONE 18

1.5 COMPARAZIONE DEI PROTOCOLLI E CONSIDERAZIONI 23
1.6 MESSAGE-ORIENTED MIDDLEWARE 24
1.6.1 Broker presi in considerazione 26
1.6.2 Enterprise Integration Patterns (EIP) 29
1.6.3 Scelta del broker: conclusioni preliminari 30

INTERFACCIA I-MOLE 35

1.7 FRAMEWORK 37
1.7.1 React 37
1.7.2 Vue.js 38
1.7.3 Angular 39
1.7.4 Preact 39
1.7.5 Ember 40
1.7.6 Svelte 40
1.7.7 Blazor 40
1.8 CLASSIFICHE E BENCHMARK 41
1.9 PRESTAZIONI DEI FRAMEWORK DI FRONTEND 41
1.9.1 Diffusione / progetti 42
1.10 ANGULAR 2 VS REACT.JS 49
1.10.1 Livello di maturità Angular 51
1.10.2 Livello di maturità React 51

Software Engineering Research & Practices

1.10.3 Ricapitolando 51
1.11 CARATTERISTICHE 51
1.11.1 Angular 51
1.11.2 React 52
1.12 LINGUAGGI, PARADIGMI E PATTERN 54
1.12.1 React 54
1.12.2 Angular 55
1.13 ECOSISTEMA 58
1.13.1 Angular 58
1.13.2 React 60
1.14 ADOZIONE 64
1.14.1 React 64
1.14.2 Angular 65
1.14.3 Conclusioni 65
1.15 COMMUNITY FEEDBACK 66
1.16 SCELTA DEI FRAMEWORK 66

BASI DI DATI 69

1.17 DATABASE MANAGEMENT SYSTEM (DATABASE NOSQL) 74
1.17.2 Caratteristiche dei database NoSQL 76
1.17.3 MongoDB 80

BLOCKCHAIN 85

1.18 SMART CONTRACT SU ETHEREUM 87
1.18.1 Solidity 87
1.18.2 Truffle framework 87
1.18.3 Web3js 88
1.18.4 Ganache-CLI 88
1.18.5 MetaMask 88
1.19 CONCLUSIONE 88

BIG DATA 90

1.19.1 Apache Hadoop 93

WORKFLOW MANAGEMENT SYSTEM 102

Software Engineering Research & Practices

STORIA DEL DOCUMENTO

DATA	AUTORE	CONTENUTI & MODIFICHE
Novembre 2018	Responsabile di progetto	Canovaccio del TR e piano delle attività
Novembre 2018	Galati Giuseppe Giovanelli Alberto Murabito Rosa	Redazione capitolo 3
Novembre 2018	Attanasio Gaetano Piccolella Antonio Mascaro Mikias Salvatore	Redazione capitolo 4
Novembre 2018	Responsabile di Progetto + autori dei capitoli 3 e 4	Fusione dei capitoli 3 e 4
Novembre 2018	Responsabile di Progetto + autori dei capitoli 3 e 4	Cap. 3e4- Verifica e validazione-evidenza di eventuali rilievi. Revisione del canovacci.
Novembre 2018	Responsabile di Progetto + autori del capitolo 3	Superamento dei rilievi emersi dalla V&V e Redazione finale del cap. 3
Novembre 2018	Responsabile di Progetto + autori del capitolo 4	Superamento dei rilievi emersi dalla V&V e Redazione finale del cap. 4
Dicembre 2018	Attanasio Gaetano Avino Benedetta Buttari Carmine Galati Giuseppe Mascaro Mikias Salvatore Minniti Federica Natale Marco Sobrero Luis Alberto	Redazione capitolo 5
Dicembre 2018	Giovanelli Alberto Carazzato Marco Venturini Gianmarco	Redazione capitolo 6

Software Engineering Research & Practices

	Experto credite sas Murabito Rosa Palumbo Nicola Piccolella Antonio Speciale Francesco	
Dicembre 2018	Responsabile di Progetto + autori dei capitoli 5 e 6	Fusione dei capitoli 5 e 6
Dicembre 2018	Responsabile di Progetto + autori del capitolo 5	Cap. 5- Verifica e validazione- evidenza di eventuali rilievi
Dicembre 2018	Responsabile di Progetto + autori del capitolo 6	Cap. 6- Verifica e validazione- evidenza di eventuali rilievi
Dicembre 2018	Responsabile di Progetto + autori del capitolo 5	Superamento dei rilievi emersi dalla V&V e Redazione finale del cap. 5
Dicembre 2018	Responsabile di Progetto + autori del capitolo 6	Superamento dei rilievi emersi dalla V&V e Redazione finale del cap. 6
Dicembre 2018	Giovanelli Alberto Carazzato Marco Venturini Gianmarco Experto credite sas Murabito Rosa Palumbo Nicola Piccolella Antonio Speciale Francesco	Redazione capitolo 7
Dicembre 2018	Responsabile di Progetto + autori del capitolo 7	Cap. 7- Verifica e validazione- evidenza di eventuali rilievi
Dicembre 2018	Responsabile di Progetto + autori del capitolo 7	Superamento dei rilievi emersi dalla V&V e Redazione finale del cap. 7
Dicembre 2018	Ser&P+UNIBA+ Responsabile progetto	Revisione del documento sulla base delle evidenze emerse durante l'avanzamento del progetto – Fusione, redazione

Software Engineering Research & Practices

		e pubblicazione del Rapporto finale.

Software Engineering Research & Practices

INTRODUZIONE

La Gap Analisys è una analisi qualitativa volta a identificare i gap rispetto al modello architetturale ipotizzato, ai processi di funzionamento, ai possibili utilizzatori della soluzione. Attraverso tale analisi si intende, inoltre, valutare gli interventi necessari per la realizzazione delle soluzioni identificate e a identificare i benefici attesi, nonché i vincoli e le criticità per la realizzazione della soluzione.

In considerazione dei risultati emersi dalla comparazione tra stato dell'arte e della pratica, si è proceduto ad individuare le soluzioni tecnologiche che potranno essere utilizzate per la produzione dei dimostratori previsti dal progetto. Le tecnologie e le componenti di riferimento oggetto della presente Gap Analysis riguardano gli elementi cardine del progetto che sono:

1. **Infrastruttura informatica.** Soluzioni infrastrutturali da utilizzarsi per ospitare la piattaforma software i-mole e le sue componenti.
2. **Gestione IoT.** Protocolli e strumenti per la gestione della iterazione tra i dispositivi IoT e la piattaforma i-mole.
3. **Interfaccia utente.** Strumenti da utilizzarsi per la realizzazione delle interfacce utilizzate da i-mole per interagire con l'esterno.
4. **Basi di dati** gestori di dati da utilizzarsi per il progetto.
5. **Blockchain**. Sono state analizzare e scelte sia la soluzione infrastrutturale da utilizzarsi per la realizzazione della blockchain di progetto, sia le soluzioni ed i linguaggi per la realizzazione degli smart contract.
6. **Big-Data**. Strumenti, linguaggi e applicativi per la gestione dei Big Data e per la gestione delle attività di Data Mining.
7. **Workflow management system**. Valutazione e selezione dei motori di WFMS per la realizzazione della componente di workflow prevista da i-mole che si deve occupare di orchestrare i processi di business che si definiranno con la realizzazione dei dimostratori.

Tra le soluzioni elencate ve ne sono alcune che potranno essere direttamente impiegabili, mentre altre richiederanno opportune estensioni ed integrazioni.

Software Engineering Research & Practices

SELEZIONE DELLA PIATTAFORMA E OTTIMIZZAZIONE DELLA INFRASTRUTTURA INFORMATICA

Con questi termini si suole indicare i processi volti a stabilire quale sia l'infrastruttura e la piattaforma informatica "migliore" per eseguire una data applicazione. Nel fare ciò potrebbe essere utile affidarsi ad un approccio rigoroso e quantitativo basato sulle caratteristiche non funzionali.

1.1 Introduzione e definizioni

Si definisce **Infrastruttura Informatica** l'insieme degli apparati hardware, del software, delle strutture di rete e collegamenti che implementano lo scenario in cui è possibile generare e gestire tutte le piattaforme informatiche necessarie a fornire l'ambiente di esecuzione nel quale convivono applicazioni e dati. Spesso il disegno di una infrastruttura informatica rappresenta un processo continuo di adattamento di situazioni attuali e prospettive future. Tuttavia, alcuni paradigmi, quali flessibilità ed affidabilità della struttura informatica, costituiscono punti fermi del progetto di una infrastruttura informatica: quest'ultima non deve mai rappresentare un ostacolo alla dinamicità dei processi di business che creano il vantaggio competitivo tra le aziende.

Si definisce **Piattaforma Informatica** l'insieme di un calcolatore e di un sistema operativo. Tale termine consente di definire uno scenario elaborativo identificando le caratteristiche salienti dell'hardware e del software che su di esso opera. Tali definizioni ci permettono di identificare con precisione una Piattaforma Informatica come una coppia di un **Sistema Operativo** (OS) e di una **Architettura Hardware**, considerando che a volte lo stesso OS (o quanto meno OS con lo stesso nome e le stesse funzionalità) può funzionare su architetture differenti, ovvero che la stessa architettura può ospitare sistemi operativi diversi. Facendo riferimento a tecnologie quali la virtualizzazione è possibile definire, in maniera più precisa, una Piattaforma Informatica come una coppia di un Sistema Operativo e di una **Architettura Virtuale**, considerando questa come composta da un'architettura hardware sottostante atta a supportare un VMM o Hypervisor.

Il concetto di piattaforma informatica risulta particolarmente utile tutte le volte nelle quali si vogliano identificare e confrontare ambienti diversi in grado di ospitare un'applicazione informatica. Con questa definizione un'applicazione informatica potrà essere eseguita/ospitata (hosted) su

Software Engineering Research & Practices

diverse piattaforme informatiche e quindi, a parità di applicazione avrà un senso il confronto e la selezione tra le piattaforme possibili. La figura seguente rappresenta il concetto di Piattaforma Informatica per come qui definito: come si diceva l'architettura hardware denominata z/Architecture potrà ospitare OS diversi, in questo caso z/OS e Linux. Nella denominazione della piattaforma informatica si conviene di mettere al primo posto il nome del Sistema Operativo seguito da quello dell'Architettura Hardware: ci sarà così ad esempio la Piattaforma Linux/x86 ovvero Linux/zArchitecture ecc.

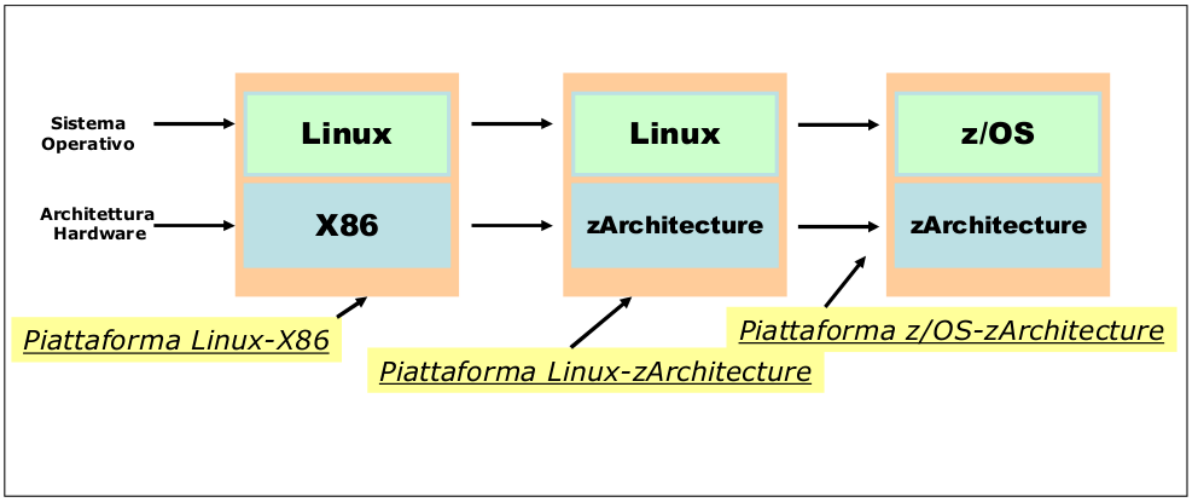


Figura 1 - Piattaforma Informatica

Osserviamo tuttavia che alcuni OS possono essere attivati solo su una Architettura Hardware (ad esempio z/OS può funzionare solo su macchine di z/Architecture), parimenti non tutte le combinazioni (coppie ordinate) di un OS ed una Architettura Hardware danno luogo ad una Piattaforma Informatica esistente (ad esempio la Piattaforma Informatica MS Windows/zArchitecture non esiste).

Una **Applicazione Informatica** è un insieme di manufatti software (programmi, procedure, dati) in grado di svolgere un lavoro utile attraverso un sistema di elaborazione. Una applicazione informatica può essere eseguita su una o più piattaforme informatiche differenti. Una Applicazione Informatica si dirà Tipica o caratteristica di una sola Piattaforma Informatica, quando essa può essere eseguita su una ed una sola Piattaforma Informatica: in tale caso essa viene comunemente detta "Legata" a quella Piattaforma Informatica [Legacy]. Una Applicazione Informatica si dirà portabile attraverso una lista di Piattaforme Informatiche nel caso in cui essa possa essere eseguita su una serie di diverse Piattaforme Informatiche [Open]. Non

Software Engineering Research & Practices

è detto che una Applicazione Informatica portabile possa essere eseguita su tutte le Piattaforme Informatiche possibili.

Per meglio comprendere quest'ultima affermazione occorre analizzare quali sono gli elementi che realizzano la portabilità della Applicazione Informatica attraverso le diverse Piattaforme Informatiche. Un elemento certamente essenziale può essere rappresentato dal linguaggio di programmazione utilizzato, o dal tipo di codifica dei dati usato o dalla presenza di un sottosistema (Middleware) che si interpone tra l'applicazione ed il sistema operativo.

In base a questi elementi avremo quindi:

- Applicazioni portabili a livello di codice sorgente – si tratta delle applicazioni scritte nello stesso linguaggio di programmazione, le quali sono portabili a meno di ricompilazione o reinterpretazione.
- Applicazioni portabili a livello di codice eseguibile (portabilità binaria) nel caso in cui attraverso la presenza di un sottosistema, di un Middleware o di un emulatore lo stesso codice compilato potrà essere utilizzato sottosistemi operativi diversi; in questo caso il processo di adattamento tra il codice compilato ed il sistema operativo sarà effettuato dal middleware o dall'emulatore.

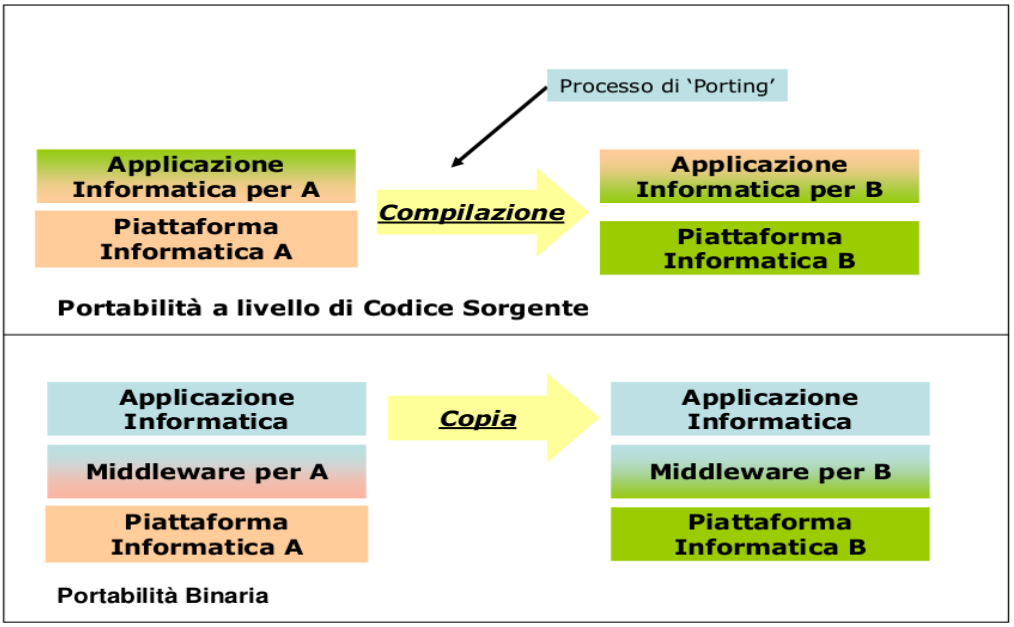


Figura 2 - Portabilità dell'Applicazione Informatica

Da quanto detto si intuisce facilmente che la condizione di portabilità di una applicazione tra diverse piattaforme dipende innanzitutto dalla portabilità dell'applicazione attraverso gli OS e solo in minima parte dall'architettura

Software Engineering Research & Practices

hardware. Quest'ultima è usualmente "schermata" dal Sistema Operativo ed interviene solo su elementi molto profondi della applicazione come, ad esempio, il formato di rappresentazione dei dati.

Così come la presenza di uno strato intermedio (Middleware) facilita la portabilità dell'applicazione attraverso diversi sistemi operativi e quindi diverse piattaforme informatiche, anche gli Standard costituiscono un elemento utile alla portabilità: gli standard de iure o de facto, come ad esempio TCP/IP, XML, Java, J2EE, si aggiungono ai middleware (Apache, WebSphere, TomCat, ecc.) nel facilitare la possibilità di esecuzione (deploy) dell'applicazione su diverse piattaforme con lo stesso risultato funzionale e senza modifiche al codice.

L'ambiente costituito dal Sistema Operativo e dalla Architettura Hardware qui definito come "Piattaforma Informatica", unito all'insieme degli strumenti di ausilio alla gestione operativa e sistemistica prende il nome di Ambiente Operativo (Operating Environment).

Grazie alla portabilità ha quindi senso definire un processo di selezione della piattaforma informatica, in quanto la stessa applicazione, con modifiche piccole o nulle al codice, potrà essere eseguita sotto diversi sistemi operativi, o più frequentemente sotto lo stesso sistema operativo e diverse architetture hardware, con diverse caratteristiche (es. prestazionali o di costo). Il processo di selezione della piattaforma definisce implicitamente anche il relativo ambiente operativo e quindi più in generale le caratteristiche di gestibilità complessiva della applicazione informatica.

1.2 Selezione della Piattaforma Informatica

Si definisce **Selezione della Piattaforma Informatica** (o **Platform Selection**) il processo atto ad individuare la piattaforma informatica più idonea ad eseguire una data Applicazione Informatica in relazione ad una serie di caratteristiche non funzionali assegnate (es. le prestazioni, le caratteristiche di gestibilità operative, i costi di esercizio, continuità del servizio, sicurezza, ecc.). E' bene precisare che si sta parlando di variazione dei requisiti che non attengano esplicitamente al funzionamento della applicazione, ma che possono solo variarne alcuni aspetti (migliorarne o peggiorarne il funzionamento). I requisiti funzionali, al contrario, essendo legati alle funzioni che l'applicazione deve svolgere, non possono essere in alcun modo negoziabili: essi sono parte integrante della applicazione stessa.

Software Engineering Research & Practices

La Platform Selection ha senso ed è possibile solo in funzione del fatto che la portabilità dell'applicazione possa essere fattibile. Non sempre l'applicazione risulta "portabile" in quanto a volte essa presenta legami stretti con la piattaforma sulla quale è stata costruita ("legacy").

Il processo di Platform Selection viene generalmente eseguito a posteriori su applicazioni già esistenti ed in uso con lo scopo di ottimizzarne l'Operating Environment. Ciò non toglie che tale processo possa essere eseguito a priori ovvero in fase di progettazione e rilascio di una nuova applicazione. In questo caso esso assumerà il ruolo di analisi preventiva allo scopo di determinare a priori quale possa essere l'ambiente operativo più opportuno per eseguire l'applicazione una volta completata. Sempre più spesso questo secondo modo di operare a priori si sta diffondendo presso gli sviluppatori ed i progettisti, fornendo tra l'altro l'opportunità di selezionare gli strumenti di sviluppo ed i Middleware usati rispetto ai criteri di portabilità e standard piuttosto che la conoscenza o popolarità degli strumenti stessi.

La Platform Selection deve essere mediata da valutazioni strategiche ed organizzative complessive. Tra gli elementi non funzionali più importanti e quindi determinanti ai fini della Platform Selection possiamo annoverare:

1. I costi di esercizio dell'Applicazione Informatica in relazione alla Piattaforma Informatica analizzata.
2. L'ambiente operativo (Operating Environment) tipico della Piattaforma Informatica analizzata.
3. La presenza in Azienda/Ente di personale con esperienza specifica (Skills) su quella data Piattaforma.
4. Le prestazioni specifiche della Applicazione su quella piattaforma.
5. I Costi di Acquisizione di Hardware e Software

Nel caso in cui un'applicazione esistente e portabile venga trasportata su una diversa piattaforma informatica solo in base a criteri non funzionali (tipicamente una riduzione dei costi operativi) si suole parlare di processo di "re-hosting". Il re-hosting è quindi un processo di porting di un'applicazione da una piattaforma ad un'altra allo scopo di migliorarne alcuni aspetti non funzionali. Nella realtà i processi di re-hosting tendono ad essere eseguiti minimizzando l'impatto sul codice: ciò può essere reso possibile grazie ad opportuni strumenti o alla presenza di middleware. La valutazione di un processo di re-hosting è materia sufficientemente complessa: questo processo deve tenere in considerazione delle buone pratiche ed elementi

Software Engineering Research & Practices

quantitativi in grado di fornire valutazioni oggettive a seguito dell'individuazione dei requisiti non funzionali maggiormente rilevanti.

Mentre, nel caso di applicazione esistente ed ove questa presentasse legami (legacy) con la piattaforma informatica, segnatamente con il sistema operativo, si può valutare un processo di "legacy transformation" volto a ridefinire o riscrivere l'applicazione usando strumenti che ne garantiscono la portabilità e la maggior aderenza agli standard.

1.3 Ottimizzazione della Infrastruttura Informatica

L'Infrastruttura Informatica è l'ambiente di esecuzione di una o più Applicazioni Informatiche. Fanno quindi parte della Infrastruttura Informatica:

- I Serventi (Elaboratori Elettronici) con il loro software di base (Hypervisor o Sistemi Operativi) e Middleware
- Apparati di memorizzazione (SAN, NAS, ecc.)
- Apparati di comunicazione (Switch, Router, Firewall, ecc.)
- Altri apparati hardware
- I Dati gestiti dall'Applicazione

Per come qui definita, nell'Infrastruttura Informatica potranno essere presenti una o più Piattaforme Informatiche diverse e, di conseguenza, una Infrastruttura Informatica potrà supportare l'esecuzione di una o più Applicazioni Informatiche.

A causa del rapido sviluppo delle Applicazioni Informatiche e della Rete, negli ultimi decenni l'Infrastruttura Informatica si è notevolmente complicata, tanto che oggi essa è diventata un elemento di grande complessità e criticità. Il modello applicativo "Client Server" ha creato una proliferazione abbastanza incontrollata di serventi, che venivano installati in grande numero ed in locazioni differenti. Lo stesso modello applicativo ha introdotto il paradigma secondo il quale ad ogni funzione debba corrispondere un servente. La proliferazione dei serventi (Server Proliferation) non è l'unico problema riscontrabile in una infrastruttura informatica. Lo stesso modello applicativo è anche causa di uno scarso utilizzo dei sistemi serventi (Under Utilization): i sistemi con il tempo sono diventati sempre più potenti e di conseguenza dovendo e potendo svolgere una sola funzione sono diventati sempre meno utilizzati. Altri problemi correlati riguardano ad esempio la Qualità del Servizio e la Resilienza. Sembra abbastanza intuitivo che in entrambi i casi ridurre il numero di

Software Engineering Research & Practices

componenti discreti che compongono una infrastruttura informatica, riducendone quindi la complessità, migliora tali caratteristiche. A queste problematiche, infine, si aggiungono anche esigenze di tipo ambientale legate al contenimento dei consumi energetici, della necessità di raffreddamento e alla carenza di spazio (Green Computing).

Detto questo, il processo di ottimizzazione della infrastruttura informatica può avvenire intervenendo su fattori ed elementi differenti. Gli elementi più comunemente presi in considerazione sono:

- Efficienza: si può intervenire sull'infrastruttura cercando di migliorarne l'efficienza, indipendentemente dagli altri fattori.
- Gestibilità: si può intervenire sull'Infrastruttura solo per renderla più gestibile.
- Costi: si può operare sull'Infrastruttura al fine di diminuirne i costi operativi e di acquisizione.
- Resilienza o Continuità di Servizio: si può operare sulla infrastruttura solo per migliorarne le capacità di essere ricoverata in caso di disastro o la sua flessibilità operativa.
- Consumi Energetici: prende sempre più consistenza la valutazione degli impatti ambientali come elemento per l'ottimizzazione.

Una volta determinati gli elementi su cui porre l'attenzione è necessario individuare le azioni da intraprendere o da valutare. Ad esempio, azioni possibili possono essere:

- diminuire il numero di serventi;
- consolidare gli apparati di memorizzazione;
- diminuire il numero di istanze di sistema operativo;
- virtualizzare i sistemi;
- virtualizzare le reti.

Le azioni sopra indicate, che peraltro non sono le uniche possibili, possono essere facilitate da alcune tecnologie, che diventano abilitanti ai fini del processo di ottimizzazione. Tra gli elementi abilitanti è bene sottolinearne alcuni che sono particolarmente funzionali a questa trattazione:

- La Virtualizzazione, in quanto consente di ridurre il numero di Serventi Fisici e spesso anche il numero di quelli logici (Server Consolidation)
- L'uso di Grandi Sistemi Centrali al posto di molti serventi distribuiti; tale uso, abbinato alla virtualizzazione, può consentire grossi recuperi di efficienza complessiva.

Software Engineering Research & Practices

- Gli strumenti di Cloud Computing volti a migliorare la gestibilità, la resilienza e i costi dell'infrastruttura.

1.4 Matrici di selezione

A sostegno dei processi di selezione ed ottimizzazione visti in precedenza, di seguito vengono riportate delle matrici di corrispondenza tra tutti quelli che sono gli elementi necessari alla realizzazione di una infrastruttura informatica.

1.4.1 Matrice Architettura/Hypervisor

Di seguito viene riportata la matrice di corrispondenza tra le architetture hardware e i prodotti di virtualizzazione visti in precedenza.

	HYPER-V	IBM z/VM	KVM	VMware	XEN
x86 Architecture	X		X	X	X
ARM Architecture			X		X
z/Architecture		X			

Contestualizzando questa matrice di selezione al progetto i-mole, rispetto alla tipologia di architetture è possibile escludere l'architettura di tipo z/Architecture non utilizzabile e di conseguenza anche la soluzione IBM z/VM. Mentre rimangono utilizzabili le altre soluzioni: HYPER-V, KVM, VMware, XEN.

1.4.2 Matrice Sistema Operativo/Hypervisor

Di seguito viene riportata la matrice tra i prodotti di virtualizzazione e i corrispondenti sistemi operativi supportati.

	HYPER-V	IBM z/VM	KVM	VMware	XEN
GNU/Linux	X	X	X	X	X
Microsoft Windows	X		X	X	X

Software Engineering Research & Practices

Reiterando la scelta delle soluzioni con la matrice di comparazione relativa al sistema operativo è possibile eliminare dalla scelta le soluzioni che possono essere attuate su IBM z/VM.

1.4.3 Matrice Cloud Manager/Hypervisor[1]

Di seguito viene riportata la matrice di corrispondenza tra gli Hypervisor e i prodotti Cloud Manager visti in precedenza.

	HYPER-V	IBM z/VM	KVM	VMware	XEN
Cloud Stack			X	X	X
Eucalyptus			X	X	X
OpenNebula			X	X	X
OpenStack	X		X	X	X
TSAM		X	X	X	X
vCD				X	

Infine, grazie alla matrice di comparazione Cloud Manager vs Hypervisor è stato possibile restringere il campo alla soluzione di cloud manager utilizzabile che risulta essere OpenStack.

[1] Fonte: Open Source IaaS Clouds: An Overview- April 2016International Journal of Applied Engineering Research 11(1):22-27

Software Engineering Research & Practices

IoT: PROTOCOLLI E STRUMENTI DI GESTIONE

Uno dei punti di attenzione parlando di soluzione IoT è relativo alla numerosità delle tipologie dei dispositivi IoT, presenti sul mercato, che una soluzione deve consentire di connettere. La necessità di disporre di un sistema che riesca ad integrarsi con il più alto numero possibile di dispositivi IoT nasce, come detto in precedenza, dalle previsioni di crescita e di diffusione dei dispositivi utilizzabili e quindi dei dati descriventi la realtà raccoglibili. Difatti, le previsioni ad oggi effettuate dalle più importanti agenzie e aziende del settore, tra cui CISCO e IHS, inerentemente ai dispostivi IoT parlano di un mercato di circa 50 miliardi di dispostivi connessi già nel 2020.

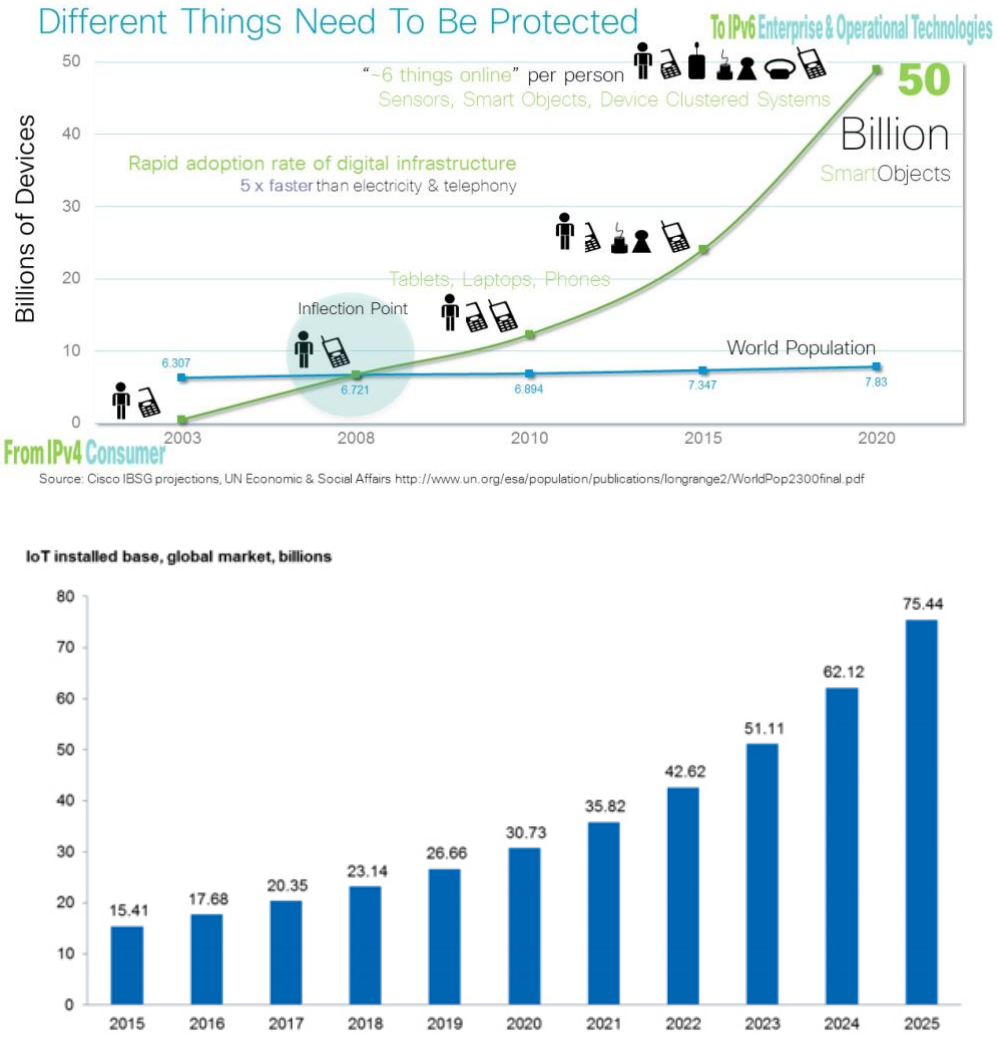
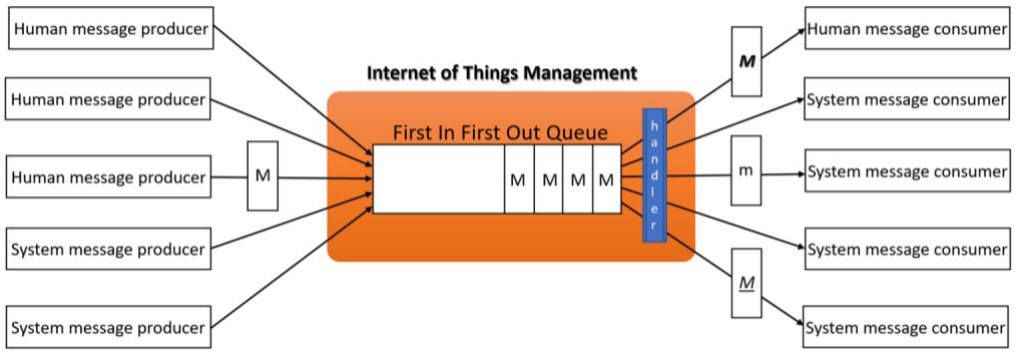


Figura 3 Over the course of a decade, the Internet of Things could jump from 15.4 billion devices to 75.4 billion. Source: IHS

Software Engineering Research & Practices

È utile ricordare che mai come oggi ci si è trovati nelle condizioni di poter raccogliere una tale quantità di dati da una tale numerosità di dispositivi e questo apre le porte a nuovi mercati di riferimento e fornisce materiale prezioso per una serie di ricerche non effettuabili su dataset limitati. Relativamente al secondo punto, basti pensare che con i dati raccolti, da smartwatch, su soli 14.000 soggetti è stato possibile identificare dei marker per rilevare il diabete molto prima che questi si manifesti. Detto questo, è facile pensare che uno strumento in grado di raccogliere quanti più dati possibili dai dispositivi IoT, creerebbe il presupposto per analizzare i dati raccolti (big data) da molti punti di vista e consentirebbe di scoprire molte altre relazioni come quella del diabete.

Un primo punto di decisione riguarda la logica con cui devono essere presi in carico i messaggi inviati dai dispositivi IoT collegati, in altri termini se arrivano 5 messaggi quale deve essere gestito prima: il più grosso? il più piccolo? e così via. Fortunatamente, nel contesto I-MOLE la risposta migliore, non essendoci al momento altri criteri discriminanti, è la gestione FIFO (First In First Out) dei messaggi. Il metodo FIFO rappresenta la modalità di immagazzinamento in cui il primo oggetto introdotto è il primo ad uscire.



Un altro aspetto da considerare riguarda la sostenibilità dei costi di trasmissione dei dati raccolti dai dispositivi collegati, difatti, alla prospettiva teorica rosea di disporre di un numero infinito di dati da cui discernere informazioni ad oggi sconosciute si contrappone l'aspetto pratico della sostenibilità della soluzione, difatti nelle applicazioni IoT i dati trasferiti tra i dispositivi e il cloud rappresentano un importante fattore di costo. Soprattutto quando i dispositivi sono collegati su una rete mobile, il trasferimento dei dati è piuttosto costoso. Inoltre, quando si tratta di molti dispositivi, anche il traffico sul sito cloud potrebbe avere un impatto sui

Software Engineering Research & Practices

costi. Pertanto, è fondamentale pensare alla dimensione del payload del messaggio.

Per questo nuovo problema di sostenibilità economica, il primo passo da compiere è quello di trovare un protocollo di comunicazione adatto e compatto pescando tra i due più utilizzati ovvero il protocollo basato su scambio di messaggi ed il protocollo basato su scambio di byte.

1. Message-stream-oriented protocol. L'UDP è un protocollo di trasporto orientato ai messaggi. UDP trasporta un flusso di datagrammi UDP. Ogni datagramma UDP contiene un messaggio (ad esempio, un messaggio dal protocollo SIP o un messaggio dal protocollo CoAP nell'IoT). Un datagramma UDP può contenere anche un blocco dati speciale, che può anche essere considerato come una sorta di messaggio. TCP è un protocollo orientato alla connessione quindi prima di iniziare la trasmissione dei dati i due host si scambiano dei pacchetti su come fare la comunicazione e sarà cura del TCP farsi carico di portare i pacchetti alla destinazione, ritrasmettendoli se necessario e riordinandoli prima di consegnarli.
2. Byte-stream-oriented protocol. Il TCP è un protocollo di trasporto orientato ai byte. Nel caso di TCP, viene trasportato un flusso di pacchetti TCP e ogni pacchetto TCP contiene una sequenza di byte di dati. Nell'UDP a contrario del TCP il concetto di connessione non c'è, ogni pacchetto fa storia a sé, se viene ricevuto bene, se non viene ricevuto non importa sarà compito dei protocolli di più alto livello gestire questo. UDP è quindi molto più leggero di TCP. È inutile usare un protocollo pesante come TCP sempre anche se è ovvio la maggior parte dei servizi ha bisogno di una certezza di trasmissione e quindi di TCP.

In questa ottica di contenimento dei costi e dei payload scambiati il protocollo ISO standard (ISO/IEC PRF 20922) MQTT (MQ Telemetry Transport or Message Queue Telemetry Transport) risulta essere una buona soluzione al problema, poiché l'overhead è molto piccolo rispetto ad HTTP. Tuttavia, il payload stesso non è definito dai protocolli di comunicazione, quindi di seguito esamineremo le differenti dimensioni dei pacchetti payload HTTP e MQTT trascurando l'overhead del protocollo. Esistono due opzioni comuni per i payload dei messaggi IoT: JSON (nella nostra architettura identificato come (MOM Hight level) e protocollo byte (MOM Low Level).

Software Engineering Research & Practices

Cominciamo col dire che mentre JSON è un protocollo standard e piuttosto compatto, un protocollo byte di basso livello di solito è progettato per il caso d'uso specifico e quindi è l'opzione più efficiente, che d'altra parte presenta alcuni inconvenienti.

Per effettuare un confronto tra l'occupazione delle due tipologie di protocollo possiamo utilizzare un esempio tradizionale di codifica di informazioni [36] di un semplice dispositivo IoT come una lampadina. La luce deve trasferire il suo numero seriale, la temperatura, la potenza, il livello debole e lo stato di corrente. Quindi abbiamo i seguenti 5 valori:

- Numero di serie: stringa con 12 caratteri (ad es. LXA34-691E90)
- Temperatura: (ad es. 23,5 ° C)
- Potenza: (ad es. 20 W)
- Livello di attenuazione: (ad es. 50%)
- On: booleano (ad es. Vero)

La rappresentazione JSON potrebbe apparire così:

```
{
    "serialNumber": "LXA34-691E90",
    "temperatura": 23,5,
    "potenza": 50,5,
    "dimLevel": 75,
    "on": vero
 }
```

Quando si rimuovono tutti gli spazi e le interruzioni di riga, ciò porta a una dimensione complessiva del messaggio di 87 byte.

Contrariamente alla rappresentazione JSON, possiamo anche progettare un protocollo byte specifico. I primi 12 byte sono riservati per i caratteri del numero seriale, seguiti da 4 byte per il galleggiante di temperatura, 4 byte aggiuntivi per il galleggiante di potenza, 2 byte per il livello attenuato senza segno corto e 1 byte per il valore booleano, mentre true è decodificato come 1 e falso come 0.

I byte codificati esadecimali, utilizzando per semplicità la serializzazione fornita dal "Google Protocol Buffers", per gli stessi valori di JSON sono indicati di seguito:

```
4C 58 41 33 34 2D 36 39 31 45 39 30 41 B8 00 00 42 48 00 00 00 4B 01
```


Software Engineering Research & Practices

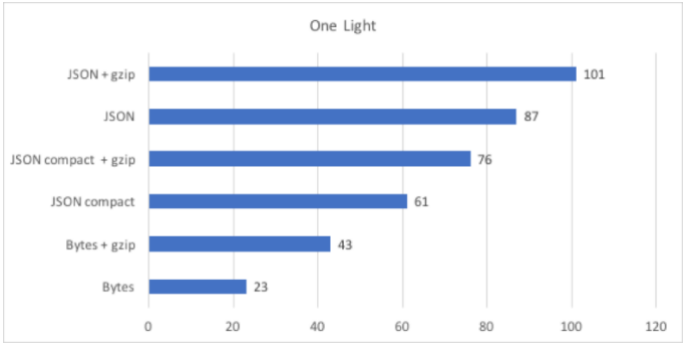
Nella versione binaria il messaggio ha una dimensione di 23 byte. Ciò significa che il payload in byte è di circa 4 volte più piccolo del payload JSON.

È chiaro che i risultati possono essere migliorati effettuando delle compressioni dei messaggi inviati, anche se questo amplifica ulteriormente il divario dispositivi presenti sul mercato e dispositivi in grado di produrre quanto richiesto (JSON compresso o Byte message compresso). Per chiarezza, un messaggio low byte può essere generato da dispositivi con bassa capacità di elaborazione e basso spazio di memoria mentre la generazione di un messaggio JSON richiede maggiori risorse a cui aggiungere anche le risorse necessarie per la compressione e tutto questo comporta quindi maggiori costi di produzione.

Nei fatti quindi adottando questa soluzione JSON O BYTE con compressione il numero di dispostivi integrabili sarebbe limitato rispetto al numero di dispostivi totali presenti sul mercato per questo motivo pur riportando il confronto tra i due la compressione non è una capacità prevista nel nostro protocollo anche perché per il progetto non abbiamo particolari problemi di costi relativi alla trasmissione dei messaggi.

Il protocollo di compressione scelto per l'esempio ed il relativo confronto è gzip poiché è ampiamente diffuso e offre buoni risultati per diversi tipi di dati.

Ritornando all'esempio precedente possiamo rilevare che nei dati riportanti non c'è quasi nessuna ripetizione nell'intero messaggio, né nelle chiavi, né nei valori. Per questo motivo la compressione non può sostituire nulla in modo efficace, ma comporta costi aggiuntivi. Il gzip del payload comporta 101 byte (14 byte in più) per il payload JSON e 43 byte (20 byte in più) per il payload dei byte.



Software Engineering Research & Practices

Quindi una conclusione importante è già che la compressione non porta alcun vantaggio se vengono inviati molti valori diversi per un dispositivo. La compressione stessa comporta anche un sovraccarico nella produzione e nell'elaborazione del carico utile.

1.5 Comparazione dei protocolli e considerazioni

Nella tabella seguente si riportano le specifiche formali dei protocolli presi in esame. Va notato che sono presenti delle eccezioni per quanto riguarda alcune delle caratteristiche prese in esame e riportate per ogni riga nella tabella seguente. Ad esempio, sebbene MQTT non offra caratteristiche di sicurezza integrate, queste possono essere inserite a livello di applicazione.

	MQTT	MQTT-SN	CoAP	AMQP	STOMP	HTTP / RESTful
Model	MOM pub/sub	MOM pub/sub	RESTful	MOM	MOM	RESTful
Discovery protocol	No	Yes (via gateways)	Yes	No	No	Yes
Resource demands	Low	Very Low	Very Low	High	Medium	Very High
Header Size (bytes)	2	2	4	8	8	8
Average power usage	Lowest	Low	Medium	High	Medium	High
Authentication	No (SSL/TLS)	No (/TLS)	No (DTLS)	Yes	No	Yes (TLS)
Encryption	No (SSL/TLS)	No (SSL/TLS)	No (DTLS)	Yes	No	Yes (TLS)
Access controls	No	No	No (proxy)	Yes	No	Yes
Communication overhead	Low	Very Low	Very Low	High	High, verbose	High

Software Engineering Research & Practices

Protocol complexity	Low	Low	Low	High	Low	Very High
TCP/UDP	TCP	TCP/UDP	UDP	TCP/UDP	TCP	TCP
Broadcasting	Indirect	Indirect	Yes	No	No	No
Quality of Service	Yes	Yes	With CON messages	Yes	No	No

MQTT, CoAP e HTTP sono di gran lunga i protocolli Internet IoT predominanti con il supporto di quasi tutti i provider cloud. MQTT e MQTT-SN forniscono un modello di comunicazione pub/sub scalabile ed efficiente mentre CoAP fornisce tutte le funzionalità rilevanti di un modello HTTP RESTful senza l'overhead intrinseco del protocollo HTTP. Quando si opera una scelta è necessario considerare l'overhead, la potenza, la larghezza di banda e le risorse necessarie per supportare un determinato protocollo e avere una visione sufficientemente lungimirante per garantire la scalabilità della soluzione.

1.6 Message-Oriented Middleware

Indipendentemente dal protocollo scelto per far comunicare i dispositivi IoT che si utilizzeranno con la soluzione i-mole è necessario adottare uno strato software denominato Message-Oriented Middlaware il cui scopo è di rendere possibile la comunicazione tra applicazioni e dispositivi tramite message-passing asincrono. Il MOM altro non è che una inftrastruttura che consente lo scambio di messaggi, secondo lo standard definito dal protocollo di comunicazione scelto, tra client e server e che prevedono meccanismi per gestire i messaggi non recapitati per indisponibilità del client memorizzando gli stessi in code che vengono utilizzate insieme a meccanismi di polling delle risorse per recapitare il messaggio al destinatario quando questo ritorna in grado di riceverli. Tutto questo è trasparente al gruppo di sviluppo che difatti non deve preoccuparsi di gestire i messaggi non recapitati.

Le soluzioni Message-Oriented Middleware presenti possono essere catalogate in due macro-categorie: i MOM commerciali e i MOM open-source. Servizi come WebSphere MQ, Amazon Simple Queue Service, Azure Service

Software Engineering Research & Practices

Bus e Cloud Pub/Sub sviluppati rispettivamente da IBM, Amazon, Microsoft e Google, permettono di eliminare la complessità e il sovraccarico di lavoro associati alla gestione e al funzionamento del Message-Oriented Middleware, consentendo agli sviluppatori di concentrarsi sul lavoro rimanente. Con questi servizi basati sul cloud è possibile inviare, memorizzare e ricevere qualsiasi volume di messaggi tra componenti software, senza perdite e senza la necessità di impiegare altro personale e altri servizi per mantenere la disponibilità. In pochi minuti è possibile creare un layer intermedio per l'integrazione di sistemi distribuiti, creare topic al quale iscrivere servizi e visualizzare le informazioni ottenute tramite web-console o interfaccia a riga di comando. Tra gli altri vantaggi da annoverare in caso di scelta di una soluzione commerciale ci sono:

- la garanzia della gestione delle operazioni e dell'infrastruttura necessaria a fornire un servizio di coda di messaggi scalabile e ad alta disponibilità; le code vengono gestite in modo dinamico, e le relative risorse calibrate accuratamente in base al carico di lavoro, al fine di consentire una rapida crescita ed alta efficienza delle applicazioni.
- l'affidabilità nella trasmissione dei messaggi. Ognuno di questi servizi può infatti trasmettere qualsiasi volume di dati a qualsiasi volume di throughput, senza perdere né messaggi o richiedere la disponibilità di ulteriori servizi. Inoltre, per ogni messaggio vengono memorizzate più copie con ridondanza su diverse zone di diponibilità, per renderle disponibili in caso di bisogno.
- costi di gestione calcolati in base all'utilizzo, garantendo, a loro dire, notevoli risparmi rispetto alle soluzioni di modelli di messaggistica gestiti manualmente.

L'alternativa ai servizi commerciali e proprietari è data dalle soluzioni di tipo open source. L'utilità delle soluzioni open source non dipende dal solo costo di licenza ma più dalle possibilità che questo consenti di sfruttare: consente in contesti esplorativi e innovativi come i-mole di inglobare nella soluzione una componente software, consolidata nel tempo e quindi stabile, di cui si ha il completo controllo. Questo è un grosso vantaggio, per un lavoro di ricerca industriale, poiché consente di esaminare il codice, controllare che il software non faccia nulla per cui non sia stato programmato, ereditare l'esperienza utente nell'uso del software, riusare le tutto il codice di interesse ed eliminare le parti di codice non utilizzate ottenendo così una reale risorsa utilizzabile nell'ottica delle linee di prodotto e inoltre è possibile estendere il software adottato per rispondere ad altri requisiti.

Software Engineering Research & Practices

Infine, il software open source può rivelarsi un'arma vincente nei progetti a lungo termine, poiché l'utente può costruire parte dei suoi servizi su un software che certamente non sparirà o cesserà di essere manutenuto se i creatori originali smetteranno di lavorare su di esso, cosa non possibile con software commerciali.

Tra i tanti, rientrano in questa categoria software quali ActiveMQ definito come “la scelta più popolare e il sistema di messaggistica più potente”, Artemis e Kafka della Apache Software Foundation, e RabbitMQ di Pivotal usato “da tutti, dalle piccole start-up della Silicon Valley, ai più grandi nomi dell’internet”.

Ma come scegliere un massage broker? Non c'è una semplice risposta, ogni sistema da sviluppare ha il proprio set di requisiti da soddisfare ed ogni gruppo di sviluppo le proprie competenze ed esperienze. Di seguito vengono analizzati i precedenti quattro prodotti presi in considerazione, in modo da fare una scelta secondo i requisiti che devono essere soddisfatti.

1.6.1 Broker presi in considerazione

1.6.1.1 RabbitMQ

Figura 4 RabbitMQ logo

RabbitMQ è un broker di messaggi "tradizionale" che implementa una discreta varietà di protocolli di messaggistica. È stato uno dei primi message-broker open source a raggiungere un elevato livello di funzionalità, librerie client e strumenti di sviluppo per i linguaggi più popolari, il tutto accompagnato da una documentazione di qualità. Originariamente sviluppato per implementare AMQP (Advanced Message Queuing Protocol, uno standard aperto che definisce un protocollo a livello applicativo per i Message-Oriented Middleware), è stato più tardi esteso attraverso la sua architettura a plug-in per supportare il protocollo MQTT (Message Queuing Telemetry Transport), STOMP (Streaming Text Oriented Messaging Protocol) e altri protocolli.

RabbitMQ, progettato come message broker per uso generale, utilizza diverse varianti dei modelli di comunicazione point to point, request/reply and pub-sub. Adotta una politica smart broker/dumb consumer incentrata

Software Engineering Research & Practices

sulla costante consegna di messaggi ai consumer, che processano i messaggi allo stesso passo in cui il broker glieli consegna. È un prodotto ormai maturo, dalle ottime performance se configurato correttamente, ben supportato (ci sono libreria client scritte in Java, .Net, node.js, Ruby, PHP, e molti altri linguaggi ancora) e ha a disposizione dozzine di plugin che ne estendono le funzionalità a molti casi d'uso e scenari di integrazione. La comunicazione in RabbitMQ può essere di tipo sincrono o asincrono, in base alle necessità. Può essere configurato per formare cluster multinodo in cluster federation, e non ha dipendenze su servizi esterni (servizi Amazon, Google o Microsoft).

1.6.1.2 Apache Kafka

Figura 5 Apache Kafka logo

Apache Kafka è stato sviluppato originariamente da LinkedIn in Scala come un modo per collegare i diversi sistemi interni all'azienda. A quel tempo, infatti, LinkedIn si stava muovendo verso un'architettura più distribuita e aveva bisogno di re-immaginare le funzionalità come l'integrazione dei dati e l'elaborazione del flusso in tempo reale, rompendo i precedenti approcci "monolitici" a questi problemi. Kafka oggi è adottato all'interno dell'ecosistema Apache ed è particolarmente utile nelle architetture event-driven.

Kafka è progettato per volumi elevati di messaggi pub-sub e stream, destinato ad essere durevole, veloce e scalabile. Essenzialmente fornisce un archivio per i messaggi durevole eseguito in un cluster server, simile ad un registro, che memorizza i flussi di messaggi in forma chiave-valore in categorie chiamate topic. I dati possono essere suddivisi in partizioni differenti all'interno di topic differenti. All'interno di una partizione, i messaggi sono rigorosamente ordinati in base al loro offset (la posizione del messaggio all'interno di una partizione) e indicizzati e memorizzati insieme ad un timestamp. Per l'elaborazione degli stream, Kafka offre le Stream API che permettono di scrivere applicazioni Java che consumano i dati presi da Kafka, reinserendoli nel broker ma in forma modificata. Contrariamente a RabbitMQ, Kafka adotta una politica dumb-broker/smart-consumer per

Software Engineering Research & Practices

leggere dal suo buffer. Il broker, infatti, non tiene traccia dei messaggi letti da ciascun utente o conserva solo i messaggi non letti, piuttosto Kafka conserva tutti i messaggi per un determinato periodo di tempo, lasciando la responsabilità ad ogni consumer di tenere traccia della loro posizione in ogni registro.

Kafka gira su uno o più di un server chiamati broker, e le partizioni di tutti i topic sono distribuite tra i nodi del cluster (replicando ogni partizione su più broker in base al fattore di ridondanza scelto in partenza). Questa architettura consente a Kafka di generare un enorme flusso di messaggi, conservando un alto livello di tolleranza agli errori, e permettendo di sostituire i sistemi di messaggistica più tradizionali come Java Message Service (JMS), Advanced Message Queuing Protocol (AMQP), ecc.

1.6.1.3 Apache ActiveMQ e Apache ActiveMQ Artemis

Figura 6 Apache ActiveMQ logo

Apache ActiveMQ, come RabbitMQ, rientra tra i broker di messaggi "tradizionali". Progettato dalla Apache Software Foundation nel 2004 è ancora attivamente supportato nel 2019, con l'ultima release stabile rilasciata a Novembre 2018, ed è definito come "la scelta più popolare e più potente tra i sistemi di messaggistica open source". Scritto in Java, con supporto JMS (un insieme di API che consentono lo scambio di messaggi tra applicazioni Java distribuite sulla rete) ad oggi supporta una vasta varietà di client multilingue in PHP, Pearl, Python, C, C++, C#, Ruby, ecc. Inoltre, supporta nativamente, senza necessità di integrazione attraverso plug-in come RabbitMQ dei protocolli di comunicazione OperWire (per client ad alte prestazioni scritti in Java, C, C++ e C#), MQTT (Message Queuing Telemetry Transport) supporto che permette connessioni nell'ambiente IoT, STOMP (Streaming Text Oriented Messaging Protocol) e AMQP (Advanced Message Queuing Protocol). Sebbene più scarno a livello di documentazione rispetto a RabbitMQ, gode comunque di una grande fama che l'accompagna sin dagli esordi ed è seguito da una vasta community. Ha ottime performance, discretamente supportato, aggiornato costantemente e facile da configurare.

Software Engineering Research & Practices

HornetQ è un sistema di messaggistica asincrona open-source, sviluppato da Jboss a partire dal 2007 (anno di pubblicazione dei lavori su RabbitMQ), progettato per essere multiprotocollo, integrabile, dalle performance elevate, facilmente clusterizzabile e scalabile. A fine 2014 il codice del Middleware è stato donato alla Apache Software Foundation e adesso è gestito come progetto parallelo di ActiveMQ, con il nome di ActiveMQ Artemis. A sei mesi di distanza, è stato rilasciato Apache Artemis 1.0.0.

La speranza è che l'unione di due grandi community come quella di HornetQ e ActiveMQ segnerà il percorso di una nuova generazione di message broker dalle funzionalità più avanzate, con prestazioni migliori e dalla maggiore stabilità. La community di sviluppatori spera di raggiungere questi obiettivi utilizzando il core di HornetQ con le sue elevate prestazioni, in combinazione con la vasta offerta di funzionalità di ActiveMQ. Ad oggi supporta un grande numero di protocolli di comunicazione come AMQP, STOMP, il protocollo di comunicazione nativo di ActiveMQ, OpenWire (in alpha, con supporto per i client ActiveMQ basati su JMS), e JMS2. Inoltre, il team ha iniziato lo sviluppo sul supporto del protocollo MQTT.

1.6.2 Enterprise Integration Patterns (EIP)

Apache Camel è un potente framework open source scritto in Java che fornisce un motore di routing e mediazione basato su regole prestabilite, usato per facilitare l'integrazione di sistemi distribuiti. È un'implementazione degli Enterprise Integration Patterns (EIP), modelli raccolti in un libro di design pattern utilizzati per l'integrazione dei sistemi nuovi o esistenti in un ambiente aziendale. Il libro, il cui titolo completo è Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions è stato scritto da Gregor Hohpe e Bobby Woolf e pubblicato nel 2003. EIP fornisce 65 modelli di progettazione e include un linguaggio basato su icone, simile all'UML, chiamato GregorGrams.

Un modello di progettazione è una soluzione generale ad un problema di progettazione che ricorre ripetutamente in diversi progetti. Un modello descrive un problema e la sua soluzione e discute qualsiasi altro fattore importante che segue al caso. I messaggi, in questo caso, facilitano la comunicazione dei programmi tra i diversi ambienti di programmazione poiché l'unica cosa che ogni ambiente deve capire è il comune formato e protocollo del messaggio.

Software Engineering Research & Practices

Camel rende questa integrazione più accessibile agli sviluppatori, implementando tutti i pattern più diffusi, collegando un gran numero di componenti (https://github.com/apache/camel/tree/master/components) e API, fornendo diversi DSL (domani-specific language) per integrare ogni componente.

Tra le varie componenti a disposizione troviamo i broker selezionati in precedenza quali RabbitMQ e Apache Kafka, ma anche altri servizi di messaggistica più o meno usati come ZeroMQ, Kestrel, IronMQ ma anche Amazon Simple Queue Service. Le componenti a disposizione ad oggi sono più di 291.

1.6.3 Scelta del broker: conclusioni preliminari

La seguente tabella riassume e compara le caratteristiche dei broker presi in considerazione: Apache ActiveMQ (nella sua forma base, Artemis al momento è ancora un prodotto incompleto), Apache Kafka e RabbitMQ.

	Apache ActiveMQ	*Apache Kafka*	*RabbitMQ*
Anno di creazione	2004	2011	2007
Licenza	Apache (Open Source)	Apache (Open Source)	Mozilla Public License (Open Source)
Linguaggio	Java	Scala	Erlang
Supporto protocolli	AMQP, AUTO, MQTT, OpenWire, REST, RSS and Atom, Stomp, WSIF, WS Notification, XMPP[39]	Protocollo proprietario	STOMP via plugin, MQTT via plugin, AMQP[41]
Client ufficiali	Java	Java	Java, .NET/C#, Erlang

Software Engineering Research & Practices

Client non ufficiali	~16 linguaggi (Python, .NET, C++, ecc.)	~10 linguaggi (Python, Node.js, ecc.)	~13 linguaggi (Python, PHP, Node.js, ecc.)
Documentazione	Datata, nuovo sito web in costruzione per Artemis. Continuamente supportata con esempi ufficiali	Completa	Completa
Supporto community (forum, stackoverflow, testi, ecc.)	Ottimo, sopperisce alle mancanze della documentazione ufficiale	Ottimo	Eccellente
Spazio di archiviazione	Temporaneamente su disco	Su disco	RAM
Persistenza dei contenuti	Opzionale, completa di default	Obbligatoria e completa	Opzionale, temporanea di default
Cancellazione messaggi	Dopo la consegna del messaggio	Dopo aver raggiunto la dimensione massima delle code, o dopo un	Immediatamente dopo la consegna del messaggio

Software Engineering Research & Practices

		limite di tempo impostato	
Compressione dei dati	Non presente	Presente	Non presente
Code predefinite nei broker	Presenti	Presenti	Presenti
Definizione code da remoto	Possibile	Non possibile	Possibile
Routing avanzato sui messaggi	Possibile sfruttando l'integrazione con il framework Apache Camel	Non presente	Possibile
Supporto agli Enterprise Integration Pattern	Supporto completo sfruttando l'integrazione con il Framework Apache Camel	Possibile utilizzando Apache Camel e utilizzando Apache Kafka come componente esterno	Possibile utilizzando Apache Camel e utilizzando RabbitMQ come componente esterno
Autosufficienza	Pienamente autosufficiente	No, necessita del supporto di Apache ZooKeeper	Pienamente autosufficiente

Software Engineering Research & Practices

Interfaccia di monitoraggio	JMX (howt.io), Web	JMX e CLI-based	Web e CLI-based

Apache Kafka è un Message-Oriented Middleware atipico con un insieme unico di funzionalità; il punto principale di differenziazione rispetto agli altri due broker è il modo in cui gestisce i messaggi in arrivo, dove vengono salvati e per quanto tempo rimangono in memoria anche dopo essere stati letti. Sono queste le caratteristiche che permettono al Broker di raggiungere velocità non raggiungibili da altri due middleware, e che lo rendono unico nel suo genere.

Tuttavia, questa velocità ha un prezzo da pagare:

- Il numero di protocolli standard supportati è pressoché nullo, in quanto si affida a un metodo di comunicazione proprietario.
- Il broker non è autonomo, ed ha bisogno di Apache Zookeper per gestire gli accessi alle code, il ciclo di vita dei messaggi, il loro salvataggio in memoria, ecc.

Ed è proprio per queste ragioni che, per quanto efficiente e unico nel suo genere, non è adeguato alla soluzione che è stata architettata. Adottare protocolli di comunicazione proprietari significherebbe modificare la natura della comunicazione nei dispositivi che si vanno ad integrare nel sistema. Per quanto in alcuni casi è possibile modificare questi dispositivi affinché comunichino tutti con il protocollo di comunicazione scelto e si integrino in altri sistemi, la gran parte delle volte questi devono essere presi "as is".

Apache ActiveMQ (e Artemis) e RabbitMQ, sono due Message-Oriented Middleware più tradizionali. Entrambi implementano una discreta varietà di protocolli di comunicazione ed entrambi offrono un elevato numero di funzionalità, librerie, documentazioni e strumenti di sviluppo. RabbitMQ è scritto in Erlang, ideato per essere leggero e veloce. La documentazione ufficiale è di gran lunga migliore di quella di ActiveMQ (sebbene con Artemis si mira a raggiungere il livello di RabbitMQ) e gode di un eccellente supporto della community. Tuttavia, per ottenere tutti i vantaggi che il broker offre, al setup iniziale deve seguire una lunga fase di configurazione dei plugin. La comunicazione tramite protocollo è gestita tramite plugin, così come la gestione dei messaggi tramite le tecniche di routing.

Software Engineering Research & Practices

Al contrario, Apache ActiveMQ è distribuito con il supporto completo agli Enterprise Integration Patterns (gli stessi descritti nel libro di Gregor Hohpe e Bobby Woolf) attraverso la libreria di Apache Camel. È possibile aggiungere facilmente qualsiasi pattern di integrazione in ActiveMQ (sia sul client JMS, sia nel processo del broker) per implementare smart routing, trasformazione dei messaggi e altri pattern di integrazione (la libreria di Camel è integrabile direttamente nell'applicazione ad esempio via Spring). Inoltre, rispetto a RabbitMQ gestisce nativamente un numero superiore di protocolli di comunicazione. Nel contesto della soluzione i-mole in cui prospetticamente deve poter essere possibile integrare il maggior numero di dispositivi IoT e di informazioni da questi generabili la configurabilità dei pattern di manipolazione dei dati e la numerosità dei protocolli di comunicazione gestibili ha fatto sì che la tecnologia scelta sia **Apache ActiveMQ**.

Software Engineering Research & Practices

Nell'ambito dello sviluppo software con il termine "stack" ci si riferisce a qualsiasi combinazione di linguaggi e tecnologie di programmazione o una combinazione di prodotti software utilizzati per realizzare siti e applicazioni.

La struttura di base di uno stack prevede un sistema operativo, un server web, un database e un interprete di script.

Le richieste crescenti delle sempre più complesse applicazioni web in termini di facilità d'uso, stabilità, raggiungibilità e di scalabilità ha comportato la nascita di nuovi stack tecnologici.

In particolare, tra tutti gli stack applicativi si sono affermati tre che sono di seguito riportati.

Un classico: LAMP

LAMP è, come già menzionato, la piattaforma software più conosciuta, che ha rappresentato per molto tempo il pacchetto ottimale per i progetti web. Presenta:

- Linux come sistema operativo,
- Apache come server web,
- un database MySQL
- e un interprete di script PHP.

Tutti i componenti usati sono software open source, con licenza libera, modificabili facilmente secondo le proprie esigenze e anche disponibili gratuitamente. Una piattaforma LAMP può comprendere sin dall'inizio anche soluzioni proprietarie simili ed è stata ad esempio utilizzata per sviluppare MediaWiki, il software di Wikipedia. Il pacchetto LAMP è particolarmente adatto per la realizzazione di siti dinamici composti da molte pagine. Gli utenti traggono soprattutto vantaggio dalla grande community, dai tutorial dettagliati e dalla possibilità di usare il linguaggio PHP e tool potenti come phpMyAdmin.

Con lo sviluppo della tecnologia e di nuovi software con codice libero, sono state create diverse varianti LAMP, in cui vengono sostituiti alcuni componenti comuni, tra cui quelli più conosciuti sono:

- WAMP (con sistema operativo Windows)
- MAMP (con sistema operativo Mac OS X)

Software Engineering Research & Practices

- XAMPP (con un sistema operativo qualsiasi, con Perl e PHP come interpreti di script; in più un server FTP multipiattaforma)
- LAPP (con PostgreSQL come database per realizzare progetti aziendali).

La piattaforma software di Microsoft: WISA

Al contrario della piattaforma LAMP, i cui singoli componenti non sono stati appositamente sviluppati per essere utilizzati insieme, la Microsoft ha programmato quelli della WISA proprio per questo scopo, garantendo così una migliore funzionalità del pacchetto. Questa piattaforma è composta:

- dal sistema operativo Windows Server,
- dal web server Internet Information Services (IIS),
- dal sistema di database relazionale SQL Server,
- e dalla libreria del linguaggio di programmazione ASP.NET.

Grazie alla libreria di ASP.NET si possono programmare siti con Visual C# o Visual Basic.NET, che possono essere eseguiti direttamente dal sistema grazie ad un compilatore. WISA non prevede perciò l'uso di un interprete di script e presenta per questo delle prestazioni migliori rispetto a piattaforme simili. La piattaforma software di Microsoft è una buona soluzione per progetti complessi, che si contraddistinguono per l'alto numero di visitatori e di accessi.

La struttura moderna per applicazioni web semplici: MEAN

Diversa è la piattaforma software MEAN: per via delle sue componenti non sono necessari diversi linguaggi di programmazione per il server (come PHP) e per il client (come JavaScript), visto che tutti gli elementi di questo pacchetto supportano JavaScript. Inoltre, il sistema operativo e il web server si collocano in secondo piano. I componenti di MEAN sono:

- il database NoSQL MongoDB,
- il framework per applicazioni web lato server Express.js,
- il framework front end lato client AngularJS
- e la piattaforma Node.js basata sull'ambiente di runtime di JavaScript.

Questo stack consente un processo di sviluppo notevolmente semplificato e accelerato, oltre che una buona scalabilità del progetto. Inoltre, è possibile realizzare in modo ottimale le applicazioni web semplici lato client o le app

Software Engineering Research & Practices

per i dispositivi mobili, grazie alla trasmissione dei dati in formato JSON. Anche per lo stack MEAN ci sono diverse varianti come MEEN (con Ember.js come framework front end) o MERN (con React.js come framework front end). Un esempio di applicazione dello stack MEAN è la piattaforma JavaScript Meteor.

Inquadrato il problema generale, passiamo nel prosieguo all'analisi di un punto dei punti da analizzare rispetto all'approccio MEAN ovvero il framework front end.

1.7 Framework

Con il termine framework si intende un oggetto che mette a disposizione degli sviluppatori la struttura di un programma che viene utilizzata come base nello sviluppo software. In mancanza di un framework uno sviluppatore dovrebbe scrivere, ogni volta, la struttura di base del programma, andando nei fatti a riscrivere soluzioni già pronte sotto forma di codice.

Nel caso in cui il framework viene utilizzato come struttura di base delle applicazioni web, si parla di web application framework. Negli ultimi anni, si è vissuta una crescita esponenziale delle applicazioni web e dei web application framework pensati per supportare la sempre maggiore necessità di supporto nella realizzazione delle stesse applicazioni web. Per questo motivo ad oggi esistono moltissimi framework o librerie (collezione di oggetti) pensati per supportare lo sviluppo web. Le librerie per la realizzazione delle interfacce utente a differenza dei framework che consentono di realizzare tutti i livelli MVC di un intero sistema, consentono di realizzare solo il livello di View.

Quanto riportato sino ad ora in aggiunta alla maggiore richiesta di incremento della qualità, di standardizzazione dei sistemi software, della riduzione dei costi di sviluppo, della flessibilità del gruppo di lavoro e così via hanno reso florido il mondo dei framework e delle librerie per lo sviluppo web. Di seguito si riportano alcuni dei web framework più utilizzati.

1.7.1 React

React è una "libreria JavaScript dichiarativa, efficiente e flessibile per la creazione di interfacce utente".

Software Engineering Research & Practices

Realizzato da Facebook (la prima versione è stata rilasciata a Maggio 2013), React semplifica la creazione di interfacce utente interattive. La libreria (che non si parli di framework) è progettata per la creazione di applicazioni incentrate sui componenti e tenendo presente la compatibilità con le versioni precedenti in modo che si possa essere certo della sua longevità.

Github mostra come Vue abbia oltre 6700 whatcher, più di 151000 stelle, quasi 30000 fork e oltre 1400 contributori attivi. Ad oggi Github mostra come il framework sia utilizzato il oltre 4.000.000 di progetti. React è decisamente la libreria JS più popolare del momento, ma non la più completa (essendo solo una libreria).

Il sondaggio "State of JavaScript" ha rivelato un aumento della ritenzione degli interessi dal 64% al 71% su un periodo di 12 mesi. Il rovescio della medaglia, il numero di intervistati che non vogliono utilizzare reagisce nuovamente è aumentato del 2%. Praticamente ogni sviluppatore JavaScript ha sentito parlare di React e la percentuale di sviluppatori che non sono interessati è diminuita dal 9% al 7% quest'anno.

1.7.2 Vue.js

Vue.js è un framework avanzato per la creazione di interfacce utente. Consiste in una libreria principale che si concentra solo sul livello di visualizzazione e in un ecosistema di librerie di supporto che permettono di affrontare la complessità nelle single page app di grandi dimensioni. Sviluppato da Evan You (la prima versione è stata rilasciata nel 2014) ha subito riscosso un grande successo nella comunità di sviluppatori che, secondo i dati GitHub, hanno utilizzato il framework in quasi 1,2 milioni di progetti.

Vue.js ha registrato una crescita massiccia negli ultimi 12 mesi e i dati del sondaggio "State of JavaScript" suggeriscono che il divario nell'utilizzo con i framework più utilizzati si sta riducendo: il 34% degli sviluppatori vorrebbe imparare Vue, anche se in calo rispetto al 47% del rapporto precedente. Inoltre, i tassi di fidelizzazione sono saliti alle stelle dal 28% a un enorme 40%.

Github mostra come Vue abbia oltre 6.200 whatcher, più di 166.000 stelle, oltre 25.200 fork ma soli 300 contributori attivi. Ad oggi Github mostra come il framework sia utilizzato il oltre 93.000 progetti.

Software Engineering Research & Practices

C'è ancora un divario di adozione rispetto ai framework più utilizzati e, forse, una mancanza di contenuti per chi è alle prime armi, ma il team di Vue e la comunità, giorno dopo giorno, affrontano e sistemano questi inconvenienti.

1.7.3 Angular

Angular è un framework open source per lo sviluppo di applicazioni web con licenza MIT, evoluzione di AngularJS. Sviluppato principalmente da Google, la sua prima release è avvenuta il 14 settembre 2016. Angular è stato completamente riscritto rispetto a AngularJS (altro framework sviluppato nel 2010 da Google) e le due versioni non sono compatibili.

Il framework consente agli sviluppatori di creare applicazioni che vivono su Web, dispositivi mobili e desktop. Angular combina modelli dichiarativi, dependency injection, end-to-end tooling e suggerisce le best practice per risolvere le sfide di sviluppo. Ha un'utile CLI, quindi i principianti possono iniziare facilmente. C'è anche un client GUI chiamato Console.

Github mostra come Angular abbia oltre 3.200 whatcher, più di 62.000 stelle, quasi 17.000 fork e oltre 1.200 contributori attivi. Al momento GitHub non è stato configurato per vedere il numero di progetti che utilizzano il framework.

Ad oggi Angular è sicuramente il framework più completo. Ha al suo interno tutto ciò che serve per la creazione di single-page-app senza scomodare strumenti terzi (che pur ci sono) come si è obbligati a fare con React o Vue. La documentazione è eccellente e la community attiva è enorme.

1.7.4 Preact

Preact è una libreria JavaScript che si pone come un'alternativa più veloce e leggera (da 3kB) di React JS che utilizza la stessa moderna API del browser e con una strutturazione a componenti e un virtual DOM. Più di 18.000 progetti utilizzano Preact e ha quasi 25.000 stelle su GitHub.

Preact ha registrato un buon livello di crescita dallo scorso anno, con un tasso di fidelizzazione in aumento al 9% dal 6% dello scorso anno. Anche la consapevolezza della sua esistenza è aumentata; il numero di persone che non hanno sentito parlare della libreria è sceso dal 28% al 24% negli ultimi 12 mesi.

Software Engineering Research & Practices

1.7.5 Ember

Ember è un framework web JavaScript open source basato sul pattern model-view-view-model. Consente agli sviluppatori di creare single-page-app scalabili incorporando nel framework tecniche e best practice dei framework più noti.

Il sondaggio ha rivelato un crescente interesse per Ember negli ultimi 12 mesi, con la percentuale di persone che non erano interessate a Ember che è scesa dal 67% al 64%.

1.7.6 Svelte

Svelte ha un nuovo approccio radicale per la creazione di interfacce utente. Mentre i framework tradizionali come React e Vue svolgono la maggior parte del loro lavoro nel browser, Svelte ha un ulteriore passaggio di compilazione che si verifica quando si crea la tua app. Invece di utilizzare tecniche come i virtual DOM, Svelte scrive codice che aggiorna componente per componente il DOM quando lo stato dell'app cambia.

Quest'anno, Svelte è diventata una delle prime sei librerie di frontend JS, con quasi il 45% degli sviluppatori che ha indicato un interesse per l'apprendimento. Gli autori del rapporto hanno anche nominato Svelte il vincitore del suo "Prediction Award", riconoscendo la biblioteca per il suo "debutto esplosivo" nel 2019.

1.7.7 Blazor

Blazor è un framework per realizzare web UI interattive. Permette di usare il linguaggio C# e tutti gli strumenti che già vengono utilizzati con ASP.NET Core per sviluppare la parte client di un'applicazione web. Grazie ad esso è possibile ridurre al massimo l'uso del JavaScript.

Grazie a Blazor possiamo scrivere codice C# per reagire all'interazione dell'utente, come ad esempio il click su un bottone, e scatenare l'aggiornamento della web UI.

Software Engineering Research & Practices

Blazor, quindi, propone un modello di sviluppo dell'interfaccia HTML supportato da codice C#. Per questo motivo può sembrare reminiscente di ASP.NET WebForms ma, all'atto pratico, ha più somiglianze con framework client-side come VueJS.

Blazor offre una via semplice allo sviluppo web anche a chi non ha particolare esperienza nello sviluppo front-end o che magari si è cimentato solo con lo sviluppo di applicazioni desktop con WinForms e WPF.

1.8 Classifiche e benchmark

Come abbiamo potuto vedere dal breve riepilogo precedente il mondo dei framework per il frontend è: vasto, vario e dinamico al punto che al punto che classifiche di popolarità o gradimento, classificazioni di tipologie e benchmark di prestazioni e risultato ne esistono a bizzeffe. La presenza delle numerose classifiche, che posizionano framework differenti nelle posizioni di testa, conseguenza diretta della difficoltà di recuperare informazioni precise rende arduo il compito di decidere quale framework utilizzare per produrre lo strato di comunicazione con l'utente finale. Non avendo delle precise valutazioni a cui fare riferimento ma solo dati parziali e soggettivi a valle dello studio degli stessi rimango molti punti aperti, come: qual è la vita media di un framework, dove sta andando la community degli sviluppatori, qual è il framework i cui punti di forza rispecchiano le esigenze dell'azienda che lo deve adottare?

Per arrivare ad una decisione, che tenga presente l'organigramma aziendale, delle conoscenze del team di sviluppo, delle strategie di evoluzione della soluzione, oltre che del mercato di riferimento analizzeremo alcuni dei dati presenti in letteratura che possono essere un parametro da utilizzarsi nella scelta.

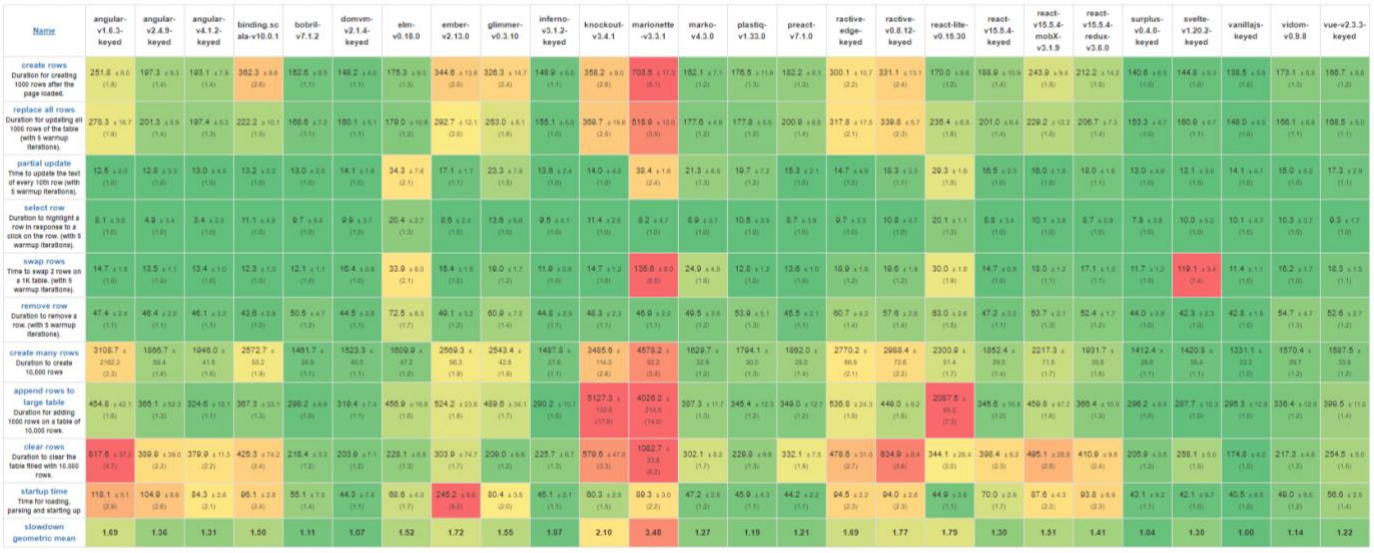
Per completezza, per raggiungere gli scopo della piattaforma i-mole nel seguito metteremo sullo stesso piano librerie e framework poiché utilizzeremo di entrambi la sola parte di presentazione.

1.9 Prestazioni dei framework di frontend

Partendo dalle classifiche dei framework più utilizzati e studiati, si è provveduto a creare una matrice di framework funzionale agli scopi di i-mole, che riporta il posizionamento dei framework rispetto ai competitor dal punto di vista delle prestazioni e della occupazione memoria.

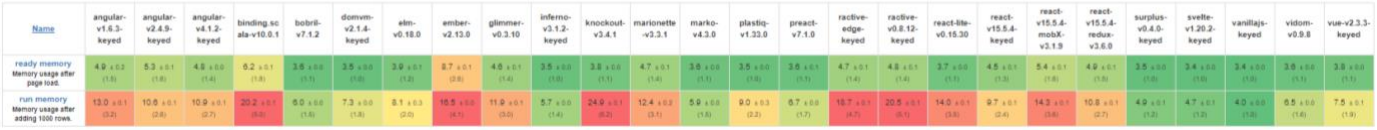
Software Engineering Research & Practices

Duration in milliseconds ± standard deviation (Slowdown = Duration / Fastest)



Name	angular-v1.6.3-keyed	angular-v2.4.9-keyed	angular-v4.1.2-keyed	binding.scala-v10.0.1	bobril-v7.1.2	domvm-v2.1.4-keyed	elm-v0.18.0	ember-v2.13.0	glimmer-v0.3.10	inferno-v3.1.2-keyed	knockout-v3.4.1	marionette-v3.3.1	marko-v4.3.0	plastiq-v1.33.0	preact-v7.1.0	ractive-edge-keyed	ractive-v0.8.12-keyed	react-lite-v0.15.30	react-v15.5.4-keyed	react-v15.5.4-mobX-v3.1.9	react-v15.5.4-redux-v3.6.0	surplus-v0.4.0-keyed	svelte-v1.20.2-keyed	vanillajs-keyed	vidom-v0.9.8	vue-v2.3.3-keyed
create rows Duration for creating 1000 rows after the page loaded.	251.8 ±8.0 (1.6)	197.3 ±9.3 (1.4)	193.1 ±7.9 (1.4)	382.3 ±9.6 (2.0)	152.6 ±8.5 (1.1)	148.2 ±4.0 (1.1)	175.3 ±8.3 (1.3)	344.6 ±13.6 (2.5)	326.3 ±14.7 (2.4)	148.9 ±8.6 (1.1)	358.2 ±9.0 (2.6)	703.5 ±17.3 (5.1)	162.1 ±7.1 (1.2)	176.5 ±11.8 (1.3)	182.2 ±8.3 (1.3)	300.1 ±10.7 (2.2)	331.1 ±13.1 (2.4)	170.0 ±8.6 (1.2)	188.9 ±10.9 (1.4)	243.9 ±9.4 (1.8)	212.2 ±14.2 (1.5)	140.6 ±6.5 (1.0)	144.8 ±8.3 (1.0)	138.5 ±8.6 (1.0)	173.1 ±8.5 (1.3)	166.7 ±8.6 (1.2)
replace all rows Duration for updating all 1000 rows of the table (with 5 warmup iterations).	278.3 ±16.7 (1.9)	201.3 ±9.9 (1.4)	197.4 ±8.3 (1.3)	222.2 ±10.1 (1.5)	166.6 ±7.2 (1.1)	160.1 ±8.1 (1.1)	179.0 ±10.6 (1.2)	292.7 ±12.1 (2.0)	263.0 ±8.1 (1.8)	155.1 ±8.0 (1.0)	369.7 ±19.6 (2.5)	518.9 ±12.0 (3.5)	177.6 ±8.9 (1.2)	177.8 ±8.5 (1.2)	200.9 ±8.8 (1.4)	317.8 ±17.5 (2.1)	339.8 ±8.7 (2.3)	235.4 ±8.8 (1.6)	201.0 ±8.4 (1.4)	229.2 ±12.2 (1.5)	206.7 ±7.3 (1.4)	153.3 ±4.7 (1.0)	160.9 ±4.7 (1.1)	146.0 ±4.8 (1.0)	166.1 ±6.8 (1.1)	168.8 ±5.0 (1.1)
partial update Time to update the text of every 10th row (with 5 warmup iterations).	12.5 ±2.0 (1.0)	12.8 ±3.3 (1.0)	13.0 ±4.5 (1.0)	13.2 ±2.2 (1.0)	13.0 ±2.0 (1.0)	14.1 ±1.8 (1.0)	34.3 ±7.6 (2.1)	17.1 ±1.7 (1.1)	23.3 ±7.8 (1.5)	13.6 ±2.4 (1.0)	14.0 ±4.5 (1.0)	38.4 ±1.6 (2.4)	21.3 ±6.5 (1.3)	19.7 ±7.2 (1.2)	15.3 ±2.1 (1.0)	14.7 ±4.8 (1.0)	18.3 ±3.3 (1.1)	29.3 ±1.8 (1.8)	16.5 ±2.3 (1.0)	16.0 ±1.8 (1.0)	18.0 ±1.6 (1.1)	13.0 ±4.6 (1.0)	13.1 ±3.0 (1.0)	14.1 ±4.7 (1.0)	16.0 ±5.3 (1.0)	17.3 ±2.9 (1.1)
select row Duration to highlight a row in response to a click on the row. (with 5 warmup iterations).	8.1 ±3.6 (1.0)	4.9 ±3.4 (1.0)	3.4 ±2.3 (1.0)	11.1 ±4.9 (1.0)	9.7 ±5.6 (1.0)	9.9 ±3.7 (1.0)	20.4 ±2.7 (1.3)	8.6 ±2.4 (1.0)	13.6 ±5.0 (1.0)	9.5 ±4.1 (1.0)	11.4 ±2.6 (1.0)	8.2 ±4.7 (1.0)	8.9 ±3.7 (1.0)	10.5 ±3.9 (1.0)	8.7 ±3.9 (1.0)	9.7 ±3.3 (1.0)	10.8 ±4.7 (1.0)	20.1 ±1.1 (1.3)	8.8 ±3.4 (1.0)	10.1 ±3.8 (1.0)	8.7 ±2.9 (1.0)	7.8 ±3.8 (1.0)	10.0 ±5.2 (1.0)	10.1 ±4.7 (1.0)	10.3 ±3.7 (1.0)	9.3 ±1.7 (1.0)
swap rows Time to swap 2 rows on a 1K table. (with 5 warmup iterations).	14.7 ±1.8 (1.0)	13.5 ±1.7 (1.0)	13.4 ±1.0 (1.0)	12.3 ±1.3 (1.0)	12.1 ±1.1 (1.0)	16.4 ±0.8 (1.0)	33.9 ±8.0 (2.1)	16.4 ±1.5 (1.0)	19.0 ±1.7 (1.2)	11.0 ±0.8 (1.0)	14.7 ±1.2 (1.0)	135.6 ±8.0 (8.5)	24.9 ±4.8 (1.6)	12.8 ±1.2 (1.0)	13.6 ±1.0 (1.0)	18.9 ±1.8 (1.2)	19.5 ±1.9 (1.2)	30.0 ±1.8 (1.9)	14.7 ±0.8 (1.0)	18.0 ±1.2 (1.1)	17.1 ±1.3 (1.1)	11.7 ±1.2 (1.0)	119.1 ±3.4 (7.4)	11.4 ±1.1 (1.0)	16.2 ±1.7 (1.0)	10.3 ±1.5 (1.1)
remove row Duration to remove a row. (with 5 warmup iterations).	47.4 ±2.4 (1.1)	46.4 ±2.0 (1.1)	46.1 ±3.2 (1.1)	43.6 ±2.9 (1.0)	50.5 ±4.7 (1.2)	44.5 ±2.8 (1.1)	72.5 ±8.3 (1.7)	49.1 ±3.2 (1.2)	60.9 ±7.2 (1.4)	44.8 ±2.5 (1.1)	48.3 ±2.3 (1.1)	46.9 ±2.2 (1.1)	49.5 ±3.5 (1.2)	53.9 ±5.1 (1.3)	45.5 ±2.1 (1.1)	60.7 ±4.2 (1.4)	57.6 ±2.8 (1.4)	63.0 ±2.6 (1.5)	47.2 ±3.2 (1.1)	53.7 ±2.1 (1.3)	52.4 ±1.7 (1.2)	44.0 ±2.6 (1.0)	42.3 ±2.3 (1.0)	42.8 ±1.5 (1.0)	54.7 ±4.7 (1.3)	52.6 ±2.7 (1.2)
create many rows Duration to create 10,000 rows	3108.7 ± 2162.2 (2.3)	1866.7 ± 55.4 (1.4)	1946.0 ± 41.8 (1.5)	2572.7 ± 55.2 (1.9)	1461.7 ± 24.8 (1.1)	1523.3 ± 40.0 (1.1)	1609.9 ± 47.2 (1.2)	2569.3 ± 56.3 (1.9)	2543.4 ± 42.8 (1.9)	1487.8 ± 27.8 (1.1)	3485.6 ± 114.8 (2.6)	4578.2 ± 62.2 (3.4)	1629.7 ± 32.8 (1.2)	1794.1 ± 30.0 (1.3)	1852.0 ± 29.0 (1.4)	2770.2 ± 66.6 (2.1)	2968.4 ± 72.6 (2.2)	2300.9 ± 51.4 (1.7)	1852.4 ± 29.0 (1.4)	2217.3 ± 71.6 (1.7)	1931.7 ± 35.8 (1.5)	1412.4 ± 28.8 (1.1)	1420.8 ± 35.4 (1.1)	1331.1 ± 22.2 (1.0)	1570.4 ± 29.7 (1.2)	1587.5 ± 33.9 (1.2)
append rows to large table Duration for adding 1000 rows on a table of 10,000 rows.	454.8 ±42.1 (1.6)	365.1 ±52.3 (1.3)	324.6 ±10.1 (1.1)	367.3 ±33.1 (1.3)	296.2 ±8.8 (1.0)	319.4 ±7.4 (1.1)	456.9 ±16.6 (1.6)	524.2 ±23.6 (1.8)	489.6 ±34.1 (1.7)	290.2 ±10.7 (1.0)	5127.3 ± 132.0 (17.8)	4026.2 ± 214.5 (14.0)	387.3 ±11.7 (1.3)	345.4 ±12.3 (1.2)	349.0 ±12.7 (1.2)	536.8 ±24.3 (1.9)	449.0 ±9.2 (1.6)	2067.5 ± 66.2 (7.3)	345.6 ±10.4 (1.2)	459.8 ±47.2 (1.6)	366.4 ±10.9 (1.3)	296.2 ±8.8 (1.0)	287.7 ±10.3 (1.0)	295.3 ±12.8 (1.0)	336.4 ±12.8 (1.2)	399.5 ±11.0 (1.4)
clear rows Duration to clear the table filled with 10.000 rows.	617.6 ±37.2 (4.7)	399.9 ±39.0 (2.2)	379.9 ±11.3 (2.2)	425.3 ±74.2 (2.4)	216.4 ±3.3 (1.2)	203.9 ±7.1 (1.2)	228.1 ±6.9 (1.3)	303.9 ±74.7 (1.7)	209.0 ±6.6 (1.2)	225.7 ±6.7 (1.3)	579.6 ±47.5 (3.3)	1082.7 ± 33.6 (6.2)	302.1 ±6.2 (1.7)	229.8 ±9.6 (1.3)	332.1 ±7.5 (1.9)	478.6 ±31.0 (2.7)	634.9 ±8.4 (3.6)	344.1 ±28.4 (2.0)	398.4 ±8.2 (2.3)	495.1 ±28.6 (2.8)	410.9 ±9.6 (2.4)	205.9 ±3.8 (1.2)	258.1 ±5.0 (1.5)	174.8 ±4.2 (1.0)	217.3 ±4.6 (1.3)	254.5 ±5.0 (1.5)
startup time Time for loading, parsing and starting up	118.1 ±5.1 (2.9)	104.9 ±8.6 (2.6)	84.3 ±2.6 (2.1)	96.1 ±2.8 (2.4)	55.1 ±7.0 (1.4)	44.3 ±7.4 (1.1)	68.6 ±4.0 (1.7)	245.2 ±8.6 (6.0)	80.4 ±3.5 (2.0)	45.1 ±2.1 (1.1)	60.3 ±2.6 (1.5)	89.3 ±3.0 (2.2)	47.2 ±2.5 (1.2)	45.9 ±4.3 (1.1)	44.2 ±2.2 (1.1)	94.5 ±2.2 (2.3)	94.0 ±2.6 (2.3)	44.9 ±2.6 (1.1)	70.0 ±2.9 (1.7)	67.6 ±4.3 (2.2)	93.8 ±6.9 (2.3)	43.1 ±3.2 (1.1)	42.1 ±3.7 (1.0)	40.5 ±3.5 (1.0)	49.0 ±5.0 (1.2)	56.6 ±2.5 (1.4)
slowdown geometric mean	1.69	1.36	1.31	1.50	1.11	1.07	1.52	1.72	1.55	1.07	2.10	3.48	1.27	1.19	1.21	1.69	1.77	1.79	1.30	1.51	1.41	1.04	1.30	1.00	1.14	1.22

Memory allocation in MBs ± standard deviation

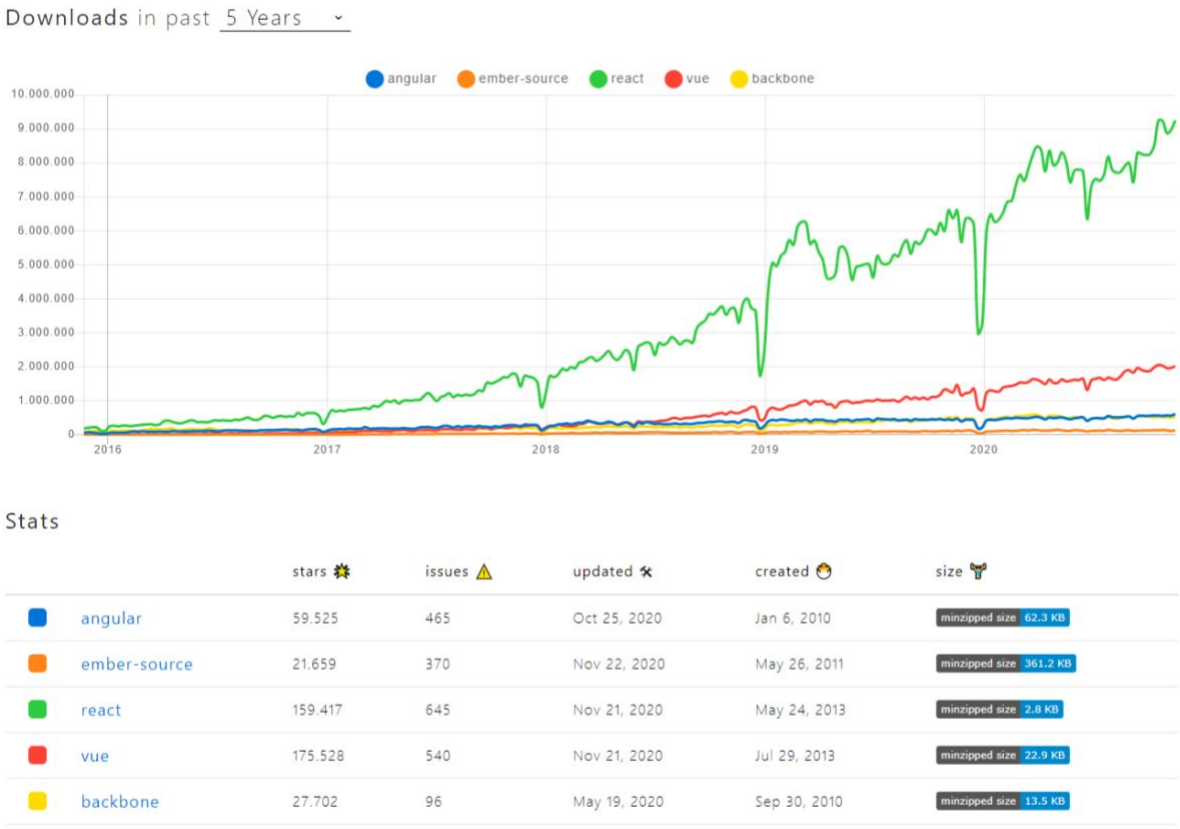


Name	angular-v1.6.3-keyed	angular-v2.4.9-keyed	angular-v4.1.2-keyed	binding.scala-v10.0.1	bobril-v7.1.2	domvm-v2.1.4-keyed	elm-v0.18.0	ember-v2.13.0	glimmer-v0.3.10	inferno-v3.1.2-keyed	knockout-v3.4.1	marionette-v3.3.1	marko-v4.3.0	plastiq-v1.33.0	preact-v7.1.0	ractive-edge-keyed	ractive-v0.8.12-keyed	react-lite-v0.15.30	react-v15.5.4-keyed	react-v15.5.4-mobX-v3.1.9	react-v15.5.4-redux-v3.6.0	surplus-v0.4.0-keyed	svelte-v1.20.2-keyed	vanillajs-keyed	vidom-v0.9.8	vue-v2.3.3-keyed
ready memory Memory usage after page load.	4.9 ±0.3 (1.5)	5.3 ±0.1 (1.6)	4.8 ±0.0 (1.4)	6.2 ±0.1 (1.8)	3.6 ±0.0 (1.1)	3.5 ±0.0 (1.0)	3.9 ±0.1 (1.2)	8.7 ±0.1 (2.6)	4.6 ±0.1 (1.4)	3.5 ±0.0 (1.0)	3.8 ±0.0 (1.1)	4.7 ±0.1 (1.4)	3.6 ±0.0 (1.1)	3.5 ±0.0 (1.0)	3.6 ±0.1 (1.1)	4.7 ±0.1 (1.4)	4.8 ±0.1 (1.4)	3.7 ±0.0 (1.1)	4.5 ±0.1 (1.3)	5.4 ±0.1 (1.6)	4.9 ±0.1 (1.5)	3.5 ±0.0 (1.0)	3.4 ±0.0 (1.0)	3.4 ±0.0 (1.0)	3.6 ±0.0 (1.1)	3.8 ±0.0 (1.1)
run memory Memory usage after adding 1000 rows.	13.0 ±0.1 (3.2)	10.6 ±0.1 (2.6)	10.9 ±0.1 (2.7)	20.2 ±0.1 (4.9)	6.0 ±0.0 (1.5)	7.3 ±0.0 (1.8)	8.1 ±0.3 (2.0)	16.5 ±0.0 (4.1)	11.9 ±0.1 (3.0)	5.7 ±0.0 (1.4)	24.9 ±0.1 (6.2)	12.4 ±0.2 (3.1)	5.9 ±0.0 (1.5)	9.0 ±0.3 (2.2)	6.7 ±0.0 (1.7)	18.7 ±0.1 (4.7)	20.5 ±0.1 (5.1)	14.0 ±0.1 (3.5)	9.7 ±0.1 (2.4)	14.3 ±0.1 (3.6)	10.8 ±0.1 (2.7)	4.9 ±0.1 (1.2)	4.7 ±0.1 (1.2)	4.0 ±0.0 (1.0)	6.5 ±0.0 (1.6)	7.5 ±0.1 (1.9)

1.9.1 Diffusione / progetti

Di seguito andremo ad analizzare alcune tendenze nel tempo dei framework più diffusi, allo scopo di utilizzare le informazioni estratte per fare delle ipotesi circa la lunghezza della vita del framework, la disponibilità di esempi applicativi per risolvere problemi di tipo generale, ecc.

Come primo dato riportiamo il trend dei download negli ultimi 5 anni dei framework più scaricati ed utilizzati per creare applicazioni.

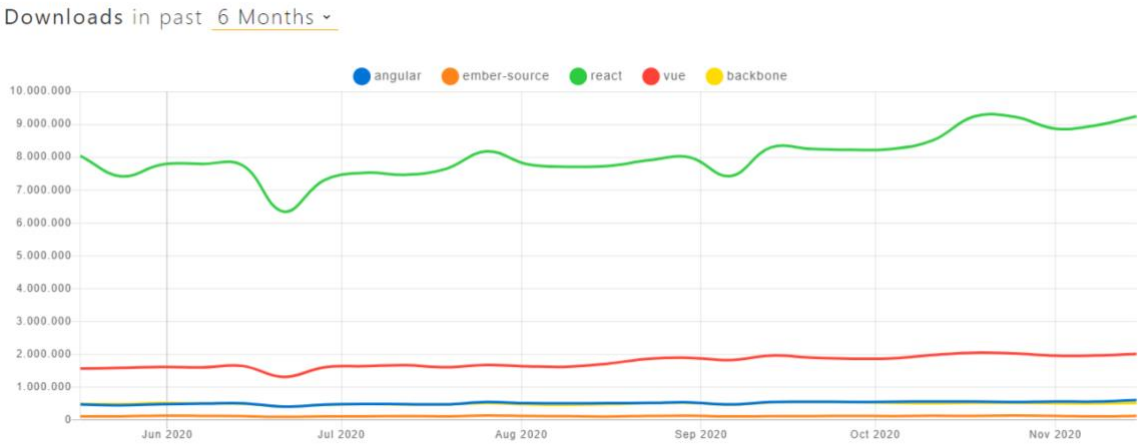


Stats

	stars	issues	updated	created	size
angular	59.525	465	Oct 25, 2020	Jan 6, 2010	minzipped size 62.3 KB
ember-source	21.659	370	Nov 22, 2020	May 26, 2011	minzipped size 361.2 KB
react	159.417	645	Nov 21, 2020	May 24, 2013	minzipped size 2.8 KB
vue	175.528	540	Nov 21, 2020	Jul 29, 2013	minzipped size 22.9 KB
backbone	27.702	96	May 19, 2020	Sep 30, 2010	minzipped size 13.5 KB

Lo stesso dato analizzato negli ultimi 6 mesi risulta più chiaro da leggere ed interpretare.

Software Engineering Research & Practices



Stats

	stars	issues	updated	created	size
angular	59.525	465	Oct 25, 2020	Jan 6, 2010	minzipped size 62.3 KB
ember-source	21.659	370	Nov 22, 2020	May 26, 2011	minzipped size 361.2 KB
react	159.417	645	Nov 21, 2020	May 24, 2013	minzipped size 2.8 KB
vue	175.528	540	Nov 21, 2020	Jul 29, 2013	minzipped size 22.9 KB
backbone	27.702	96	May 19, 2020	Sep 30, 2010	minzipped size 13.5 KB

Analizzando per gli stessi framework l'interesse e l'attività degli sviluppatori sul portale di Knowledge Sharing denominato Stack Overflow, riusciamo a portare alla luce una importante dato non rilevato in precedenza rispetto al framework JQuery, ovvero che questo tra il 2010 e il 2016 ha dominato gli interessi degli sviluppatori mentre negli ultimi 5 anni sono i framework più scaricati che giustamente hanno sostituito JQuery.

Software Engineering Research & Practices

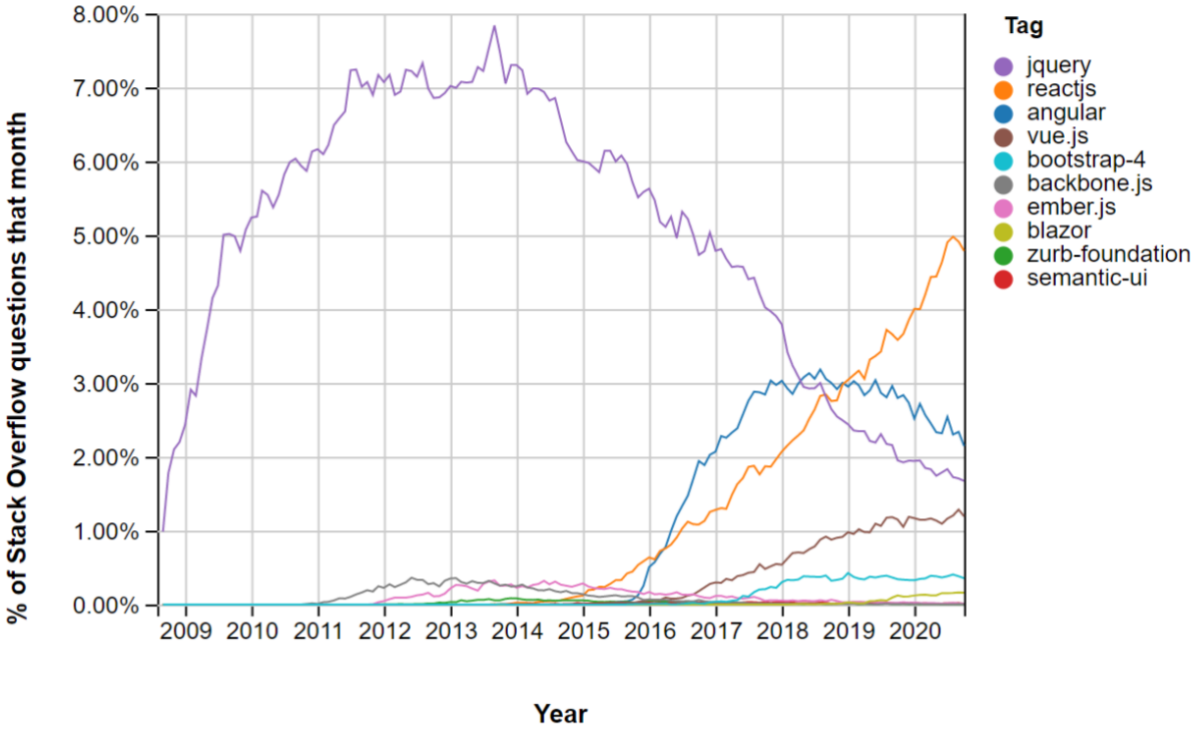
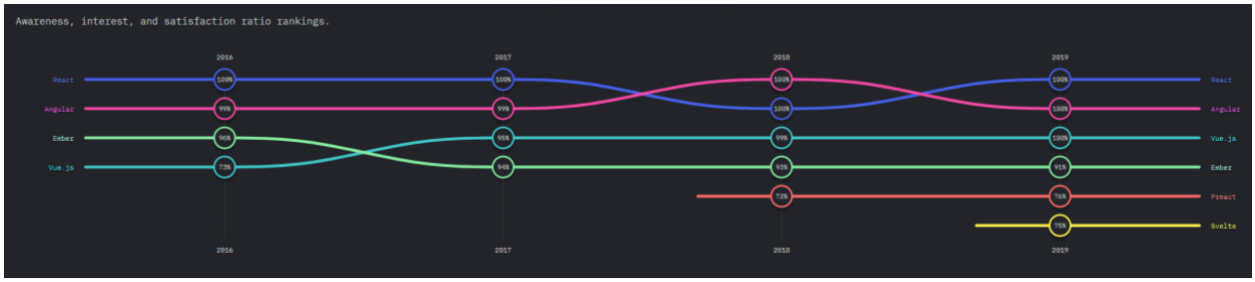


Figure 1 – Stack Overflow Trends

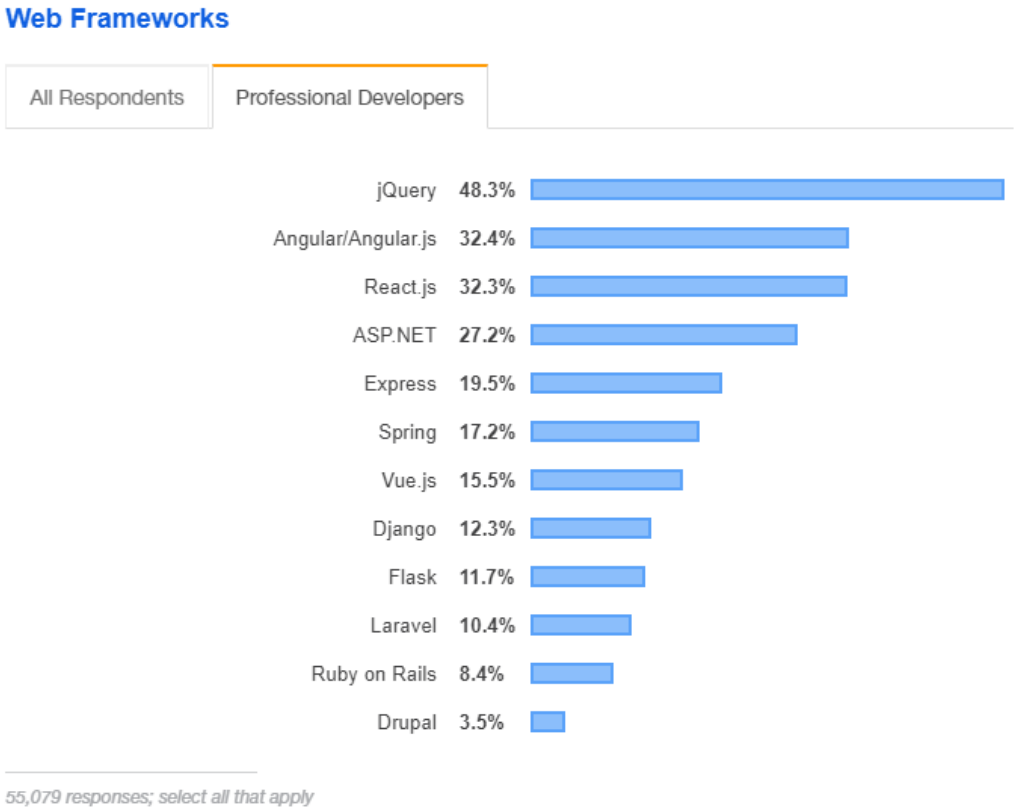
Per cercare di capire quindi verso quale framework gli sviluppatori più esperti e più attivi si stanno orientando, e dare maggiore evidenza al grafico con i dati di Stack Overflow, si riportano di seguito i dati di un altro questionario sottoposto agli sviluppatori da cui è possibile rilevare quale tra i framework uno sviluppatore sceglierebbe per sviluppare un nuovo progetto nella propria azienda.



Infine, concludiamo questa visione numerica riguardante la diffusione e l'utilizzo dei web framework con un interessante raffronto dei dati ripresi dai sondaggi 2019/2020 prodotti da Stackoverflow.

Nel primo grafico si riporta la risposta che di circa 58.500 sviluppatori professionali hanno fornito alla domanda: qual è il web framework che hai più utilizzato nei progetti?

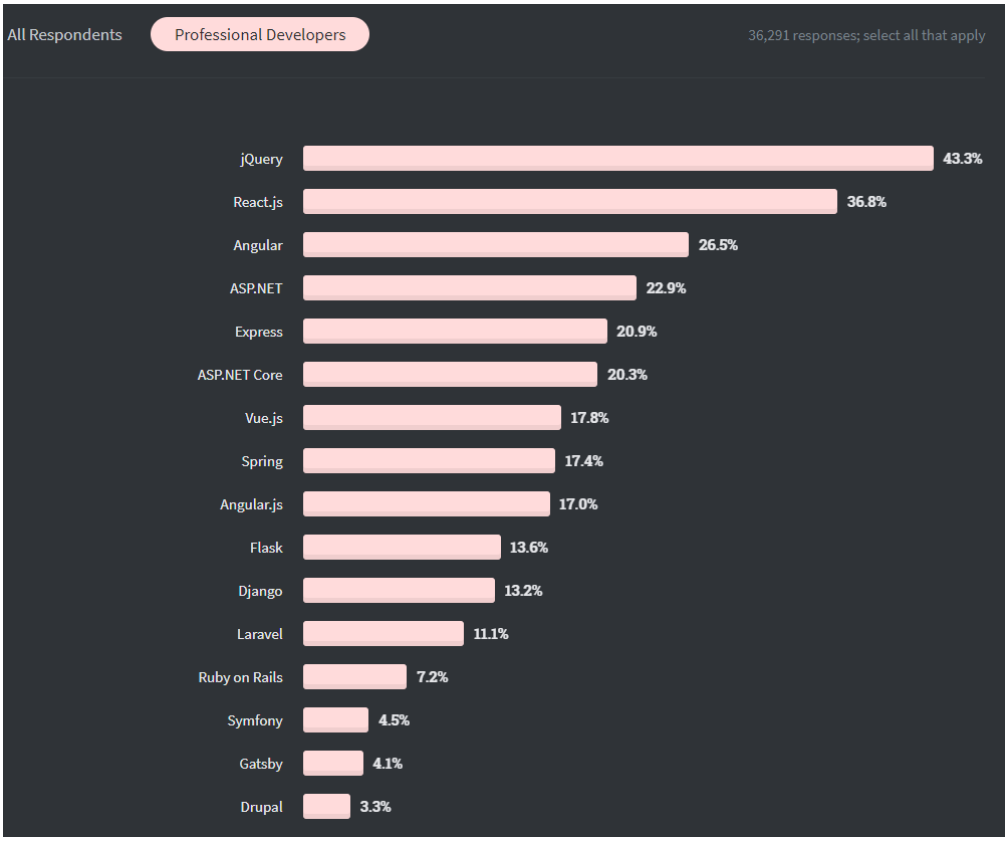
Software Engineering Research & Practices



Come si può facilmente evincere dai dati, ci sono Jquery, Angular e React anche se molti degli sviluppatori che hanno risposto al sondaggio hanno detto che nel 2018 sono passati da jquery ad angular o a react, ancora una volta confermando le tendenze visti in precedenza.

Mentre in questo secondo grafico si riporta la risposta alla stessa domanda: qual è il web framework che hai più utilizzato nei progetti? È importante evidenziare che il numero di sviluppatori che hanno risposto è sensibilmente più basso ed è di circa 36.000 individui.

Software Engineering Research & Practices

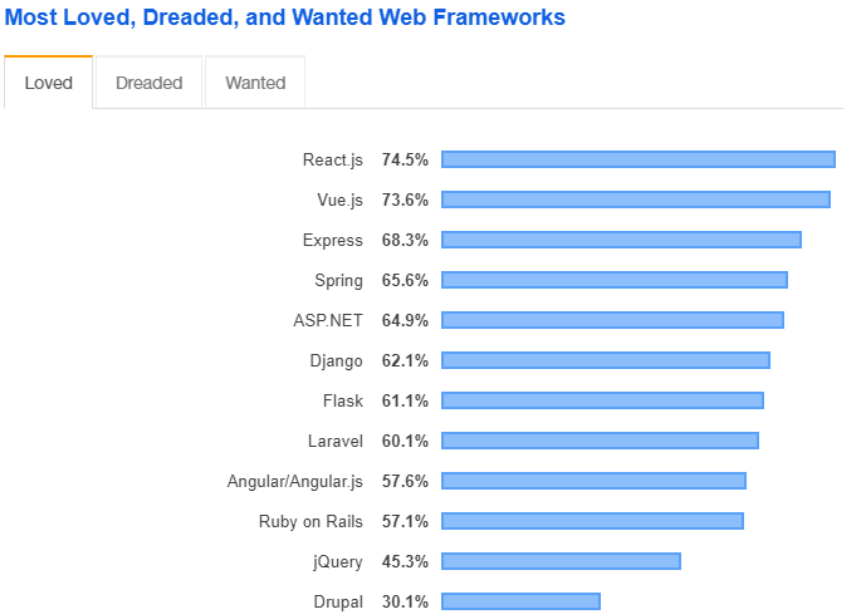


Questi numeri confermano quindi quanto rilevato dal sondaggio del 2019 e mettono in evidenza il fatto che Jquery continua a perdere mercato a “favore” dei nuovi framework tra cui React e Angular 2.

Inoltre, è particolarmente interessante, per capire in quale direzione il mondo degli sviluppatori si sta muovendo e per capire su quale web framework sarà ragionevolmente supportato nei prossimi anni, analizzare un altro dato estratto dallo stesso sondaggio in cui si chiede qual è il web framework più amato e che si cercherà di utilizzare nei prossimi progetti di sviluppo.

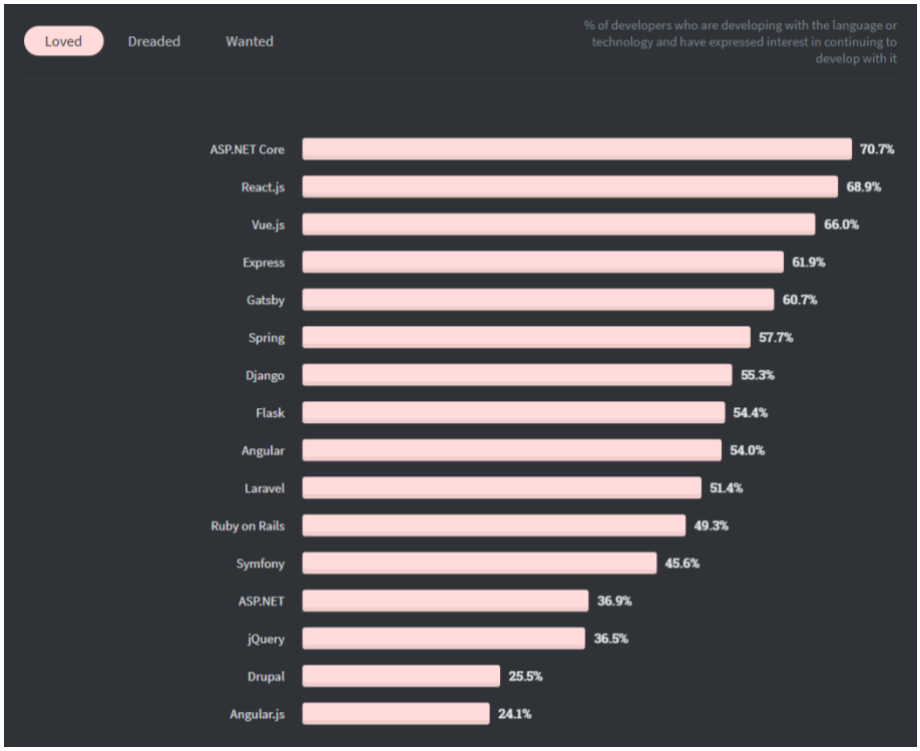
Nel grafico seguente sono riportate le risposte del 2019 in cui ai primi posti ci sono React e Vue che hanno scavalcato Jquery.

Software Engineering Research & Practices



Il sondaggio 2020 ha confermato il trend evidenziato nel sondaggio 2019 e messo in evidenza che i nuovi framework si sono ormai confermati come prossime tecnologie di sviluppo che gli sviluppatori vorranno utilizzare: React.js (68.9%), Vue.js (66.0%) e Angular 2 (54,0%). Un aspetto particolarmente interessante è dato dalla presenza al primo posto della classifica di ASP.NET Core (70.7%). Questo è legato al rilascio, nel corso del 2020, del framework .net core che si presenta come rivoluzionario rispetto al mercato e rispetto al modo di produrre applicazioni per il web consentendo ad esempio di superare due dei grandi problemi e/o vincoli delle precedenti versioni ovvero la possibilità di generare applicazioni multipiattaforma ed il fatto che il nuovo .Net core è rilasciato gratuitamente ed infine è pensato per creare applicazioni altamente distribuite nube, microservices e contenitori (finestra mobile).

Software Engineering Research & Practices



Per concludere, questo breve excursus numerico, possiamo considerare validi e attendibili i confronti che considerano Angular 2 e React.js i framework di riferimento per l'attuale produzione di applicazioni web e questo è ancora valido poiché la nuova versione di ASP.NET Core a cui gli sviluppatori vogliono migrare è stata rilasciata da pochi mesi e quindi sarà necessario attendere almeno un anno di tempo per confermare se si dimostrerà all'altezza di React ed Angular 2 oppure no.

1.10 Angular 2 vs React.js

Come visto in precedenza dovendo effettuare una scelta conservativa al momento sono da prender in esami due soluzioni per lo sviluppo del frontend di una applicazione che sono il framework web Angular 2 e la libreria web React.js. Nel prosieguo non si riprenderanno i dati di performance utilizzati in precedenza per arrivare a definire i due framework ma si riporteranno alcuni degli aspetti di interesse, non riportati in precedenza, che è necessario analizzare per effettuare una scelta tra i due.

Innanzitutto, entrambi si basano sul concetto di template, ovvero la possibilità di definire componenti e descriverli tramite codice di mark-up

Software Engineering Research & Practices

simile ad html, accompagnato dalla logica di sostituzione che descrive come associare i dati al template per generare il codice html che sarà renderizzato dal browser.

Angular 2 è un framework web onnicomprensivo di tutti i servizi necessari per sviluppare un'applicazione, evitando allo sviluppatore di dover trovare, selezionare e integrare librerie esterne. La comodità di Angular è proprio che una volta appreso nella sua interezza il framework non si ha la necessità di dover fare altre scelte tecnologiche per lo sviluppo. Questo facilita lo sviluppo quando c'è un gruppo di sviluppatori che lavorano sullo stesso progetto, ma di contro richiede un tempo maggiore per poter arrivare a conoscere tutte le funzionalità messe a disposizione dal framework. Inoltre, utilizzando Angular le applicazioni sviluppate contempleranno comunque funzionalità del framework che magari non userai mai.

React, invece, mette a disposizione degli sviluppatori il solo motore di template e di rendering lasciando a questi la scelta e l'integrazione delle librerie necessarie a fornire i servizi utili all'applicazione. Nel lavoro di gruppo questo potrebbe essere un punto di debolezza perché per risolvere problemi similari due sviluppato potrebbero usare librerie esterne differenti, ma di contro il tempo per apprendere il funzionamento del solo motore è sensibilmente più basso del tempo necessario per apprendere l'intero framework Angular 2.

Infine, utilizzando React, a differenza di Angular, nella applicazione ci saranno solo le funzioni che questa utilizza.

Un altro aspetto, particolarmente interessante per le idee di evoluzione di i-mole, è il supporto di Angular per il data binding bidirezionale ovvero l'azione dell'utente provoca l'aggiornamento dello stato del componente e una modifica dello stato genera un cambiamento del componente e il conseguente aggiornamento della interfaccia utente. Il supporto al binding bidirezionale rende l'esperienza di sviluppo molto simile a quella che si ha con un'applicazione wpf dove è presente nativamente il concetto di data binding bidirezionale.

L'aspetto negativo del databinding bidirezionale risiede nella difficolta maggiore nella esecuzione del debugging perché lo stato dell'applicazione è potenzialmente modificabile da tutti i componenti che sottoscrivono gli eventi della interfaccia rendendo difficile capire chi effettivamente ha modificato lo stato e come.

Software Engineering Research & Practices

React supporta il databinding unidirezionale ovvero uguale al serverside render.

1.10.1 Livello di maturità Angular

Angular è un framework open source per lo sviluppo di applicazioni web con licenza MIT, evoluzione di AngularJS. Sviluppato principalmente da Google, la sua prima release è avvenuta il 14 settembre 2016. Angular è stato completamente riscritto rispetto a AngularJS (altro framework per applicazioni web open source, sviluppato nel 2010 dalla stessa Google e dalla community) e le due versioni non sono compatibili. Il linguaggio di programmazione usato per AngularJS è JavaScript mentre quello di Angular è TypeScript.

1.10.2 Livello di maturità React

React invece è una libreria JavaScript per la creazione di interfacce utente. È mantenuto da Facebook e da una community composta di singoli sviluppatori e aziende. Creato un ingegnere di Facebook, è stato influenzato da XHP, una libreria di componenti HTML per PHP. È stato utilizzato inizialmente sul News Feed di Facebook nel 2011 e successivamente su Instagram nel 2012. È stato rilasciato al pubblico al JSConf US del maggio 2013.

1.10.3 Ricapitolando

Ricapitolando Angular, sviluppato da Google, è un framework di sviluppo front-end scritto in TypeScript e ha tutte le funzionalità e i componenti chiave necessari per lo sviluppo di applicazioni web. React, d'altra parte, è un prodotto Facebook, è ampiamente utilizzato per i progetti di Facebook ed è una libreria piuttosto che un framework. Per creare una app utilizzando React, si dovrà integrare una gamma di strumenti di supporto come container, router, bundle di codice, software di gestione dello stato e utility per il test delle app...

1.11 Caratteristiche

1.11.1 Angular

Angular è un framework MVC (MVVM) a tutti gli effetti mentre React è semplicemente una libreria JavaScript. Angular è considerato un framework perché offre forti suggerimenti su come dovrebbe essere strutturata la tua

Software Engineering Research & Practices

applicazione, contenendo al suo interno tutte le funzionalità perché questa possa essere creata. Ha anche molte più funzionalità "out of the box". Non è necessario decidere quali librerie di routing utilizzare o altre considerazioni simili: è possibile iniziare a scrivere codice sin dall'inizio. Tuttavia, uno svantaggio è la minor flessibilità: bisogna usare ciò che Angular dice che si deve usare.

Angular "out of the box" fornisce le seguenti funzionalità:

- Templates, basati su una versione estesa dell'HTML.
- Protezione XSS.
- Dependency injection.
- Richieste Ajax, tramite @ angular/http.
- Routing, tramite @ angular/router.
- Component CSS encapsulation.
- Utility per lo unit-testing dei componenti.
- @angular/forms per la costruzione dei form.

Alcune di queste funzionalità sono integrate nel nucleo del framework e non c'è possibilità di non utilizzarle. Ciò richiede agli sviluppatori di avere familiarità con funzionalità come la dependency injection per creare anche una piccola applicazione Angular. Altre funzionalità come il client HTTP o i form sono completamente opzionali e possono essere aggiunti in base alle necessità.

1.11.2 React

React, d'altra parte, dà allo sviluppatore molta più libertà. Fornisce solo la "View" dell'architettura MVC: è necessario fornire indipendentemente Model e Controller. Per questo motivo, è possibile scegliere una qualsiasi libreria si ritenga opportuna. Inoltre, ogni progetto React è diverso e richiede una scelta indipendente per l'organizzazione della gerarchia e l'architettura delle cartelle.

React fornisce il seguente "out of the box":

- Invece dei classici template, ha JSX, un linguaggio simile a XML costruito su JavaScript
- Protezione XSS
- Nessuna iniezione di dipendenza
- Recupero per richieste Ajax
- Utility per lo unit-testing dei componenti

Software Engineering Research & Practices

Alcune librerie popolari per aggiungere funzionalità sono:

- React-router per il routing
- Redux o MobX per la gestione dello stato
- Enzym per utilità di test aggiuntive

Software Engineering Research & Practices

1.12 Linguaggi, paradigmi e Pattern

1.12.1 React

Parlando di React, alcune delle cose importanti che saltano subito in mente sono: JSX, component e hook.

1.12.1.1 JSX

A differenza della maggior parte dei framework, React non ha un linguaggio di template separato. Invece di seguire un approccio classico di separazione di markup e logica, React ha deciso di combinarli all'interno di componenti utilizzando JSX, un linguaggio simile a XML che consente di scrivere markup direttamente nel codice JavaScript.

Sebbene i meriti del mescolare markup con JavaScript possano essere discutibili, ha un vantaggio indiscutibile: **l'analisi statica del codice**. Se si commette un errore nel markup JSX, il compilatore emetterà un avviso invece di continuare in silenzio. Questo aiuta rilevando immediatamente errori di battitura o altri errori simili.

1.12.1.2 Component

In React è possibile definire i component usando funzioni e classi.

I componenti di classe consentono di scrivere il codice utilizzando le classi ES e strutturare la logica dei componenti in metodi. Consentono inoltre di utilizzare i metodi tradizionali del ciclo di vita di React per eseguire la logica personalizzata quando un componente viene montato, aggiornato, smontato e così via. Anche se questa notazione è più facile da capire per le persone che hanno familiarità con la programmazione OOP, è necessario essere consapevoli di tutte le sottili sfumature che JS ha.

I componenti funzionali sono definiti come funzioni semplici. Spesso sono funzioni pure e forniscono una chiara mappatura tra i props di input e l'output renderizzato. Il codice funzionale è solitamente meno accoppiato e più facile da riutilizzare e testare. Prima dell'introduzione degli hook, i componenti funzionali non potevano essere stateful e non avevano un'alternativa ai metodi del ciclo di vita.

C'è una tendenza tra gli sviluppatori React di abbandonare i componenti di classe a favore di componenti funzionali più semplici, ma con gli,

Software Engineering Research & Practices

funzionalità introdotta più di recente, è possibile osservare un mix di entrambi gli approcci in progetti React più grandi.

1.12.1.3 Hook

Gli hook sono una nuova funzionalità di React introdotta nella versione 16.8. Sono funzioni che consentono di classificare lo stato dei componenti e le caratteristiche del metodo del ciclo di vita nei componenti funzionali. Ci sono due hook forniti da React: useState per la gestione dello stato e useEffect per la creazione di effetti collaterali, come il caricamento dei dati o la modifica manuale del DOM.

Gli hook sono stati introdotti per rendere i componenti funzionali più semplici e componibili. È ora possibile dividere funzioni di grandi dimensioni in parti atomiche più piccole, consentendo di suddividere parti correlate di funzionalità, separandole dalla logica di rendering e riutilizzandole in componenti diversi. Gli hook sono un'alternativa più pulita all'uso di componenti di classe e altri modelli, come funzioni di rendering e componenti di ordine superiore, che possono diventare rapidamente eccessivamente complicati.

React fornisce modi per strutturare la tua applicazione senza coinvolgere molti complicati livelli di astrazione. L'utilizzo di componenti funzionali insieme agli hook consente di scrivere codice più semplice, più atomico e riutilizzabile. Anche se l'idea di combinare codice e modelli potrebbe sembrare controversa, separare la presentazione e la logica dell'applicazione in funzioni diverse consente di ottenere ugualmente ottimi risultati.

1.12.2 Angular

Angular ha anche alcune caratteristiche molto interessanti, a partire dalle astrazioni di base come componenti, i servizi e moduli, l'utilizzo di TypeScript, RxJS e Angular Elements, nonché il suo approccio alla gestione dello stato.

1.12.2.1 Concetti principali

Angular ha un livello di astrazione più alto di React, introducendo così concetti più fondamentali con cui familiarizzare. I principali sono:

Software Engineering Research & Practices

- componenti: definiti come classi ES decorate, sono responsabili dell'esecuzione della logica dell'applicazione e del rendering del modello
- servizi: classi responsabili dell'implementazione della logica di business e applicativa, utilizzate dai componenti
- moduli: essenzialmente Dependency injection container per il cablaggio di componenti, servizi, pipes e altre entità correlate.

Angular fa un uso massiccio di classi e concetti come Dependency injection, che sono meno popolari nel mondo dello sviluppo front-end, ma dovrebbero essere familiari a chiunque abbia esperienza di sviluppo back-end.

1.12.2.2 TypeScript

TypeScript è un nuovo linguaggio basato su JavaScript e sviluppato da Microsoft. È un super set di JavaScript ES2015, ma include funzionalità delle versioni più recenti del linguaggio. È possibile utilizzarlo al posto di Babel per scrivere JavaScript allo stato dell'arte. Dispone inoltre di un potente sistema di analisi per analizzare staticamente il codice utilizzando una combinazione di annotazioni e inferenza di tipo.

C'è anche un vantaggio più sottile. TypeScript è stato fortemente influenzato da Java e .NET, quindi se i tuoi sviluppatori hanno un background in uno di questi linguaggi, è probabile che trovino TypeScript più facile da imparare rispetto al semplice JavaScript. Sebbene Angular sia stato il primo framework importante a adottare attivamente TypeScript, ora sta ottenendo trazione anche in molti altri progetti, come Deno (un runtime nativo di TypeScript), Puppeteer e TypeORM.

È anche possibile (e saggio) usare TypeScript insieme a React.

1.12.2.3 RxJS

RxJS è una libreria che consente una gestione più flessibile di operazioni ed eventi asincroni. È una combinazione dei modelli Observer e Iterator uniti alla programmazione funzionale. RxJS consente di trattare qualsiasi cosa come un flusso continuo di valori ed eseguire varie operazioni su di esso come mappatura, filtraggio, divisione o unione.

Software Engineering Research & Practices

La libreria è stata adottata da Angular nel suo modulo HTTP anche per alcuni usi interni. Quando viene eseguita una richiesta HTTP, restituisce un Observable invece della solita Promise. Questo approccio apre la porta a possibilità interessanti, come la possibilità di annullare una richiesta, riprovare più volte o lavorare con flussi di dati continui come WebSocket. Ma questa è solo la superficie. Per padroneggiare RxJS, è necessario conoscere i diversi tipi di Observable, Subject, nonché un centinaio di metodi e operatori.

1.12.2.4 Gestione dello stato

Simile a React, i componenti di Angular hanno un concetto di stato del componente. I componenti possono memorizzare i dati nelle loro proprietà di classe e "legare" i valori ai loro template. Se si desidera condividere lo stato attraverso l'applicazione, è possibile spostarlo in un servizio stateful che può essere successivamente inserito nei componenti. Poiché la programmazione reattiva e RxJS sono pilastri fondanti in Angular, è comune fare uso di Observable per ricalcolare parti dello stato in base a qualche input. Ciò, tuttavia, può diventare complicato nelle applicazioni più grandi poiché la modifica di alcune variabili può attivare una cascata multidirezionale di aggiornamenti che è difficile da seguire; tuttavia, esistono librerie per Angular che consentono di semplificare la gestione dello stato su larga scala.

1.12.2.5 Angular Element

Gli Angular Element forniscono un modo per creare pacchetti di componenti Angular come elementi personalizzati. Conosciuti anche come componenti Web, gli elementi personalizzati sono un modo standardizzato, indipendente dal framework, per creare elementi HTML personalizzati controllati dal codice JavaScript. Una volta definito un elemento di questo tipo e aggiunto al registro del browser, verrà automaticamente visualizzato ovunque sia referenziato nell'HTML. Gli elementi angolari forniscono un'API che crea il wrapper necessario per implementare l'API del componente personalizzato e farlo funzionare con il meccanismo di rilevamento delle modifiche di Angular. Questo meccanismo può essere utilizzato per incorporare altri componenti o intere applicazioni Angular nell'applicazione host,

Software Engineering Research & Practices

potenzialmente scritti in un framework diverso con un ciclo di sviluppo diverso.

1.13 Ecosistema

1.13.1 Angular

1.13.1.1 Angular CLI

Una tendenza popolare con i framework moderni è avere uno strumento CLI che ti aiuta a eseguire il bootstrap del tuo progetto senza dover configurare manualmente la build. Angular ha Angular CLI per questo. Permette di generare ed eseguire un progetto con solo un paio di comandi. Tutti gli script responsabili della creazione dell'applicazione, dell'avvio di un server di sviluppo e dell'esecuzione dei test sono nascosti in node_modules. È possibile anche usarlo per generare nuovo codice durante lo sviluppo e installare le dipendenze.

Angular introduce un nuovo modo interessante di gestire le dipendenze del progetto. Quando si utilizza *ng add*, è possibile installare una dipendenza e verrà automaticamente configurata per l'utilizzo. Ad esempio, quando esegui *ng add @angular/material*, Angular CLI scarica Angular Material dal registro npm ed esegue il suo script di installazione che configura automaticamente l'applicazione per utilizzare Angular Material. Questo viene fatto usando Angolar Schematics. Schematics è uno strumento per il flusso di lavoro che consente alle librerie di apportare modifiche al codice base. Ciò significa che gli autori della libreria possono fornire metodi automatici per risolvere problemi di incompatibilità con le versioni precedenti che si potrebbero incontrare durante l'installazione di una nuova versione.

1.13.1.2 Librerie delle componenti

Un aspetto importante nell'utilizzo di qualsiasi framework JavaScript è poterli integrare con una libreria di componenti a scelta, per evitare di dover costruire tutto da zero. Angular offre integrazioni con la maggior parte delle librerie di componenti più diffuse e con le proprie librerie native.

Per esempio:

- ng-bootstrap per l'utilizzo dei widget Bootstrap

Software Engineering Research & Practices

- Material UI, per i componenti di Material Design di Google
- NG-ZORRO, una libreria di componenti che implementano le specifiche Ant Design
- Onsen UI for Angular, una libreria di componenti per applicazioni mobili
- PrimeNG, una raccolta Angular Component

1.13.1.3 Librerie di gestione dello stato.

Se le funzionalità di gestione dello stato nativo non sono sufficienti, in quest'area sono disponibili diverse librerie di terze parti popolari.

- Il più popolare è NgRx. È ispirato a Redux di React ma utilizza anche RxJS per guardare e ricalcolare i dati nello stato. L'uso di NgRx aiuta a imporre un flusso di dati unidirezionale comprensibile, oltre a ridurre l'accoppiamento nel tuo codice.
- NGXS è un'altra libreria di gestione dello stato ispirata a Redux. A differenza di NgRx, NGXS si sforza di ridurre il codice boilerplate utilizzando le moderne funzionalità TypeScript e migliorando la curva di apprendimento e l'esperienza di sviluppo complessiva.
- Akita consente di salvare lo stato in più fonti, applicare aggiornamenti immutabili e utilizzare RxJS per interrogare e trasmettere i valori.

1.13.1.4 Ionic Framework

Ionic è un framework popolare per lo sviluppo di applicazioni mobili ibride. Fornisce un container Cordova che è ben integrato con Angular e una libreria di Material Component. Usandolo, è possibile facilmente configurare e creare un'applicazione mobile.

Angular universal

Angular universal è un progetto che raggruppa diversi strumenti per abilitare il rendering lato server per le applicazioni Angular. È integrato con Angular CLI e supporta diversi framework Node.js, come Express e Hapi, nonché con .NET core.

1.13.1.5 Augury

Augury è un'estensione del browser per Chrome e Firefox che aiuta a eseguire il debug delle applicazioni Angular in esecuzione in modalità di sviluppo. Puoi utilizzarlo per esplorare l'albero dei componenti, monitorare il rilevamento delle modifiche e ottimizzare i problemi di prestazioni.

Software Engineering Research & Practices

1.13.1.6 Compodoc

Compodoc è un generatore di documentazione statica per Angular. Simile ad altri generatori di documentazione, può creare documentazione HTML statica basata sui commenti TSDoc nel codice. Compodoc, tuttavia, viene fornito con funzionalità specifiche per Angular, come la navigazione della struttura del modulo, i percorsi e la classificazione delle classi in componenti, servizi e così via.

1.13.1.7 Ngx-admin

Ngx-admin è un framework per la creazione di dashboard personalizzati con Angular e l'utilizzo di Nebular o Angular Material come librerie di componenti.

1.13.2 React

1.13.2.1 Create React App

Create React App è un'utilità CLI per React per mettere in piedi rapidamente nuovi progetti. Simile ad Angular CLI, consente di generare un nuovo progetto, eseguire l'app in modalità di sviluppo o creare un bundle di produzione. Utilizza Jest per i test unitari, supporta la profilazione delle applicazioni utilizzando variabili di ambiente, proxy back-end per lo sviluppo locale, TypeScript, Sass, PostCSS e molte altre funzionalità.

1.13.2.2 Librerie di componenti

Simile ad Angular, React ha un'ampia varietà di librerie di componenti tra cui scegliere:

- ant-design
- Material UI
- react-bootstrap
- Semantic UI
- Onsen UI, ottimizzata per applicazioni mobili
- Blueprint, per la creazione di applicazioni desktop

1.13.2.3 Librerie di gestione dello stato

L'introduzione degli hook ha sicuramente aiutato la gestione dello stato in React. Ci sono discussioni in corso se c'è anche bisogno di una libreria di gestione statale di terze parti. Anche se gli hook risolvono la necessità immediata di lavorare con lo stato, altre librerie possono comunque spingere oltre consentendoci di utilizzare modelli di implementazione collaudati nel tempo, molte librerie aggiuntive e strumenti di sviluppo.

- Redux è una libreria per la gestione dello stato ispirata a Flux, ma con alcune semplificazioni. L'idea chiave di Redux è che l'intero stato dell'applicazione è rappresentato da un singolo oggetto, che viene mutato da funzioni chiamate riduttori. I riduttori stessi sono funzioni pure e vengono implementati separatamente dai componenti.
- MobX è una libreria alternativa per la gestione dello stato di un'applicazione. Invece di mantenere lo stato in un unico archivio immutabile, come fa Redux, incoraggia a memorizzare solo lo stato minimo richiesto e deriva il resto da esso. Fornisce una serie di decoratori per definire osservabili e osservatori e introdurre logica reattiva nello stato.

1.13.2.4 Librerie di stili

A differenza di Angular, React non fornisce funzionalità di incapsulamento CSS native, quindi è necessario cercare soluzioni di terze parti. Esistono numerose soluzioni a questo problema, senza un leader chiaro tra di loro. Alcuni di quelli popolari sono:

- Styled Components, una libreria che permette di creare componenti React con il tuo stile applicato, così come lo stile dei tuoi componenti
- CSS Modules, che consentono di importare file CSS e generare nomi di classi isolati univoci per fare riferimento agli stili
- Emotion, che combina gli approcci dei componenti stilizzati e dei moduli CSS in un'unica libreria

1.13.2.5 PropTypes

PropTypes è una funzionalità opzionale di React che consente di introdurre la convalida delle prop in fase di esecuzione dei componenti. A differenza dell'utilizzo del controllo del tipo statico con TypeScript, PropTypes eseguirà i controlli del tipo quando l'applicazione è effettivamente in esecuzione. Ciò è particolarmente utile quando si sviluppano librerie quando non si può

Software Engineering Research & Practices

essere certi che i propri client stiano utilizzando TypeScript. A partire da React 15.5, i PropTypes sono stati spostati in una libreria separata di PropTypes e ora sono completamente opzionali. Considerati i suoi vantaggi, ti consigliamo di utilizzarlo per migliorare l'affidabilità della tua applicazione.

1.13.2.6 React Native

React Native è una piattaforma sviluppata da Facebook per la creazione di applicazioni mobili native utilizzando React. A differenza di Ionic, che produce un'applicazione ibrida, React Native produce un'interfaccia utente veramente nativa. Fornisce un set di componenti React standard che sono associati alle loro controparti native. Consente inoltre di creare i componenti e associarli al codice nativo scritto in Objective-C, Java o Swift.

1.13.2.7 Next.js

Next.js è un framework per il rendering lato server delle applicazioni React. Fornisce un modo flessibile per eseguire il rendering completo o parziale dell'applicazione sul server, restituire il risultato al client e continuare nel browser. Cerca di semplificare il compito complesso di creare applicazioni universali, quindi l'installazione è progettata per essere il più semplice possibile, con una quantità minima di nuove primitive e requisiti per la struttura del progetto.

1.13.2.8 React Admin

React-admin è un framework per la creazione di applicazioni CRUD a partire da API REST o GraphQL esistenti. Viene fornito con funzionalità utili, come un'interfaccia utente costruita con Material Design, internazionalizzazione, tematizzazione, convalida dei dati e altro ancora.

1.13.2.9 UI Development Environments

Una delle principali tendenze nello sviluppo front-end negli ultimi due anni è stata il boom di strumenti di sviluppo che consentono di sviluppare, testare

Software Engineering Research & Practices

e documentare il componente in modo interattivo e separato dall'applicazione. Storybook si è affermato come uno dei leader in questo settore, con il supporto sia per React che per Angular. Tuttavia, ci sono altre alternative per React.

- React Styleguidist, analogamente a Storybook, consente di creare documentazione interattiva dei tuoi componenti. A differenza di Storybook, l'interfaccia utente generata sembra più un readme interattivo che un set separato di storie. Mentre Storybook brilla come ambiente di sviluppo, Styleguidist è più uno strumento di documentazione.

1.13.2.10 Testing Helpers

Il test dei componenti dell'interfaccia utente di solito comporta la necessità di eseguire il loro render in un ambiente sandbox, simulare l'interazione dell'utente e convalidare i risultati di output. Queste attività di routine possono essere semplificate utilizzando i testing helper appropriati. Per Angular, questa funzionalità è coperta da TestBed, integrata nel framework. Per React, ci sono due candidati popolari: Enzyme e Testing Library.

- Enzyme è la scelta standard de facto. Consente di eseguire il rendering dei componenti in un DOM completo o superficiale e di interagire con il componente di rendering. Segue principalmente un approccio di test white box, in cui i test possono fare riferimento ad alcuni degli interni del componente come i suoi componenti figli, props o stati.
- Testing Library segue un approccio diverso e spinge a interagire con i componenti come farebbe un utente, senza conoscere l'implementazione tecnica. I test creati in questo modo sono generalmente meno fragili e più facili da mantenere. Sebbene sia più popolare con React, la libreria di test è disponibile anche per Angular.

1.13.2.11 Gatsby

Gatsby è un generatore di siti Web statici che utilizza React.js. Consente di utilizzare GraphQL per eseguire query sui dati per i tuoi siti Web definiti in

Software Engineering Research & Practices

Markdown, YAML, JSON, API esterne e sistemi di gestione dei contenuti più diffusi.

1.13.2.12 React 360

React 360 è una libreria per la creazione di applicazioni di realtà virtuale per browser. Fornisce un'API React dichiarativa costruita sopra le API del browser WebGL e WebVR, rendendo così più semplice la creazione di esperienze di realtà virtuale a 360 °.

1.13.2.13 React Developer Tools

React Dev Tools è un'estensione del browser per Chrome per il debug delle applicazioni React che consente di attraversare l'albero dei componenti React e vedere i loro prop (valori trasferiti da una componente all'altra in fase di rendering) e gli stati.

1.14 Adozione

1.14.1 React

Con React, la prima cosa che si è incontra è JSX. Potrebbe sembrare scomodo l'utilizzo per alcuni sviluppatori. Tuttavia, non aggiunge molta complessità: solo espressioni, che sono JavaScript e una sintassi speciale simile all'HTML. Si dovrà anche imparare a scrivere componenti, utilizzare proprietà per la configurazione e gestire lo stato interno. Non è necessario imparare una nuova sintassi del template, poiché tutto questo è semplice JavaScript. Sebbene React supporti componenti basati su classi, con l'introduzione degli hook, lo sviluppo funzionale sta diventando sempre più popolare. Ciò richiederà la comprensione di alcuni modelli di sviluppo funzionale di base.

La documentazione ufficiale è un ottimo posto per iniziare a imparare React. Una volta terminata, si acquisisce familiarità con il router. Il React Router potrebbe sembrare leggermente complesso e non convenzionale, ma niente di cui preoccuparsi. A seconda delle dimensioni, della complessità e dei requisiti del progetto, si dovranno trovare e imparare ad utilizzare alcune librerie aggiuntive e questa potrebbe essere la parte complicata. Ma dopo questo, tutto dovrebbe andare liscio.

Software Engineering Research & Practices

Anche gli sviluppatori con un background di sviluppo back-end ed esperienza limitata nello sviluppo front-end sono in grado di recuperare rapidamente. I messaggi di errore che si potrebbero incontrare lungo il percorso sono generalmente chiari e forniscono spiegazioni su come risolvere il problema sottostante.

Lo svantaggio è che si dovrà investire tempo nella scelta delle librerie per supportare le tue attività di sviluppo. Data la notevole quantità, questo può rappresentare un problema.

Sebbene TypeScript non sia richiesto per React, è vivamente consigliato di valutarlo e adottarlo in ogni progetto.

1.14.2 Angular

Imparare Angular introdurrà lo sviluppatore a una quantità maggiore di nuovi concetti rispetto a React. Prima di tutto, sarà necessario familiarizzare con TypeScript. Per gli sviluppatori con esperienza in linguaggi tipizzati staticamente come Java o .NET, questo potrebbe essere più facile da capire rispetto a JavaScript, ma per gli sviluppatori JavaScript puri, ciò potrebbe richiedere un certo sforzo.

Il framework stesso è ricco di argomenti da apprendere, a partire da quelli di base come moduli, dependency injection, decorator, component, servizi, pipe, modelli e directives e ad argomenti più avanzati come rilevamento delle modifiche, zone, compilazione AoT e Rx.js. Queste tematiche sono tutte trattate nella documentazione ufficiale. Rx.js è un argomento pesante di per sé ed è descritto in modo molto dettagliato sul sito ufficiale. Sebbene relativamente facile da usare a livello di base, diventa più complicato quando si passa ad argomenti avanzati.

1.14.3 Conclusioni

La curva di apprendimento iniziale per Angular è più alta rispetto a React. La maggiore numerosità di nuovi concetti e di funzioni disponibili richiede un tempo maggiore per essere appreso. E anche dopo che hai iniziato, l'esperienza potrebbe essere un po' difficile, dal momento che bisogna tenere a mente cose come la gestione delle sottoscrizioni in Rx.js, le prestazioni di rilevamento delle modifiche e le "bananas in a box" (sezione reale dalla documentazione ufficiale).

Software Engineering Research & Practices

1.15 Community Feedback

Entrambi i framework sono molto popolari e hanno comunità di seguaci e sostenitori. Le statistiche di download npm mostrano quasi cinque volte più download per React rispetto ad Angular. Le tendenze di Google riportano anche più ricerche di React in tutto il mondo.

Il rapporto sullo stato di JavaScript del 2019 elenca React come il framework front-end più popolare, con Angular che è il penultimo, precedendo solo Ember. Il 71% dei partecipanti ha affermato di aver utilizzato React e che lo userebbe di nuovo. Solo il 21% dei partecipanti ha detto lo stesso di Angular, e il 35% ha affermato di aver utilizzato Angular e non l'avrebbe usato di nuovo (solo l'8% lo ha detto di React).

Le tendenze delle assunzioni di Hacker News per il 2019 elencano React come la tecnologia più ricercata dai dipendenti per più di 31 mesi consecutivi.

Stack Overflow elenca React come il secondo framework più popolare dopo jQuery. Angular è elencato come il terzo. Il loro rapporto Framework Web più amato, temuto e ricercato dipinge un quadro simile a quello degli altri.

Il rapporto State of Developer Ecosystem 2020 di Jet Brains conferma la posizione di React come il framework front-end più popolare.

1.16 Scelta dei framework

In questo capitolo, abbiamo riportato le peculiarità dei framework più diffusi per sviluppare le applicazioni web, analizzato le caratteristiche principali per capire quale scegliere in base alle esigenze progettuali.

Il framework web per sviluppare una SPA o una Applicazione Complessa è importante perché a valle della creazione della applicazione sarà molto difficile cambiarlo in quanto significherebbe riscrivere gran parte del codice dell'interfaccia utente. Ovviamente, l'impatto del framework web è limitato alla sola interfaccia utente se e solo se il sistema è progettato in modo modulare e se l'interfaccia comunica con lo strato di business utilizzando esclusivamente i servizi o microservizi che questo gli mette a disposizione.

Contestualizzato la scelta al mondo dei framework, benché ci siano una moltitudine di soluzioni che fioriscono ogni giorno ma che contestualmente cessano di essere supportati, è opportuno scegliere uno dei due framework citati precedentemente poiché oltre ad essere molto stabili e largamente

Software Engineering Research & Practices

adottati, sono gestiti o hanno tra i contributori grossi vendor; in più sono open source (licenza MIT per la precisione).

Un altro punto di attenzione, per la scelta del framework, riguarda l'ottimizzazione della curva di apprendimento del gruppo di lavoro che deve sviluppare il progetto. Difatti, se il gruppo di lavoro conosce lo sviluppo di applicazioni desktop con WPF oppure lo sviluppo di applicazioni web scritte con c# o ASP.NET allora la scelta meno invasiva è Angular perché ha un modello di sviluppo basato su template e databinding bidirezionale. Inoltre, l'utilizzo di Angular consente di utilizzare un unico framework per sviluppare tutto il necessario senza richiedere l'attuazione del processo di ricerca, selezione e integrazione di eventuali librerie di terze parti e questo semplificherebbe i processi decisionali tecnologici.

In questa visione, React non è da considerarsi poiché risulta essere più adatto a realizzare applicazioni basate su databinding monodirezionale, e questo consente di sviluppare applicazioni web di cui solo una parte è dinamica. È bene sottolineare che nel caso in cui si decidesse di implementare il modello Single Page Application il framework React potrebbe essere introdotto parzialmente ed in modo meno invasivo rispetto ad Angular.

Per concludere, allo stato attuale considerando la quantità degli aspetti di contesto non definiti puntualmente ma presi in considerazione sono:

- Tipo di applicazione (Single Page Application, Complex Application);
- Composizione e conoscenze pregresse del gruppo di lavoro;
- Strutturazione del gruppo di lavoro;
- Piano delle attività evolutive e migliorative di dettaglio del prodotto;
- Livello di qualità (ISO 25010) del prodotto, modularità e orientamento ai microservizi;
- Stabilità del prodotto e della sua configurazione.

Considerando anche, la velocità con cui la tecnologia di supporto si evolve, vedasi l'attuale rilascio del nuovo Framework Core Microsoft, è necessario che le attività di produzione del prototipo eseguite attenendosi strettamente ai principi chiave del progetto, ovvero: microservizi, sviluppo per componenti autonomiche e modulari, ciclo di sviluppo agile e delivery delle

Software Engineering Research & Practices

verticalizzazioni attuato utilizzando principi e strumenti propri delle linee di prodotto.

Software Engineering Research & Practices

BASI DI DATI

Come visto sino a questo momento, lo scenario i-mole vede da una parte l'insieme dei dispositivi software e hardware che si realizzeranno per il progetto e dall'altro la base di dati che servirà per memorizzare i dati raccolti da tali dispositivi allo scopo di metterli anche a disposizione di applicazioni di terze parti come, ad esempio, sistemi di Big Data o data mining utilizzabili per estrarre dai dati informazioni tacite. In questa visione, lo strumento che consente di comunicare con i dispositivi ed il database è il middleware MOM.

Venendo, quindi alla archiviazione dei dati inviati dai dispositivi IoT, il problema più grande è relativo alla quantità di dati che questi riescono a generare ed ai relativi tempi per memorizzarli e recuperarli in modo efficiente. Basti pensare che un singolo dispositivo di temperatura ambientale se rileva la temperatura ogni secondo genera in un solo giorno 86400 dati e in un anno più di 31 milioni di dati. Adesso moltiplichiamo questi valori per tutti i sensori utilizzabili in un qualsiasi contesto è pacifico che è importante scegliere un supporto tecnologico per memorizzare i dati che sia in grado di rendere fruibili questi dati.

Le soluzioni tecnologiche presenti sul mercato per la memorizzazione dei dati sono di due tipi database relazionali (SQL) e non relazionali (NO-SQL). Nell'ambito IoT nonostante, la relativamente giovane età dei database NoSQL, l'utilizzo di questi database ricopre una importante fetta di mercato così come si può rilevare da uno dei sondaggi di riferimento nel settore " Eclipse iot" del 2020.

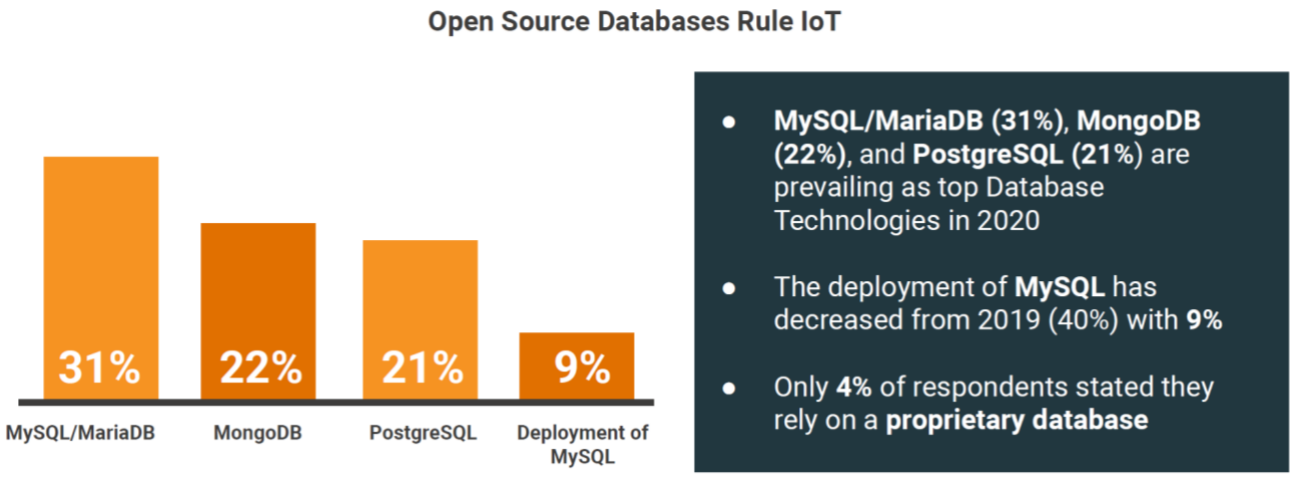


Figura17 2020 IoT Developer Survey utilizzo DB

Software Engineering Research & Practices

SQL, acronimo di Structured Query Language, come suggerisce il nome, è un linguaggio volto all'interrogazione di dati altamente strutturati. Sebbene esista uno standard SQL, nella realtà dei fatti chiunque abbia usato più di un motore SQL, sa bene che ognuno ha le sue peculiarità. Ciononostante, la sintassi di base resta sostanzialmente la stessa e, in fin dei conti, migrare da un engine all'altro ha un costo relativamente ridotto. Sull'altra faccia della medaglia ci sono invece i database NoSQL, dove il prefisso NO significa esattamente quello che ci si aspetta: Not Only SQL. I database NoSQL nascono dunque con l'idea di rappresentare dati eterogenei, per i quali voler forzare una struttura risulterebbe dispendioso e complesso da attuare.

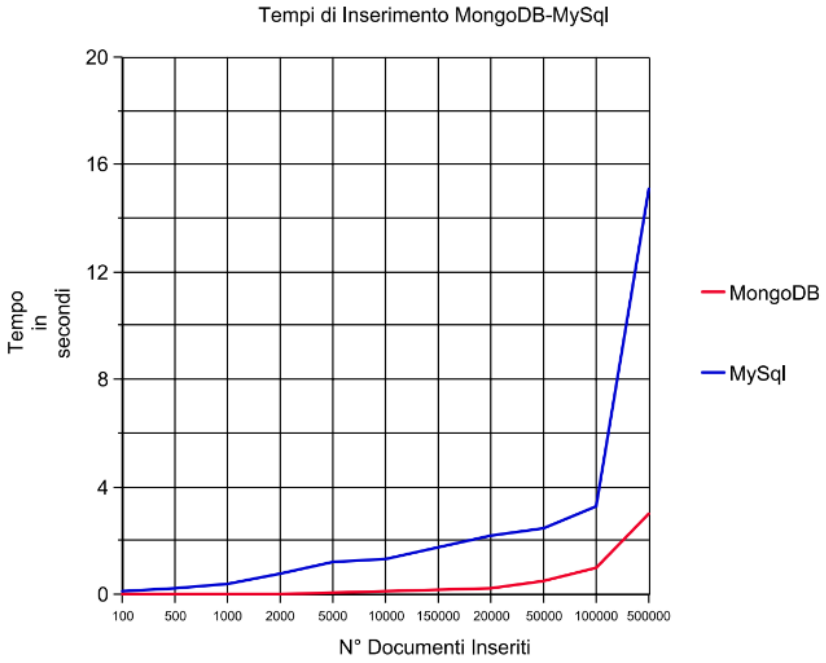
Il punto focale da prendere in considerazione per effettuare una scelta, escludendo fault di progettazione della base dati, è relativo alle prestazioni di memorizzazione e recupero dei dati e rispetto a questo aspetto i NO-SQL risultano essere vincenti, allo scopo di chiarire i motivi che hanno portato alla scelta si riportano le misurazioni di prestazioni, rilevate da due differenti, fonti utilizzate ed utilizzabili per confrontare un database relazionale tradizionale (MySQL) con un database NO-SQL (MongoDB). Sono stati selezionati i database MongoDB e MySQL perché: sono distribuiti con licenza Open e quindi non richiedono aggravi di costo in caso di adozione; sono quelli che hanno una community più grande e attiva e questo assicura che un eventuale difetto rilevato venga risolto velocemente; dispongono di una vasta documentazione che consente facilmente di apprendere quanto necessario per l'utilizzo degli stessi; risultano essere la due soluzioni più utilizzate e quindi in ottica di integrazione di i-mole con soluzioni di terze parti sono le tecnologie che consentirebbero di minimizzare i problemi di comunicazione.

Software Engineering Research & Practices

Fonte 1

Confronto tempi inserimento Mysql vs. MongoDB		
Num. documenti inseriti	Tmp. MongoDB	Tmp. MySql
100	0.004 sec	0.125 sec
500	0.009 sec	0.219 sec
1.000	0.017 sec	0.391sec
2.000	0.026 sec	0.766 sec
5.000	0.058 sec	1.188 sec
10.000	0.136 sec	1.312 sec
15.000	0.151 sec	1.735 sec
20.000	0.202 sec	2.172 sec
50.000	0.481 sec	2.438 sec
100.000	0.96 sec	3.297 sec
500.000	2.998 sec	15.094 sec

Confronto tempi inserimento dati in MySql e MongoDB



Tempi di esecuzione Query			
MongoDB senza Ind.	MongoDB con Ind.	MySql senza Ind.	MySql con Ind.
0.26 sec	0.001 sec	4.266 sec	0.109 sec

[2] Informazioni rilevate dalla sezione documentale dell'Università di Venezia: http://dspace.unive.it/bitstream/

Software Engineering Research & Practices

Fonte 2

Documenti archiviati	MySQL con base dati normalizzata	MySQL con base dati denormalizzata	Mongo DB con base dati denormalizzata
2.000	0.03 secondi	0.01 secondi	0.044 secondi
500.000	12.66 secondi	3.56 secondi	3.2 secondi
1.500.000	2 min 46 secondi	33.95 secondi	9.66 secondi

Come si può notare **maggiore è il volume di dati e più conveniente diventa l'adozione di un database NoSQL** e nel nostro caso parliamo di un volume di dati elevato.

Volendo quindi riassumere i motivi della scelta dell'utilizzo di un database NO-SQL possiamo farlo nei seguenti punti:

- **elevata velocità computazionale**, anche al crescere del volume dei dati, legata alla mancanza di operazioni di aggregazione dei dati (no join) e ad una implementazione del database ottimizzata per la gestione di enormi quantità di dati.
- **riduzione significativa dei tempi di sviluppo**, grazie alla definizione di logiche di lettura dati molto più semplici rispetto a quelle da scrivere con database relazionali che operano su dati normalizzati.
- **supporto per la scalabilità orizzontale**, cioè la capacità di garantire tempi di risposta sostanzialmente invariati al crescere del carico, tramite l'aggiunta di nuovi server.
- **schemaless**, cioè i database NoSQL sono privi di schema in quanto l'entità contiene tutti i campi necessari, senza necessità di una definizione formale e rigida del suo contenuto. In questo modo, è possibile arricchire le applicazioni di nuovi dati e informazioni, senza dover sottostare ad una rigida struttura dei dati.

Infine, relativamente alla memorizzazione dei dati è bene riportare all'attenzione che per poter funzionare, l'IoT ha bisogno di raccogliere e

Software Engineering Research & Practices

archiviare una grossa mole di dati. Ma a che servono tutti questi dati raccolti? Per far funzionare correttamente l'internet of things affinché sia davvero utile è importante processare raccogliere e analizzare grandi volumi di dati real time (ad esempio dai sensori, dai semafori, e da qualsiasi dispositivo IoT connesso). Questi dati possono essere utilizzati per diversi scopi come, ad esempio, nell'industria per migliorare la sicurezza dei luoghi di lavoro e la produttività. **Per questo c'è bisogno di sistemi integrati tra big data, database NoSQL e dati IoT**.

L'Internet delle Cose è una delle nuove frontiere, ormai in realtà consolidata, dell'uso della rete internet. Non più solo le persone o le "persone giuridiche", le imprese, sono riconoscibili sulla rete Internet, ma anche le *cose* possono esserlo. Cose, oggetti, strumenti che acquisiscono intelligenza, ovvero capacità di rilevare informazioni e di comunicarle. L'Internet delle Cose è una vera e propria Nuova Internet proprio perché apre prospettive un tempo inimmaginabili, in cui gli oggetti assumono un ruolo attivo grazie al fatto di essere in rete e di inviare e ricevere dati sulla rete.

Una videocamera non è più solo nella condizione di inviare dati e immagini, ma è nella condizione di farlo in modo intelligente, in funzione ad esempio delle immagini riprende o della temperatura o della luminosità. In questo modo è in grado di adattare il proprio comportamento in funzione di parametri di interesse che possono evolvere nel corso del tempo.

Il nostro orologio può ricordarci appuntamenti e può verificare se effettivamente li rispettiamo, ad esempio se siamo o non siamo in un certo luogo a una certa ora o, se, sempre in funzione della distanza, dobbiamo affrettarci perché altrimenti facciamo tardi.

Le confezioni di prodotti alimentari possono trasferire importanti informazioni sulla qualità del prodotto, sul modo in cui è stato realizzato e su tutti coloro che hanno partecipato alla produzione e che fanno parte della filiera food. O ancora la confezione di un farmaco ci può avvertire se non lo stiamo assumendo come stabilito e ci può dire come ovviare e a una eventuale dimenticanza.

Per non parlare poi delle automobili che possono dialogare costantemente con l'ambiente circostante e possono facilitare la nostra guida, aumentando comodità e sicurezza.

Dopo aver definito le componenti del sistema che descrive una delle possibili soluzioni al problema proposto, di seguito viene proposto in dettaglio ogni modulo analizzando i loro punti di forza e i punti di debolezza; sono elencati,

Software Engineering Research & Practices

inoltre, i software disponibili sul mercato che rispondono ai bisogni e infine viene scelta e motivata l'architettura che sarà integrata nella soluzione finale.

1.17 Database management system (database NoSQL)

I database non relazionali, pur non essendo una novità assoluta, hanno registrato una crescita esponenziale nel loro sviluppo e utilizzo negli ultimi anni grazie al sempre più crescente bisogno di scalare in orizzontale, dove i classici RDBMS (database relazionali) presentano diverse limitazioni. Basti pensare ai sempre più in voga sistemi cloud dove i nodi possono diventare veramente molti oppure al mondo IoT, gestire un RDBMS in un ambiente del genere risulta complicato e la potenza computazionale non viene sfruttata in modo ottimale (pensate per esempio alla replicazione con MySQL dove il log binario costituisce già di per sé un enorme calo prestazionale). Al giorno d'oggi sono nati diversi tipi di applicazioni sempre più esigenti sotto questi aspetti:

- Archiviazione efficiente dei big data: le grandi applicazioni come un motore di ricerca hanno bisogno di DBMS efficienti, con bassa latenza, performanti e con la capacità di gestire milioni di accessi concorrenziali in lettura e in scrittura.
- Elevata scalabilità e alta disponibilità: con il crescente numero di richieste di dati simultanee, il database deve essere in grado di supportare crescite a livello di utenti e dimensioni, mantenendo costante e rapido il servizio offerto.

Quindi una delle principali motivazioni per l'uso di tali database è rappresentata dalla scalabilità. La scalabilità è un requisito sempre più importante per le applicazioni web, e ciò è dovuto a molti fattori: l'esplosione del numero di utenti della rete, la sempre maggiore diffusione di OpenID, quindi la sinergia tra le varie community e tra esse e i fornitori di servizi, ma anche la crescente disponibilità di dispositivi con accesso ad Internet come smartphone, tablet e altri dispositivi portatili e il crescente numero di dispositivi IoT collegati.

Software Engineering Research & Practices

Vediamo allora come deve essere un sistema, in concreto, per essere scalabile. Tradizionalmente la scalabilità di un sistema è la sua proprietà di avere un incremento di prestazioni che è grosso modo direttamente proporzionale al suo aumento di risorse. Infatti, se l'aumento delle prestazioni non è direttamente proporzionale, significa che il sistema spreca risorse. Ma le risorse possono essere aumentate in due modi, ognuno dei quali genera un diverso modello di scalabilità: entrambi questi modelli di scalabilità hanno vantaggi e svantaggi.

1.17.1.1 Scalabilità orizzontale

La scalabilità orizzontale (scale out) si ha se l'aumento delle risorse si riferisce all'aumento dei nodi nel sistema, cioè il sistema riesce a parallelizzare il carico di lavoro.

Un tipico esempio di applicazione è il sito web statico: esso è quasi perfettamente scalabile orizzontalmente: raddoppiando il numero di server che lo ospitano si ottiene un esatto raddoppio dei potenziali utenti serviti.

Il vantaggio più importante dato da questo tipo di scalabilità è il costo: infatti, con un'applicazione perfettamente scalabile, si potrebbero impiegare molti nodi a basso costo, ottenendo anche una maggiore prevedibilità del costo marginale. Un altro vantaggio è la maggior fault-tolerance: l'errore in un nodo non pregiudica totalmente il funzionamento dell'applicazione.

Gli svantaggi risiedono nei maggiori sforzi in fase di progettazione perché l'applicazione deve innanzitutto supportare questo modello di scalabilità; inoltre, l'applicazione deve essere facile da amministrare, per ridurre i costi d'installazione.

1.17.1.2 Scalabilità verticale

La scalabilità verticale (scale up) si ottiene quando, per aumentare le prestazioni dell'intero sistema, si aumentano le risorse di un singolo nodo del sistema, ad esempio utilizzando una CPU con frequenza maggiore o incrementando la memoria disponibile. Il vantaggio di questo tipo di scalabilità è che generalmente non è necessario modificare le applicazioni, né sono richiesti interventi amministrativi. Lo svantaggio consiste innanzitutto nel costo, perché l'aggiornamento spinto di una macchina può essere economicamente molto più gravoso dell'acquisto di una ulteriore macchina di pari potenza.

Software Engineering Research & Practices

1.17.2 Caratteristiche dei database NoSQL

I principali vantaggi di un database NoSQL sono la capacità di leggere e scrivere i dati rapidamente, supporto alla memoria di massa, la facilità di espansione e il basso costo. Tuttavia, i DBMS NoSQL, hanno anche alcune inadeguatezze che bisogna prendere in considerazione al momento della scelta di uno spazio di archiviazione per i dati. Una delle mancanze più grandi è la mancanza del supporto allo standard di settore SQL, la mancanza delle transazioni a logica ACID, e quei difetti di gioventù che però vengono di giorno in giorno risolti coprendo quel gap con i ben più maturi database relazionali.

Le caratteristiche descritte in precedenza sono quelle comuni a tutti i DBMS NoSQL, tuttavia ogni database è conforme a diversi modelli di dati e al teorema CAP.

1.17.2.1 Modelli di dati

I database NoSQL, anche perché' spesso servono a garantire proprietà diverse dai RDBMS, utilizzano dei modelli logici di dati di tipo non relazionale. Questi modelli sono riconducibili grosso modo a quattro grandi famiglie. Vedremo che questi modelli logici non hanno la stessa potenza espressiva del modello relazionale, proprio per ciò che riguarda le relazioni tra entità. Infatti, per sintetizzare, per la scalabilità orizzontale dei RDBMS, i più grandi problemi sono dovuti a JOIN e a foreign key. Un'altra proprietà dei database NoSQL è di essere schemaless, con i pro e i contro che ne conseguono: più facilità di deploy ma anche, talvolta, maggiori difficoltà d'interrogazione.

Passiamo dunque ad analizzare le quattro tipologie di modelli di dati più comuni. Spesso un database non ne utilizza uno in modo rigoroso: in molti casi quello che succede è che vengono utilizzati dei modelli compositi che assomigliano di volta in volta all'uno o a all'altro modello.

Il modello chiave-valore

Nel modello chiave-valore ogni valore corrisponde ad una chiave; sebbene la sua struttura sia semplice, è notevolmente più veloce di un database relazionale, supporta gli archivi di massa e permette un accesso in scrittura e in lettura dei dati altamente concorrenziale. Inoltre, è stato incluso il supporto alle operazioni di query e modifica dei dati attraverso chiave primaria.

Software Engineering Research & Practices

Proprio come in una mappa, il valore è un oggetto del tutto trasparente per il sistema, cioè fondamentalmente non è possibile fare query sui valori, ma solo sulle chiavi. Il modello chiave-valore è il modello logico più semplice; ciò non significa che sia di facile implementazione, tutt'altro: può presentare problematiche molto complesse.

Sebbene sia possibile estendere una tale API per permettere transazioni che coinvolgono più di una chiave, ciò sarebbe controproducente in un ambiente distribuito, dove l'utilità di avere coppie chiavi-valore non legate fra loro permette di scalare orizzontalmente in modo molto semplice: è sufficiente, infatti, suddividere le coppie tra i server. Anche le ricerche nel database possono essere fatte in parallelo, ad esempio con funzioni Map e Reduce.

Sebbene a prima vista il modello sembra semplicistico se paragonato a quello relazionale, effettivamente però in molti ambiti possiamo utilizzare strutture di tipo chiave-valore, ad esempio in un sito di e-commerce c'è la profilatura utente, il cestino, lo storico degli ordini etc. Un esempio di questo tipo di database è Cassandra, un DBMS open source, utilizzato e sviluppato per Facebook.

Il modello orientato alle colonne

In un database orientato alle colonne vengono utilizzate le tabelle come modello di dati, ma non supporta l'associazione tra le tabelle (attraverso l'associazione tra una chiave primaria della tabella principale e le foreign key delle altre tabelle). In questa tipologia di database, i dati sono memorizzati separatamente per ogni colonna, dove ogni colonna è un indice nel database, piuttosto che in righe come nei database relazionali. Nell'uso pratico, nel mondo dei DBMS, ci sono poche differenze tra il salvare i dati in riga, piuttosto che in colonna. Entrambi i modelli posso usare i tradizionali linguaggi per effettuare le query come SQL, e sia il modello a righe sia quello a colonne possono lavorare in un sistema che gestisce i dati con le comuni operazioni di estrazione, trasformazione e caricamenti (ETL). Tuttavia, archiviando i dati in colonne piuttosto che in righe, il database può accedere con maggiore precisione ai dati di cui ha bisogno per rispondere ad una query, piuttosto che scansionare e scartare i dati indesiderati contenuti nelle righe.

Il modello orientato ai documenti

Questo modello è simile al modello chiave-valore, tranne che per il fatto che il valore non è trasparente per il database ma è in un formato che il sistema può interpretare e interrogare. I formati più usati sono XML e JSON. JSON,

Software Engineering Research & Practices

essendo semplicemente un oggetto JavaScript serializzato, può essere ovviamente molto utile in ambiente web. Generalmente i DBMS document-oriented utilizzano una o più proprietà degli oggetti per indicizzarli ed è possibile effettuare delle interrogazioni basate sulle proprietà dell'oggetto. Ad esempio, se consideriamo una collezione di documenti JSON che rappresentano gli utenti di un forum una query potrebbe essere "tutti gli utenti che hanno 'Java' tra i tag dichiarati", e potrebbe essere espressa tramite un oggetto JSON. Esempi di DBMS orientati ai documenti sono MongoDB, CouchDB, JackRabbit e TerraStore.

Il modello a Tabellare

Sebbene il nome possa ricordare le tabelle dei RDBMS, qui la grande differenza è che si parla di dati schemaless. Questo modello può essere descritto come un document-oriented con valori multidimensionali e senza schema. Chiariamo il concetto con un esempio: immaginiamo un social network in cui gli utenti possono aggiungere contenuti in varie forme (testo, link, foto, ...) e provenienti da vari canali (web, mobile, SMS, ...). Volendo impiegare un database di questo tipo per lo storage dei dati, avremmo una struttura in cui ogni contenuto ha una chiave unica e un documento. Il documento a sua volta è composto da un valore oppure una serie di tuple nome-valore e così via.

Le limitazioni, rispetto al modello relazionale, sono evidenti: a parte la chiave primaria, non è facile effettuare interrogazioni; il modello è abbastanza complesso; non è possibile garantire la consistenza con strumenti analoghi alle foreign key. Però il grande vantaggio è che è possibile immagazzinare enormi quantità di dati sparsi in un gran numero di server, proprio come viene fatto con BigTable, il database usato da Google.

1.17.2.2 Teorema CAP

Il teorema CAP, noto anche come teorema Brewer (dal nome del professore che nel 2000 ha proposto il teorema), afferma che per un sistema informatico distribuito è impossibile fornire tutte e tre le seguenti garanzie:

- Coerenza: tutti i nodi appartenenti al sistema hanno accesso agli stessi dati nello stesso momento.
- Disponibilità: garanzia che ogni richiesta riceva una risposta
- Tolleranza di partizione: il sistema continua a funzionare correttamente anche in presenza di una serie fallimenti

Software Engineering Research & Practices

nell'infrastruttura, fino a quando anche una minima parte della rete è ancora in piedi.

Secondo questo teorema, inoltre, un sistema distribuito non è in grado di soddisfare tutte e tre le garanzie, ma al massimo due contemporaneamente.

In accordo con il teorema, si può effettuare una preliminare classificazione dei DBMS NoSQL, secondo i tre possibili accoppiamenti delle precedenti proprietà che è possibile garantire contemporaneamente.

- Coerenza/Disponibilità (CA): gli interessi del database non sono focalizzati sulla tolleranza di partizione, e si fa uso della replicazione per assicurare che i dati siano coerenti tra i vari nodi e sempre disponibili. I DBMS incentrati sul dualismo CA sono i tradizionali database relazionali, Vertica (orientato alle colonne), Aster Data (relazionale), Greenplum (Relazionale), ecc.
- Coerenza/Tolleranza di partizione (CP): chi invece mette al centro dei propri interessi le capacità del DBMS di essere coerente, salvando i dati in tutti i nodi distribuiti, e tollerante, non garantisce altrettanto bene la disponibilità dei dati. Il mercato offre molte soluzioni basate sulle proprietà CP, quali BigTable (orientato alle colonne), Hypertable (orientato alle colonne), HBase (orientato alle colonne), MongoDB (orientato ai documenti), Terrastore (orientato ai documenti), Redis (chiave-valore), ecc.
- Disponibilità/Tolleranza di partizione (AP): i sistemi che garantiscono disponibilità e tolleranza di partizione sono Voldemort (chiave-valore), Tokyo Cabinet (chiave-valore), CouchDB (orientato ai documenti), Cassandra (ibrido orientato alle colonne/righe), ecc.

Tutto quanto detto, per rispondere alle esigenze progettuali e considerando l'orientamento del mercato, i punti di forza delle due tipologie di database, delle esigenze del progetto e della volontà di trarre vantaggio dell'utilizzo di dispositivi IoT è necessario adottare l'unico approccio ibrido che consente di sfruttare i punti di forza di DB relazionali e dei DB non relazionali mantenendo di fatto il dualismo che si evince dai vari sondaggi rilevabili sul

Software Engineering Research & Practices

web. In particolare, si utilizzeranno il DBMS MySQL/MariaDB per la memorizzazione di tutti i dati strutturati utili a memorizzare tutti i dati non IoT, mentre si utilizzerà MongoDB per memorizzare tutti i dati provenienti dai dispositivi IoT. Per completezza, di seguito di riporta una sintesi delle peculiarità di MongoDB utile far emergere i punti di forza che potranno diventare valore aggiunto per il progetto i-mole, non si descrivono i punti di forza di MySQL in quanto ormai è da considerarsi uno standard di fatto per DB opensource e che di conseguenza non ha bisogno di presentazioni o approfondimenti.

1.17.3 MongoDB

MongoDB (da "humongous", enorme) è un Database Management System potente, flessibile e scalabile. Combina le grandi capacità di scalabilità, con molte delle più pratiche caratteristiche dei più tradizionali database relazionali come gli indici secondari, range query e ordinamento. Inoltre, MongoDB ha tantissime funzionalità utili come il supporto integrato per l'aggregazione in stile MapReduce e gli indici geo-spaziali (in supporto alle query geo-spaziali).

Il DBMS è nato con l'idea di semplicità in mente: è basato su un modello dati developer-friendly, con opzioni di configurazione intuitive per l'amministratore ed API ideate per ottenere la massima chiarezza e praticità (sia per i driver che per la shell).

MongoDB è un database orientato ai documenti, e non di tipo relazionale. La ragione del distacco dal più tradizionale database relazionale è per semplificare e favorire le sue capacità di scalabilità, tuttavia esistono ulteriori vantaggi. L'idea di base è quella di sostituire il concetto di "riga" con un modello più flessibile: il "documento". Questo modello permette di rappresentare complesse relazioni gerarchiche con un singolo record. Questo modo di pensare la realtà è molto vicino al modo in cui gli sviluppatori pensano ai propri dati nei moderni linguaggi orientati agli oggetti. MongoDB è anche schema-free: le chiavi di un documento non sono predefinite o fissate in nessun modo. Senza uno schema da rispettare e da modificare, le massicce migrazioni dei dati solitamente non sono più necessarie. Le chiavi nuove o mancanti possono essere gestite a livello applicazione, invece di forzare tutti i dati ad avere la stessa forma. Questo

Software Engineering Research & Practices

approccio offre agli sviluppatori molta più flessibilità nel modo di lavorare con i modelli di dati in continua evoluzione.

1.17.3.1 Scalabilità

La quantità di dati generate dalle applicazioni cresce ad un ritmo incredibile. I progressi nella tecnologia dei sensori, l'aumento della larghezza di banda disponibile e la crescente popolarità dei dispositivi IoT connessi ad internet hanno creato un ambiente in cui anche le applicazioni di modeste dimensioni hanno la necessità di archiviare molti più dati di quanti molti tradizionali database possano gestire. Terabyte di dati, quelli che un tempo erano considerati una quantità inaudita, adesso sono all'ordine del giorno.

Per scalabilità si intende la proprietà di un sistema di crescere o decrescere in base alle esigenze e alle necessità di chi lo utilizza. Un sistema può essere scalato in due modi, orizzontalmente e verticalmente. Per scalabilità orizzontale si intende la possibilità di intervenire sulla propria struttura aggiungendo o eliminando server, mentre per scalabilità verticale si intende la possibilità di intervenire su di un singolo server incrementando o riducendo le prestazioni dei singoli componenti (RAM, CPU, SSD, ecc.).

Il ridimensionamento verticale è spesso il percorso più facile, ma presenta alcuni svantaggi: le macchine di grandi dimensioni sono spesso più costose e alla fine si va incontro ad un inevitabile limite tecnologico. Scalare orizzontalmente invece dà invece la possibilità di una continua (ipoteticamente infinita) crescita ma a costi più contenuti: aggiungere ulteriore spazio di archiviazione o aumentare le prestazioni del proprio sistema si traduce nell'acquistare un ulteriore server ed aggiungerlo al proprio cluster.

MongoDB è stato progettato sin dall'inizio con l'idea della scalabilità orizzontale. Il suo modello di dati orientato ai documenti consente di distribuire automaticamente i dati su più server. Ciò permette agli sviluppatori di concentrarsi sulla progettazione dell'applicazione e non nella sua scalabilità: quando hanno bisogno di maggiori capacità possono aggiungere nuove macchine al cluster e lasciare che il database capisca come organizzare il tutto.

Software Engineering Research & Practices

1.17.3.2 Prestazioni

Uno degli obiettivi più importanti da raggiungere secondo gli ideatori di MongoDB, e che ha influenzato molte decisioni di progetto, sono le prestazioni del DBMS.

MongoDB utilizza, come principale modalità di comunicazione con il server, un protocollo binario (al contrario dei più comunemente usati protocolli a sovraccarico maggiore come HTTP/REST); calibra dinamicamente una costante di riempimento (padding factor) usata per determinare quanto spazio extra MongoDB dovrebbe allocare per ogni documento, sacrificando in questo modo spazio extra su disco per performance migliori; con il motore di archiviazione predefinito utilizza file mappati in memoria, il che delega la responsabilità della gestione della memoria al sistema operativo; inoltre, fa uso di un ottimizzatore di query dinamico che, per ogni query, genera un "piano delle query" che abbina la query all'indice che restituirà i risultati nel modo più efficiente possibile. L'ottimizzatore riutilizzerà questo piano ogni volta che si esegue la query. Se una collezione cambia significativamente, l'ottimizzatore creerà un nuovo piano di query per mantenere alte le performance.

Sebbene MongoDB tenti di mantenere molte delle funzionalità dei tradizionali sistemi relazionali, non fa tutto ciò che un DBMS relazionale è programmato per fare: ad esempio, quando possibile, il database scarica la responsabilità dell'elaborazione e della logica sul lato client (qui gestita dai driver o dal codice scritto dall'utente). Solo sacrificando in parte questa autonomia, pre-allocando spazio extra su disco o generando dinamicamente piani di query per ogni collezione MongoDB può mantenere prestazioni così elevate.

1.17.3.3 Amministrazione

Un altro dei punti forti del DBMS, è la semplicità nell'amministrazione e nella manutenzione. Dopo l'avvio del server del database da parte dell'amministratore di sistema, il DBMS necessità di pochi o addirittura nessun ulteriore intervento. MongoDB fornisce due tipologie di repliche: Master-Slave e Replica Set. Il primo metodo ha il vantaggio di supportare un qualsiasi numero di nodi secondari, al costo di un meccanismo di failover manuale. Se il nodo primario viene meno, un amministratore di sistema deve scegliere un nodo secondario e configurarlo come nodo primario (e per questa ragione è passato in secondo piano rispetto alla successiva tipologia

Software Engineering Research & Practices

di replicazione). Il secondo metodo invece, Replica Set, è costituito da un gruppo di nodi in cui, tramite votazione interna, un nodo è eletto primario, mentre tutti gli altri sono definiti secondari. Il nodo primario gestisce quindi tutte le operazioni di lettura e scrittura ed effettua, in modalità asincrona, la replica delle operazioni sui nodi secondari. In caso di non raggiungibilità del nodo primario un nuovo nodo secondario viene eletto al posto del non raggiungibile, facendo sì che il sistema sia sempre disponibile in lettura e in scrittura. Quando il precedente nodo primario ritorno raggiungibile viene inserito nel cluster di nodi come secondario. Il meccanismo di replicazione replica set ha il vantaggio di rendere sempre disponibili le operazioni dei client in lettura e in scrittura, fintanto che ci sarà un nodo attivo. Inoltre, per aumentare ulteriormente le prestazioni in lettura e non sovraccaricare ulteriormente il server primario, è possibile utilizzare un nodo secondario per leggere i dati. Tuttavia, questo nodo deve essere scelto manualmente in quanto, a causa delle replicazioni in modalità asincrona, si deve fare in modo che questo nodo sia aggiornato a tutti i dati inclusi nel nodo principale.

Software Engineering Research & Practices

Software Engineering Research & Practices

BLOCKCHAIN

Una delle capacità della soluzione digitale che la soluzione i-mole deve mettere a disposizione dei suoi utilizzatori è quella di certificare transazioni o contratti tra due soggetti.

Nel mondo reale, per la gestione delle transazioni, ad esempio, in denaro tra soggetti ci sono diversi strumenti che rendono verificabile e tracciabile una transazione. Nel mondo virtuale, uno degli strumenti per verificare e certificare una transazione tra due soggetti è l'insieme delle tecnologie note come blockchain.

Relativamente ai contratti, invece, sempre nel mondo reale la correttezza e la validità di un contratto stipulato tra due soggetti normalmente viene garantita da una terza persona esterna al rapporto e al contratto, ad esempio, un notaio che si assicura anche che il contratto non possa essere alterato in alcun modo. Una volta stipulato un contratto, sono le parti stipulanti che devono assicurarsi che ognuna delle controparti rispetti quanto definito nel contratto ed in caso contrario devono avviare delle iniziative per "costringere" la controparte a tenere fede al contratto. Il corrispettivo dei contratti nel mondo virtuale sono gli smart contract basati su blockchain che non solo certificano gli accordi tra le parti, rendono immodificabile il contratto stipulato, e si occupano, anche, di verificare, in modo automatico e senza richiedere l'intervento delle parti interessate, che quanto previsto dal contratto sia rispettato.

Per ultimo sempre rispetto al progetto, i-mole, la blockchain può essere utilizzata come strumento per certificare il contenuto e la data di un documento con un meccanismo noto come notarizzazione che prevede l'inserimento di un hash nel documento. Questa caratteristica è utilizzabile in i-mole per molti scopi tra cui, a titolo di esempio, basti pensare alla notarizzazione dei movimenti di un camion che trasporta merce da un fornitore a vari clienti.

In estrema sintesi, le caratteristiche principali della tecnologia blockchain a cui i-mole è interessato sono:

- immutabilità del registro ovvero dei dati gestiti
- tracciabilità delle transazioni
- alto livello di sicurezza basato su tecniche crittografiche
- applicazione per la gestione degli smart contract.

Software Engineering Research & Practices

Venendo alla declinazione operativa della tecnologia blockchain utilizzabile per il progetto si è provveduto a valutare le soluzioni blockchain utilizzabili per rispondere alle esigenze di progetto valutando le seguenti caratteristiche:

1. Tipologia di software opensource
2. Disponibilità degli strumenti di creazione ed esecuzione degli smart contract
3. Livello di integrabilità della soluzione
4. Laicità rispetto ad un particolare dominio applicativo

Rispetto a questi quattro indicatori sono state valutate le soluzioni presenti disponibili ed in particolare le sole soluzioni opensource ottenendo il seguente insieme di soluzioni.

- Ethereum – è una soluzione creata principalmente per la creazione di smart contract tramite un linguaggio di programmazione.
- Openchain – è una piattaforma pensata per realizzare una software supply chain in ambito open source consenta di stabilire i requisiti per ottenere una gestione efficace del software libero / open source.
- AEternity – è una piattaforma pubblica di calcolo distribuito che si basa su soluzioni esistenti come Bitcoin ed Ethereum.
- Counterparty – è una soluzione creata in ambito finanziario utilizzabile per la creazione di applicazioni peer-to-peer basate sul bitcoin.

Successivamente sono state scartate le soluzioni specialistiche e non adattabili a contesti differenti AEternity, Counterparty.

Infine, valutando la predisposizione delle soluzioni rimaste rispetto alla capacità di produrre e gestire gli smart contract la scelta è ricaduta su Ethereum poiché è la soluzione costruita proprio allo scopo di creare smart contract. Inoltre, anche se non è risultata essere una discriminante Ethereum è la soluzione opensource che più è stata utilizzata come base per creare altre soluzioni basti pensare che anche in questo elenco ristretto è riportata la soluzione AEternity che è stata prodotta proprio utilizzando Ethereum e questo ci fa capire che adattare, riutilizzare e plasmare la piattaforma agli interessi del progetto è una attività fattibile.

Software Engineering Research & Practices

1.18 Smart contract su Ethereum

Fin dalla presentazione del progetto uno dei punti focali dell'idea progettuale trova applicazione nell'utilizzo degli smart contract che nel caso della blockchain Ethereum sono supportati e fortemente integrati.

Nella pratica sebbene gli smart contract altro non siano che applicazioni software che funziono su blockchain, a causa delle peculiarità dell'ambiente, la realizzazione dei programmi smart contract presenta delle differenze rispetto alla programmazione tradizionale e richiede pertanto l'adozione di pattern e di strumenti di sviluppo specifici.

I contratti vengono preventivamente compilati in bytecode, un'intersezione tra BitcoinScript ed Assembler addizionato con la funzionalità di invio di messaggi ricorsiva di Lisp, ed eseguiti dall'Ethereum Virtual Machine (EVM) con metodologia LIFO. Ogni istruzione inizia con un opcode a byte singolo e gli argomenti, se presenti, occupano byte successivi (non allineati), con valori indicati in ordine big-endian.

Per questo motivo è quindi, necessario, una volta definita la blockchain da utilizzarsi identificare gli strumenti che la stessa mette a disposizione per programmarla e quindi declinarla alle esigenze del progetto e per produrre le applicazioni smart contract, in termini operativi di seguito si declina lo stack applicativo che si utilizzerà.

1.18.1 Solidity

Gli smart contract Ethereum, assimilabili alle classi, possono essere scritti in tre differenti linguaggi di programmazione: Solidity, Viper e LLL.

Solidity, linguaggio Object-Oriented simile a Javascript, è il linguaggio ufficiale ed il più diffuso.

In ogni contratto viene specificata la versione del compilatore per abilitare le relative funzionalità e controlli.

1.18.2 Truffle framework

Gli strumenti assistivi per il testing ed il deployment risultano indispensabili in quanto l'interazione con le blockchain live è regolata dall'immutabilità dei dati, dai tempi di attesa per il mining dei blocchi e dalle commissioni per singola transazione. Truffle Framework è un insieme di librerie per la

Software Engineering Research & Practices

compilazione, linking, deployment e gestione dei sorgenti per i contratti Ethereum.

1.18.3 Web3js

Web3.js è una collezione di librerie che permettono di interagire con un nodo Ethereum, sia locale che remoto, attraverso l'uso di HTTP o IPC.

Web3 comunica con la Blockchain Ethereum attraverso JSON RPC, acronimo di "Remote Procedure Call", che permette a Web3 di effettuare richieste individuali ai nodi Ethereum per leggere e scrivere sulla rete, similmente a quanto fatto da jQuery con le JSON API per interagire con i web server.

1.18.4 Ganache-CLI

Ganache-CLI, parte della suite Truffle tra gli strumenti di sviluppo per Ethereum, è una versione a riga di comando di una Blockchain locale che utilizza EthereumJS per simulare il comportamento di un client, rendendo lo sviluppo di applicazioni decentralizzate molto rapido e (più) sicuro. Include, inoltre, tutte le funzioni RPC più diffuse e le sue funzionalità principali - come il trigger di eventi - possono essere eseguite deterministicamente.

In fase di realizzazione, una volta avviata la Blockchain nell'ambiente di produzione, Ganache-CLI fornisce degli indirizzi fittizi intesi per effettuare dei test su quanto prodotto.

1.18.5 MetaMask

MetaMask è una estensione per browser che svolge le veci di un portafoglio di crittovalute ed al contempo funge da gateway per i software ospitati su blockchain.

Grazie a MetaMask è possibile interagire direttamente con le applicazioni decentralizzate e collegarsi con diverse reti, rimuovendo anche la necessità di ospitare un nodo completo sulla postazione di sviluppo.

1.19 Conclusione

È bene evidenziare che nella analisi comparativa sono volutamente stati trascurati gli aspetti prestazionali perché non influenti nel contesto i-mole. Ovviamente, nel momento in cui si voglia valutare l'utilizzo della blcokchain

Software Engineering Research & Practices

in un contesto fortemente collaborativo e che necessita di un alto livello di prestazione e che non prevede l'utilizzo del meccanismo di consenso alla base di una qualsiasi blockchain la scelta effettuata dovrebbe essere rivista in favore di soluzioni innovative e differenti. A tale proposito nell'ambito delle Distributed Ledger Technology per superare i problemi di sistemi delle blockchain che utilizzano la PoW (Proof of Work) e i problemi di prestazioni che queste sono in grado di garantire pari a circa 7 transazioni sono state create e sono utilizzate sistemi quali HashGraph che forte delle sue prestazioni: più di 250.000 transazioni al secondo (pre-sharding) e 1gb per la memorizzazione delle transazioni al contrario dei 60GB per ethereum e affini si propone come il futuro delle blockchain. Infine, un altro elemento di debolezza della attuale tecnologia HashGraph è che non supporta nativamente gli smart contract.

Software Engineering Research & Practices

BIG DATA

La propulsione al sapere propria della natura umana, con il passare del tempo, ha fatto acquisire all'uomo enormi capacità sia di pensiero che di abilità costruttiva, difatti la storia dell'umanità è costellata di numerose invenzioni a partire da piccoli congegni, ad arrivare talvolta anche a congegni di grande complessità e precisione, alcune delle quali ha dato vita a nuovi ambiti.

Nell'anno 1623, Wilhelm Schickard così come testimoniato da alcune lettere inviate a Giovanni Keplero ha costruito il primo prototipo di una calcolatrice, chiamato Orologio calcolatore. Circa 20 anni dopo, nel 1645, il matematico francese Blaise Pascal, per aiutare il padre funzionario delle tasse, ha inventato e costruito il dispositivo di calcolo, basato su un sistema di ruote dentate, poi conosciuto come Pascalina.

Da allora ad oggi è stato un susseguirsi di invenzioni che, come per la pascalina, sono servite e servono tuttora per migliorare lo stile di vita dell'uomo dalla "macchina di Turing" (1936) al primo computer digitale automatico "Aiken–IBM Automatic Sequence Controlled Calculator Mark I" (1944) basato su valvole. Questi sono stati è stati gli eventi che hanno dato il via alla nascita dei moderni elaboratori e di quello che oggi conosciamo con il nome di Informatica, intendendo il trattamento automatico dell'informazione mediante calcolatore.

Da queste prime scoperte il mondo dell'informatica e della comunicazione digitale si è evoluto tanto da farci giungere a nuove frontiere come "Internet of things", l'Internet delle cose, che è a tutti gli effetti una nuova rivoluzione della rete. Gli oggetti si rendono riconoscibili e acquisiscono la capacità di comunicare dati, ovvero una rappresentazione elettronica dell'informazione, su sé stessi e di accedere ad informazioni aggregate da parte di altri. Le sveglie suonano prima in caso di traffico, le scarpe da ginnastica trasmettono tempi, velocità e distanza per gareggiare in tempo reale con persone dall'altra parte del globo e così via. In altre parole, grazie al collegamento alla rete tutti gli oggetti possono esercitare un ruolo attivo.

"Internet delle cose" e la "società dell'informazione" testimoniano come nella società odierna i dati e la loro manipolazione rivestono un ruolo centrale. Con il termine "società dell'informazione" si connota una realtà socioeconomica in cui gran parte della produzione di servizi, specialmente quelli in cui si manipolano dati, è basata sul valore economico della

Software Engineering Research & Practices

informazione come risorsa strategica. Una società di questo tipo è fortemente dipendente dall'uso delle tecnologie informatiche e telecomunicazione. Questa concezione vede un ruolo fondamentale nella disponibilità di infrastrutture tecnologiche che semplificheranno il lavoro, ne permetteranno lo svolgimento a distanza e ridurranno i costi di produzione dei servizi uomo centrici. Tutto ciò cambierà il lavoro e la vita degli individui, questi ultimi grazie alla disponibilità abbondante di informazioni e tecnologie potranno fare scelte di vita meno rischiose, trovare più facilmente ciò che desiderano e così via.

Veniamo al paradosso creatosi in seguito alla rapida evoluzione della informatica e delle tecnologie ad essa associata di cui sopra, ovvero: nella società attuale il reperimento dei dati e delle informazioni desiderate richiede quantità di tempo e competenze specifiche tali che sovente si abbandonano le ricerche ed i lavori ad esse correlati. Per questo motivo è importante, in un progetto come i-mole porre da subito particolare attenzione agli strumenti che è necessario utilizzare per la gestione di grandi moli di dati (gestione Big Data).

Operativamente, nel settore dell'ICT, il termine Big Data è stato utilizzato per descrivere le operazioni di immagazzinamento, gestione e analisi di grandi quantità di dati da software e da hardware "tradizionali" in un tempo tollerabile[3]. Anche se una definizione rigorosa ed univoca del fenomeno Big Data non è ancora stata ufficializzata, lo studio di Douglas Laney in cui si descrive il modello di crescita dei big data è definito come tridimensionale o modello delle "3V" è il punto di riferimento (Lo studio 3D Data Management: Controlling Data Volume, Velocity and Variety è stato eseguito tra il 2001 e il 2002).

Secondo questa analisi:

- Volume: si riferisce alla quantità di dati, strutturati o non, che sono generati da sorgenti eterogenee;
- Varietà: si riferisce alla differente tipologia di dati che vengono generati, collezionati ed utilizzati;
- Velocità: si riferisce alla celerità con cui i nuovi dati vengono generati.

[3] The McKinsey Global Institute, 2012.

Software Engineering Research & Practices

Venendo al contesto i-mole, quindi, l'utilizzo dell'insieme delle tecnologie pensate per gestire Big Data è stato preso in esame per aumentare l'efficienza dei processi produttivi, migliorare la capacità decisionale degli amministratori della soluzione, prevedere più accuratamente le tendenze di mercato e indirizzare in modo molto più mirato le diverse proposte commerciali".

Ritornando allo stack tecnologico oggetto di questo rapporto tecnico per la gestione dei Big Data i livelli operativi sono:

- Data Management e Integration. È compito di questo strato quello di raccogliere dati da fonti e formati diversi, per renderli in qualche modo omogenei ed essere così conservati ed effettuare delle attività di analisi preliminare sui dati da integrare.
- Data Analytics. È compito di questo livello effettuare le analisi vere e proprie sui dati esplorando le soluzioni di analisi a diversi livelli. Una delle analisi sfrutta la tecnica del clustering che permette di ridurre la complessità delle informazioni. Si supponga di avere dati relativi a delle squadre di calcio, il clustering viene applicato per creare sottoinsiemi che raccolgono i dati simili tra loro, ad esempio, ogni gruppo contiene i soli dati di una squadra. Successivamente si passa alla numerazione di questi insieme suddivisi per squadra ottenendo così l'output finale ovvero per squadra A 12 risultati, squadra B 2 risultati e così via. Le operazioni di clustering sono molto spesso effettuata sfruttando le potenzialità dell'Edge computing, che permette di pre-processare i dati al di fuori del database. I sistemi di analisi dei dati nella eccezione comune sono chiamati sistemi di Data Mining, possono essere quindi classificate in due categorie: metodi descrittivi e metodi previsionali. I primi si occupano di identificare proprietà e caratteristiche nei dati, mentre i secondi sono in grado di effettuare delle previsioni.

Dal punto di vista degli strumenti per l'integrazione e la raccolta dei dati la necessità è quella di disporre di un framework per la archiviazione e la elaborazione di grandi quantità di dati. Per effettuare tale scelta gli indicatori utilizzati sono:

1. Tipologia di soluzione. La soluzione deve essere opensource in quanto deve poter essere adattata ed evoluta per rispondere alle esigenze progettuali

Software Engineering Research & Practices

2. Completezza della soluzione. La soluzione deve disporre di uno stack tecnologico che consente di coprire trasversalmente diversi domini applicativi.
3. Validazione della adottabilità. Prova empirica della utilizzabilità della soluzione per creare una applicazione di gestione di grandi quantità di dati.

Utilizzando questi parametri, pur se ridotti, si è potuto identificare precisamente una sola soluzione ovvero: Apache Hadoop.

Apache Hadoop, è il motore Open Source divenuto ormai uno standard nella gestione di dati da molti big del mondo Internet, a cominciare da Yahoo, Facebook e Google. Hadoop serve per gestire applicazioni di Big Data e si presenta nella forma di un set di software Open Source distribuito sotto la licenza Apache Hadoop Open Source.

Utilizzando Hadoop sono stati realizzate diverse piattaforme di gestione dei Big Data di cui le più note sono:

- **Hortonworks Data Platform HDP** è realizzata per permettere la distribuzione a livello enterprise di soluzioni Apache Hadoop basate su archietetture centralizzate di tipo Yarn.
- **Cloudera Data Science Workbench**, è una soluzione di Machine learning che unisce analytics e team working.

1.19.1 Apache Hadoop

È un "framework di strumenti open source che fornisce elaborazione distribuita altamente affidabile e scalabile di dataset di grandi dimensioni, utilizzando modelli di programmazione semplici. Hadoop è basato su un cluster di computer di largo consumo; pertanto, fornisce una soluzione economicamente conveniente per la memorizzazione e l'elaborazione di enormi quantità di dati strutturati, semi-strutturati e non strutturati senza alcun requisito di formato. Questo rende Hadoop ideale per la creazione di data lake a supporto delle iniziative di big data analytics".

Software Engineering Research & Practices

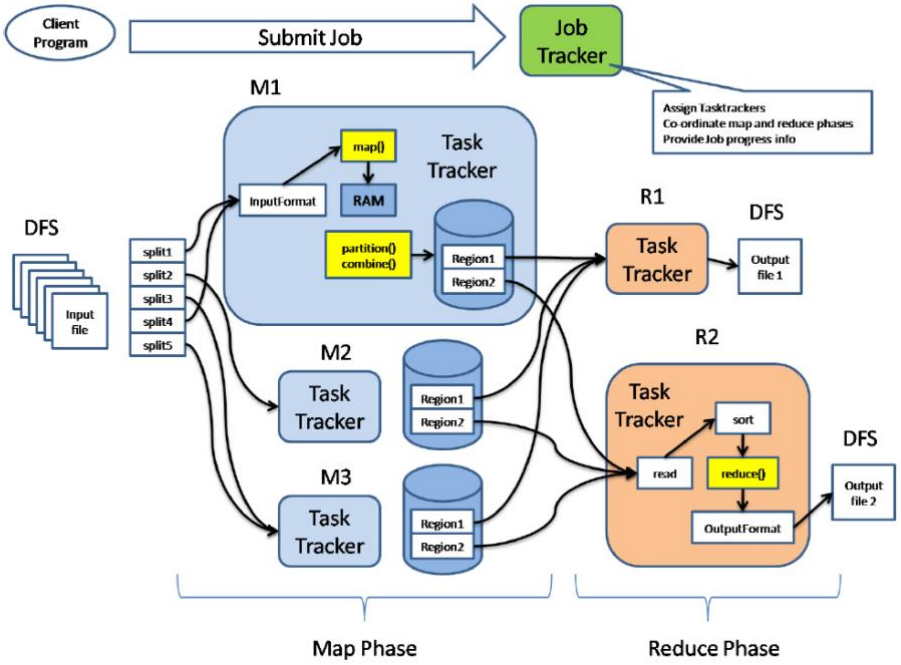


Figura 8 Funzionamento di Apache Hadoop, basata su MapReduce e HDFS

Apache Hadoop nasce per rispondere principalmente a 3 esigenze:

- Immagazzinare grandi dataset: i tradizionali RDBMS non sono capaci di immagazzinare grandi quantità di dati per questioni legati ai costi dell'hardware e del software;
- Gestire diversi tipi di dati: gli RDBMS manipolano dati strutturati, ma in realtà si ha a che fare anche con dati semi strutturati e non strutturati;
- I dati vengono generati sempre più rapidamente: la mole di dati generata giornalmente è in una fase di crescita esponenziale, per questo è necessario un sistema in grado di processare dati in tempo reale in pochi secondi. Gli RDBMS non sono adatti a rispondere a questa esigenza.

Hadoop fu consegnato nel 2008 da Yahoo ad Apache Software Foundation, e da allora sono state sviluppate altre versioni. Ad oggi ci sono diverse distribuzioni di Hadoop, ovvero Cloudera, IBM BigInsight, MapR e Hortonworks.

L'architettura di Apache Hadoop è formata da 3 componenti principali:

- HDFS

Software Engineering Research & Practices

- MapReduce
- YARN

HDFS, acronimo che sta per Hadoop Distributed File System, ovvero dei file system distribuiti, portabili e scalabili scritti in Java che forniscono memoria distribuita per il framework Hadoop. HDFS dispone di una tecnologia master-slave. Per master si intende una macchina di fascia alta e per slave macchine economiche.

I dati vengono divisi in un certo numero di blocchi. Hadoop archivia questi blocchi in modo distribuito nel cluster di nodi slave. Sul master, invece, vengono archiviati i metadati. HDFS richiede l'esecuzione di due daemon che sono:

- NameNode: eseguito sulla macchina master, svolge le seguenti funzioni:
  - È responsabile della manutenzione, del monitoraggio e della gestione di DataNodes;
  - Registra i metadati dei file come posizione dei blocchi, dimensione del file, permessi, ecc.
  - Namenode acquisisce tutte le modifiche ai metadati come la cancellazione, la creazione e la ridenominazione del file nei log di modifica.
- DataNode: eseguito sulle macchine slave, svolge le seguenti funzioni:
  - Memorizza i dati effettivi
  - Serve la richiesta di lettura/scrittura da parte dell'utente.
  - DataNode svolge il lavoro di base per creare, replicare ed eliminare i blocchi con il comando di NameNode.

In generale un cluster in Hadoop possiede uno o più Namenode, su cui risiedono i metadati dei file, e un insieme di Datanode, su cui risiedono i file dell'HDFS in blocchi di dimensione fissa. Questo permette di garantire la scalabilità orizzontale del sistema, infatti se le dimensioni di un file superano quelle della memoria che si ha a disposizione, Hadoop permette di archiviarlo distribuendolo su più server o nodi.

MapReduce rappresenta il livello dove vengono processati i dati, e questa operazione è divisa in due fasi:

Software Engineering Research & Practices

- Fase di Map: in questa fase i dati vengono convertiti in coppie chiave-valore;
- Fase di Reduce: in questa fase, viene preso l'output della precedente fase di Map e viene applicata l'aggregazione in base alla chiave delle coppie chiave-valore.

MapReduce lavora nel seguente modo:

- Il client specifica il file per l'input alla funzione Map e lo divide in tuple;
- La funzione Map definisce chiave e valore del file di input. L'output della funzione map è questa coppia chiave-valore.
- Il framework MapReduce ordina la coppia chiave-valore dalla funzione Map.
- Il framework unisce le tuple con la stessa chiave
- Reduce prende queste coppie chiave-valore come input e applica le funzioni aggregate sulla coppia chiave-valore.
- L'output viene scritto in HDFS.

Hadoop permette di gestire le funzioni di processing su questi dati distribuiti attraverso la funzione MapReduce che ha il vantaggio di distribuire la capacità elaborativa (così come HDFS permette di distribuire la capacità di archiviazione su diversi nodi). In sostanza, con HDFS e MapReduce si può passare da un'elaborazione seriale ad una di tipo parallelo. Con Hadoop si può lavorare con migliaia di nodi e petabyte di dati, e, per quanto concerne la capacità computazionale, è ispirato alla logica dell'algoritmo MapReduce di Google.

Dalla versione 2 di Apache Hadoop, oltre ad HDFS e MapReduce, è stato aggiunto nel sistema YARN, componente che si occupa di separare la gestione delle risorse e l'elaborazione effettiva. YARN garantisce quindi una migliore gestione dei cluster, riducendo il carico di lavoro a MapReduce.

YARN è composto da due componenti:

- ResourceManager: eseguito sul nodo master e svolge i seguenti compiti:
  - Controlla le risorse disponibili per ogni slave;
  - Gestisce le risorse del cluster, assegnandole alle varie attività.
- NodeManager: eseguito sulle macchine slave e svolge i seguenti compiti:

Software Engineering Research & Practices

- Gestisce i container, ovvero una frazione delle risorse del NodeManager;
- Monitora l'utilizzo delle risorse di ciascun contenitore.

In sostanza, YARN coordina l'attività da eseguire, tenendo sotto controllo CPU, memoria, larghezza di banda della rete e lo spazio di archiviazione disponibile, offrendo vantaggi in termini di scalabilità, efficienza e flessibilità.

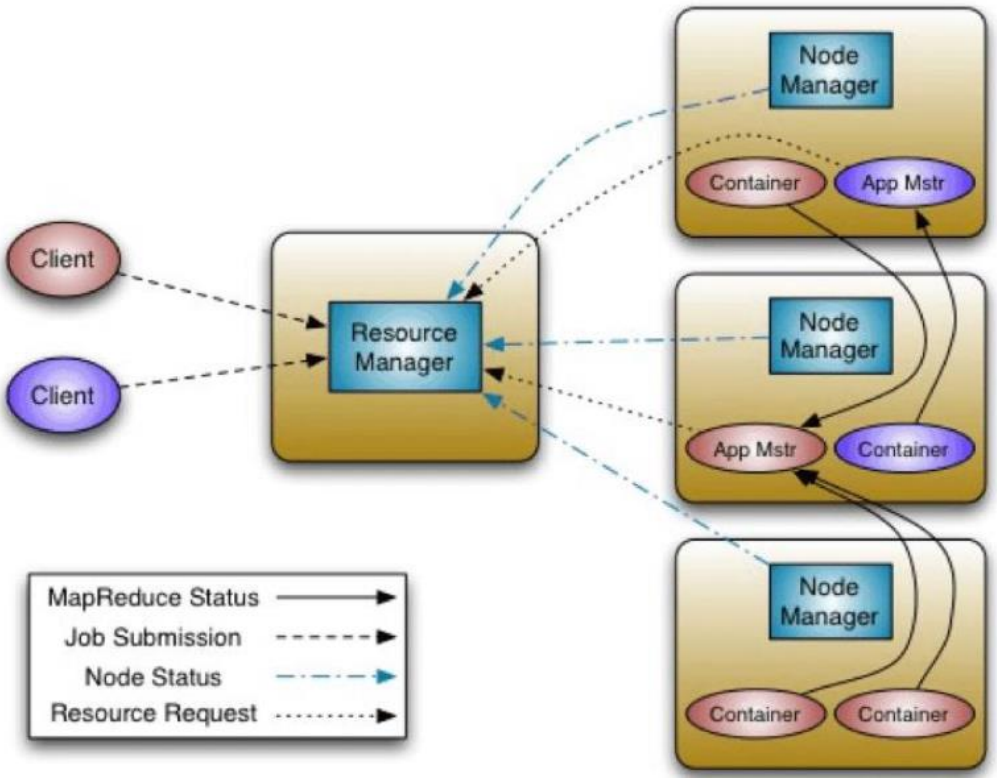


Figura 2 Architettura Apache Hadoop, basata su HDFS, MapReduce e YARN

Fanno parte dell'ecosistema di Apache Hadoop anche Apache Sqoop, Apache Flume, Apache Hive e Apache Spark.

Sqoop e Flume sono dei tool utili per gestire il flusso di dati in ambiente distribuito.

Apache Sqoop è un tool progettato per trasferire dati tra Hadoop e database strutturati, mettendo a disposizione connettori con i maggiori RDBMS e supporta il caricamento dei dati su Hive, HBase, HDFS.

Software Engineering Research & Practices

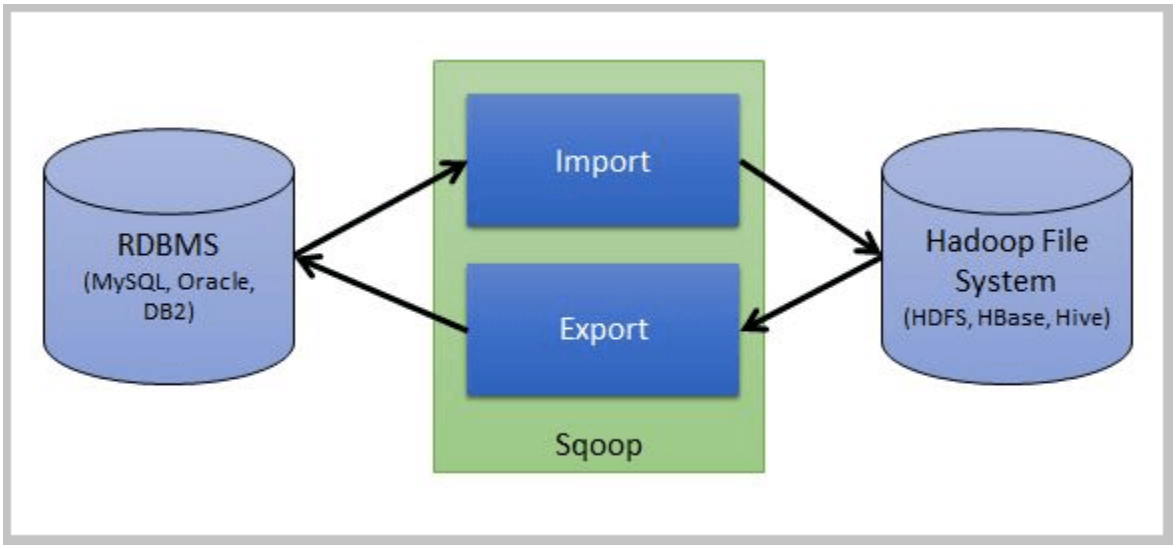


Figura 30 Funzionamento di Apache Sqoop

Apache Sqoop svolge quindi un ruolo importante nell'ecosistema Hadoop, fornendo un'interazione tra i database relazionali e HDFS.

Quindi, Apache Sqoop è uno strumento nell'ecosistema Hadoop progettato per trasferire dati tra HDFS (archiviazione Hadoop) e database relazionali come MySQL, Oracle RDB, SQLite, Postgres ecc. Apache Sqoop importa i dati da database relazionali a HDFS ed esporta i dati da HDFS a database relazionali. Trasferisce in modo efficiente dati in blocco tra Hadoop e archivi di dati esterni come i data warehouse aziendali, i database relazionali, ecc. Inoltre, Sqoop viene utilizzato per importare dati da archivi di dati esterni negli strumenti dell'ecosistema Hadoop come Hive & HBase.

Apache Flume è un tool per trasferire dati in HDFS. È stato progettato per raccogliere, aggregare e spostare grandi quantità di dati in streaming come file di registro, log file, traffico di rete, social media, messaggi di posta elettronica in maniera distribuita e affidabile su HDFS. Dispone di un'architettura semplice e flessibile basata su flussi di dati in streaming. Flume è fault-tolerant e fornisce un meccanismo di affidabilità per la tolleranza agli errori e il ripristino. I vantaggi che rendono Apache Flume una scelta migliore rispetto ad altri sono:

- Flume è scalabile, affidabile, tollerante ai guasti e personalizzabile per diverse fonti.
- Apache Flume può archiviare i dati in HBase e HDFS.
- Flume è scalabile orizzontalmente.

Software Engineering Research & Practices

- Se la velocità di lettura supera la velocità di scrittura, Flume fornisce un flusso costante di dati tra le operazioni di lettura e scrittura.
- Flume fornisce una consegna affidabile dei messaggi. Le transazioni in Flume sono basate sul canale in cui vengono mantenute due transazioni (un mittente e un destinatario) per ciascun messaggio.
- Usando Flume, possiamo inserire dati da più server in Hadoop.
- Ci offre una soluzione affidabile e distribuita che ci aiuta a raccogliere, aggregare e spostare grandi quantità di set di dati come Facebook, Twitter e siti Web di e-commerce.
- Immagazzinare dati di streaming online da varie fonti come traffico di rete, social media, messaggi e-mail, file di registro ecc. In HDFS.

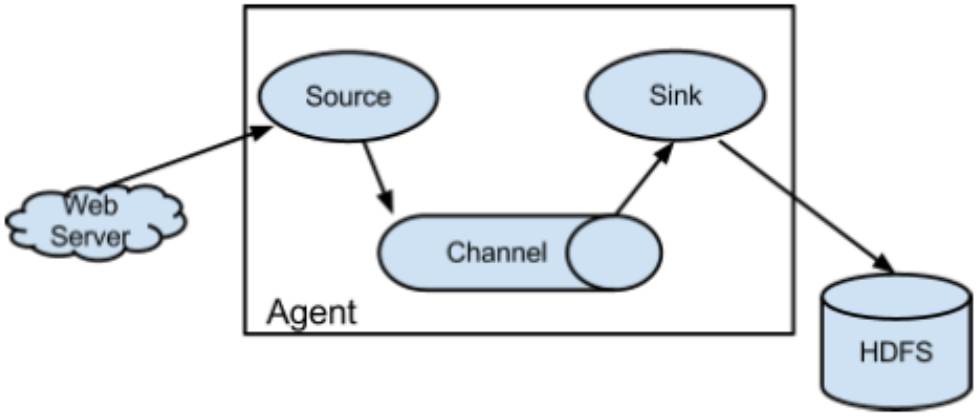


Figura 41 Funzionamento di Apache Flume

L'architettura di Flume, riportata in Figura, è basata sull'agente, composto da 3 componenti: source, sink e channel. Source accetta i dati dalla streamline in entrata e archivia i dati nel canale. Poiché la velocità di lettura è maggiore della velocità di scrittura abbiamo bisogno di un buffer per abbinare la differenza di velocità di lettura e scrittura. Fondamentalmente, il buffer funge da memoria intermedia che archivia temporaneamente i dati trasferiti e quindi impedisce la perdita di dati. Allo stesso modo, il canale funge da memoria locale o memoria temporanea tra l'origine dei dati e i dati persistenti nell'HDFS. Sink raccoglie i dati dal canale e memorizza o scrive i dati in HDFS in modo permanente.

Apache Hive è un'infrastruttura costruita su Hadoop per fornire una vista dei dati, interrogare ed analizzare, basata su linguaggio HiveQL. Apache Hive è un sistema di data warehouse basato su Hadoop e viene utilizzato per l'analisi di dati strutturati e semi-strutturati. Hive riassume la complessità

Software Engineering Research & Practices

di Hadoop MapReduce. Fondamentalmente, fornisce un meccanismo per proiettare la struttura dei dati ed eseguire query scritte in HiveQL (Hive Query Language) simili alle istruzioni SQL. Internamente, queste query o HQL vengono convertite in mappe per ridurre i lavori da parte del compilatore Hive. Pertanto, non bisogna scrivere programmi MapReduce complessi per elaborare i dati utilizzando Hadoop. Apache Hive supporta Data Definition Language (DDL), Data Manipulation Language (DML) e User Defined Functions (UDF).

Apache Spark è un framework open source per la gestione del calcolo distribuito. È stato progettato appositamente per algoritmi di apprendimento automatico e richiede un gestore di cluster e un sistema di archiviazione distribuita. Per quanto riguarda la gestione dei cluster, Spark supporta nativamente un cluster Spark, ma anche Hadoop YARN o Apache Mesos, mentre per il sistema di archiviazione distribuita, Spark può interfacciarsi con Hadoop Distributed File System (HDFS), Apache Cassandra, OpenStack Swift, Amazon S3. Spark supporta anche soluzioni pseudo-distribuite in modalità locale a scopo di test.

Venendo invece allo strato di Data Analytics gli strumenti utili ad effettuare mining sui dati che rispondo alla caratteristica di trasversalità orizzontale rispetto a un dominio applicativo sono i seguenti:

- Trifacta è una piattaforma per esplorare e preparare i dati per l'analisi; funziona con cloud e piattaforme di dati locali. Trifacta è progettato per consentire l'esplorazione, la trasformazione e la pulizia dei dati grezzi ed eterogenei in formati puliti e strutturati per l'analisi. Trifacta utilizza tecniche di machine learning.
- WEKA è un software open source costituito da una raccolta di algoritmi di apprendimento automatico a supporto del data mining. Gli algoritmi possono essere applicati direttamente ad un set di dati o richiamati dal proprio codice JAVA. WEKA contiene strumenti per la preelaborazione dei dati, la classificazione, la regressione, il clustering, le regole di associazione e la visualizzazione.
- RapidMiner è una piattaforma di data mining e predictive analytics che permette di effettuare analisi avanzate sui dati in maniera semplice e veloce, grazie a modalità di estrazione, trasformazione e visualizzazione dei dati che non richiedono particolari conoscenze di programmazione. Può essere facilmente integrato con sorgenti dati esterne, come Excel, Access, Oracle, DB2, Microsoft SQL, Sybase, Ingres, MySQL, Postgres, IBM SPSS, dBase e file di testo. RapidMiner

Software Engineering Research & Practices

dispone di un plugin, chiamato Radoop, che consente di collegarsi ad Hadoop con semplicità e permette di spostare il calcolo sul cluster, riducendo notevolmente i tempi per l'analisi.

Tra queste soluzioni solo WEKA risulta essere Open Source e quindi si è questa la soluzione che si adotterà per la realizzazione delle attività di analisi sui dati.

WORKFLOW MANAGEMENT SYSTEM

Nell'ambito del progetto i-mole la componente di workflow riveste un ruolo centrale e particolarmente critica in quanto è questa che deve consentire di disegnare e automatizzare tutti i processi che la soluzione gestirà e quindi sarà nei fatti l'orchestratore che si occuperà di far interagire le componenti del sistema e consentirà di mettere in comunicazione tra loro gli stakeholder del sistema.

Per queste ragioni, rispetto alla componente di worfkflow da utilizzare si è provveduto a identificare una serie di indicatori di dettaglio che si utilizzeranno tanto per scegliere cosa utilizzare tanto per modificare la scelta nel caso in cui risulti essere errata rispetto ai risultati intermedi ottenuti in fase di realizzazione dei dimostratori.

I sistemi posti sotto analisi nella fase iniziale sono stati 19 e tra questi si sono rilevati essere utilizzabili per la fase di realizzazione dei dimostratori solo 5 tools di seguito indicati in ordine di priorità di selezione. In altri termini per la realizzazione della soluzione si utilizzerà la piattaforma Camunda, se in questa fase o nella fase di sperimentazione questa dovesse rivelarsi non utilizzabile allora si provvederà ad utilizzare la soluzione IBM.

- Camunda
- IBM
- Jamio
- Microsoft
- Oracle

La selezione è avvenuta sulla base della presenza/assenza di funzionalità.

Si seguito si riportano gli indicatori utilizzato per la valutazione delle soluzioni di WFMS, mentre successivamente si riporteranno le tabelle di valutazione degli applicativi utilizzati riportando per leggibilità la colonna Codice indicatore e non la colonna Descrizione indicatori.

Codice Indicatore	Descrizione Indicatori
ID1	Linguaggi di orchestrazionee coreografia BPM supportati (Xlang, WSFL, WSCL, BTP, ebXML, WSCI, BPEL4WS, BPML, BPMN)
ID2	Interoperabilità
ID3	Riuso Modelli di BP
ID4	Generazione della documentazione di processo.

Software Engineering Research & Practices

	Indici della Fase di Progettazione
ID5	Strumenti per la progettazione delle interfacce
ID6	Supporto per l'importazione della struttura organizzativa da altri sistemi
ID7	Supporto per l'assegnazione automatica o manuale dei ruoli agli utilizzatori & Descrizione dei privilegi di accesso.
ID8	Traduzione (automatica/ manuale) del modello di processo di business in modello eseguibile (esempi: WS-BPEL, YAWL).
	Fase di distribuzione
ID9	Supporto per la esecuzione distribuita.
ID10	Supporto per la integrazione con altri sistemi e servizi.
ID11	Supporto alla gestione dei multi tenancy per i BP.
ID12	Supporto all'esecuzione su Cloud Computing.
	Fase di esecuzione operativa
ID13	Gestione delle versioni dei Business Process
ID14	Gestione della documentazione.
	Fase di monitoraggio e controllo
ID15	Monitoraggio consumo di Risorse
ID16	Supporto per il monitoraggio dei file di log e per la rilevazione delle informazioni richieste circa il monitoraggio.
ID17	Possibilità di cambiare il ruolo delle risorse per un'attività di una istanza di processo.
ID18	Supporto per cambiare le regole di business per una particolare istanza.
ID19	Supporto per cambiare il carico di lavoro tra gli utilizzatori
ID20	Supporto di dashboard e report.
	Fase di analisi
ID21	Supporto per la verifica dei processi.
ID22	Supporto per la simulazione dei processi.
ID23	Supporto per il monitoraggio dei file di log e per la rilevazione delle informazioni richieste circa il monitoraggio.
	Fattori Vari
ID24	Supporto per generare documentazione in PDF o HTML.
ID25	Training, esempio video training, casi di studio, esempi, webinars ed eventi di comunità.

Codice Indicatore	AgilePoint NX	Appian	Aura	Axon Ivy
ID1	BPML + BPMN	BPEL4WS	Visula Avaya Aura Orchestration Designer + Java	Import BPMN 2.0, export XML
ID2	Sì	Sì	Sì	Sì
ID3	Sì	Sì	Sì	
ID4	Sì	Sì	Sì	Sì
ID5	Sì	Sì	Sì	Sì
ID6	Sì	Sì	Sì	Sì
ID7	Sì	Sì	Sì	Sì
ID8	Sì	Sì	Sì	Sì
ID9	Sì	Sì	Sì	Sì
ID10	Sì	Sì	Sì	Sì
ID11	Sì, tramite Microsoft Azure e Amazon Web Services (AWS)	Non è stato possibile rilevarlo	Non è stato possibile rilevarlo	Sì
ID12	Sì	Sì	Sì	Sì
ID13	Sì	Sì	Sì	Sì
ID14	Sì	Sì	Sì	Sì
ID15	Sì. AgilePoint provides tools to monitor customer apps and platform operations in real time	No	Sì	Non è stato possibile rilevarlo
ID16	Sì	Non è stato possibile rilevarlo	Sì	Sì
ID17	Sì	Sì	Sì	Sì
ID18	Sì	Sì	Sì	Sì
ID19	Sì	Sì	Sì	Sì
ID20	Sì	Sì	Act smart by leveraging a wide range of functionalities including Native Reporting, Impact Identification, Deep BI, Workload Distribution, Business Rules, Analytics, BAM…	Sì
ID21	Non è stato possibile rilevarlo	Sì	Sì	Non è stato possibile rilevarlo
ID22	Non è stato possibile rilevarlo		Sì	Non è stato possibile rilevarlo
ID23	Preintegrated support for third-party performance-management tools.	Sì	Sì	Sì
ID24	Sì	Sì	Sì	Sì
ID25	learning materials for advanced topics; structured learning journeys; a developer community site with access to learning materials; a commercial marketplace with a private option; vendor-supported applications and/or accelerators.	Blog, webinar, eBook, Guides, Event	Blog, webinar, eBook, Guides, Event	Blog, webinar, eBook, Guides, Event

Software Engineering Research & Practices

Codice Indicatore	Bizagi	Bonita	Process Director di BP Logix	TotalAgility
ID1	BPMN 2.0	BPMN 2.0	BPMN 2.0	XPDL
ID2	Sì	Sì	Sì	Sì
ID3	Sì	Sì	Sì	Sì
ID4	Sì	Sì	Sì	Sì
ID5	Sì	Sì	Sì	Sì
ID6	Sì	Sì	Sì	Non è stato possibile rilevarlo
ID7	Sì	Sì	Sì	Sì
ID8	Sì	Sì	Sì	Sì
ID9	Sì	Sì	Sì	Sì
ID10	Sì	Sì	Sì	Sì
ID11	Sì	Sì	Sì	Sì
ID12	Sì	Sì	Sì	Sì
ID13	Sì	Sì	Sì	Sì
ID14	Sì	Sì	Sì	Sì
ID15	Sì	Sì	Sì	Sì
ID16	Sì	Sì	Sì	Sì
ID17	Sì	Sì	Sì	Sì
ID18	Sì	Sì	Sì	Non è stato possibile rilevarlo
ID19	Sì	Sì	Sì	Sì
ID20	Sì	Sì	Sì	No
ID21	Sì	Sì	Sì	Sì
ID22	Sì	Sì	Sì	Non è stato possibile rilevarlo
ID23	Sì	Sì	Sì	Sì
ID24	Sì	Sì	Sì	Sì
ID25	Blog, webinar, eBook, Guides, Event	Blog, webinar, eBook, Guides, Event	Blog, webinar, eBook, Guides, Event	Blog, webinar, eBook, Guides, Event

Software Engineering Research & Practices

Codice Indicatore	K2 Five	OmniFlow	Pega
ID1	BPEL	BPEL	XPDL 2.0
ID2	Sì	Sì	Sì
ID3	Sì	Non è stato possibile rilevarlo	Sì
ID4	Non è stato possibile rilevarlo	Sì	Sì
ID5	Sì	Sì	Sì
ID6	Sì	Sì	Sì
ID7	Sì	Sì	Sì
ID8	Sì	Sì	Sì
ID9	Sì	Sì	Sì
ID10	Lots of line-of-business systems? Integrate with our patented technology.	Sì	Sì
ID11	Sì	Sì	Sì
ID12	Sì	Sì	Sì
ID13	Sì	Sì	Sì
ID14	Sì	Sì	Sì
ID15	Sì	Sì	Sì
ID16	Sì	Non è stato possibile rilevarlo	Sì
ID17	Sì	Sì	Sì
ID18	Sì	Non è stato possibile rilevarlo	Sì
ID19	Sì	Non è stato possibile rilevarlo	Sì
ID20	Sì	Sì	Sì
ID21	Sì	Sì	Sì
ID22	Non è stato possibile rilevarlo	Non è stato possibile rilevarlo	Sì
ID23	Sì	Non è stato possibile rilevarlo	Sì
ID24	Sì	Sì	Sì
ID25	Blog, webinar, eBook, Guides, Event	Blog, webinar, eBook, Guides, Event	Blog, webinar, eBook, Guides, Event

Software Engineering Research & Practices

Codice Indicatore	Camunda	IBM	Microsoft	Jamio	Oracle
ID1	BPMN 2.0	BPMN 2.0+	BPMN 2.0	XML	BPMN 2.0 + BPEL
ID2	Sì	Sì	Sì	Sì	Sì
ID3	Sì	Sì	Sì	Sì	Sì
ID4	Sì	Sì	Sì	Non è stato possibile rilevarlo	Sì
ID5	Sì	Sì	Sì	Sì	Sì
ID6	Sì	Sì	Sì	Sì	Sì
ID7	Sì	Sì	Sì	Sì c'è Organizzation Manager	Sì
ID8	Sì	Sì	Sì	Sì	Sì
ID9	Sì	Sì	Sì	Sì	Sì
ID10	Sì	Sì	Sì	Sì	Sì
ID11	Sì	Sì	Sì	Sì	Sì
ID12	Sì	Sì	Sì	Sì	Sì
ID13	Sì	Sì	Sì	Sì ogni 3 mesi esce una versione	Sì
ID14	Sì	Sì	Sì	No	Sì
ID15	Sì	Sì	Sì	Sì	Sì
ID16	Sì	Sì	Sì	Sì	Sì
ID17	Sì	Sì	Sì	Sì	No
ID18	Sì	Sì	Sì	Sì	Sì
ID19	Non è stato possibile rilevarlo	Sì	Sì	Sì	Non è stato possibile rilevarlo
ID20	Sì	Sì	Sì	Sì	Sì
ID21	Sì	Sì	Sì	Sì	Sì
ID22	Non è stato possibile rilevarlo	Sì	Sì	Non è stato possibile rilevarlo	Sì
ID23	Sì	Sì	Sì	Sì	Sì
ID24	Sì	Sì	Sì	Sì	Sì
ID25	Blog, webinar, eBook, Guides, Event	Blog, webinar, eBook, Guides, Event	Blog, webinar, eBook, Guides, Event	Blog, webinar, eBook, Guides, Event	Blog, webinar, eBook, Guides, Event

Codice Indicatore	Red Hat	Tibco	LSPS
ID1	BPMN 2.0	XPDL 2.1, ebXML, WS-BPEL	GO-BPMN
ID2	Sì	Sì	Sì
ID3	Sì	Sì	Sì
ID4	Sì	Sì	Sì
ID5	Sì	Sì	Sì
ID6	Sì	Sì	Sì
ID7	Sì	Sì	Sì
ID8	Sì	Non è stato possibile rilevarlo	Non è stato possibile rilevarlo
ID9	Sì	Sì	Sì
ID10	Sì	Sì	Sì
ID11	Sì	Non è stato possibile rilevarlo	Non è stato possibile rilevarlo
ID12	Sì	Sì	Sì
ID13	Sì	Sì	Sì
ID14	Sì	Sì	Sì
ID15	Sì	Sì	Sì
ID16	Sì	Sì	Sì
ID17	Sì	Sì	Sì
ID18	Sì	Sì	Sì
ID19	Sì	Sì	Sì
ID20	Sì	Sì	Sì
ID21	Sì	Sì	Sì
ID22	Sì	Non è stato possibile rilevarlo	Non è stato possibile rilevarlo
ID23	Sì	Non è stato possibile rilevarlo	Non è stato possibile rilevarlo
ID24	Sì	Sì	Sì
ID25	Blog, webinar, eBook, Guides, Event	Blog, webinar, eBook, Guides, Event	Blog, webinar, eBook, Guides, Event

Software Engineering Research & Practices