ThoughtWorks®

TECHNOLOGY RADAR *VOL.17*

Punti di vista sulla tecnologia
e le tendenze che daranno
forma al futuro

thoughtworks.com/radar

#TWTechRadar

HANNO CONTRIBUITO

Il Technology Radar viene redatto dal
ThoughtWorks Technology Advisory Board, di cui fanno parte:

Rebecca Parsons (CTO) | Martin Fowler (Chief Scientist) | Bharani Subramaniam | Camilla Crispim | Erik Doernenburg
Evan Bottcher | Fausto de la Torre | Hao Xu | Ian Cartwright | James Lewis
Jonny LeRoy | Ketan Padegaonkar | Lakshminarasimhan Sudarshan | Marco Valtas | Mike Mason
Neal Ford | Rachel Laycock | Scott Shaw | Shangqi Liu | Zhamak Dehghani

CHE NOVITÀ CI SONO?

Temi alla ribalta in questa edizione:

L'OPEN SOURCE STA CRESCENDO IN CINA

La marea sta salendo. In seguito a cambiamenti di politica e di atteggiamento, le grandi aziende cinesi come Alibaba e Baidu stanno rilasciando rapidamente framework, strumenti e piattaforme open source. La crescita dei loro ecosistemi software accelera in sintonia con la loro espansione economica.

Dato il numero di progetti software nei mercati cinesi, che sono enormi e in rapida espansione, il numero e la qualità dei progetti open source che appaiono su GitHub e altri siti open source è destinato ad aumentare. Come mai le aziende cinesi aprono il software di così tanti progetti? La situazione è simile a quella di altri mercati caldi del software, come la Silicon Valley, dove la competizione per accaparrarsi gli sviluppatori è intensa e offrire un salario più ricco non basta.

La prospettiva di lavorare su progetti open source all'avanguardia, insieme ad altri sviluppatori di valore, è un incentivo universale. Ci aspettiamo che, fra i progetti open source più innovativi, continui il trend dei file di README scritti prima in cinese e poi in inglese.

KUBERNETES, L'ORCHESTRATORE DI CONTAINER PREFERITO

Un gran numero di elementi del Radar riguardano Kubernetes e il suo ruolo sempre più importante in molti progetti. Sembra che l'ecosistema dello sviluppo software si stia stabilizzando su Kubernetes e i tool collegati, per risolvere i problemi di deployment, scalabilità e gestione dei container.

Elementi del Radar come GKE, Kops, e Sonobuoy introducono strumenti e servizi che migliorano l'esperienza complessiva di adottare e manutenere Kubernetes. Infatti, la semplicità di lanciare container multipli come singola unità di scheduling abilita il service mesh e sidecar for endpoint security.

Kubernetes è diventato il sistema operativo di default per i container: molti fornitori di cloud lo hanno adottato, grazie alla sua architettura aperta e modulare, e ci sono strumenti che sfruttano le sue API che danno accesso ad astrazioni quali workload, cluster, configurazioni e storaggio.

Notiamo sempre più prodotti che usano Kubernete come ecosistema, trattandolo come un livello di astrazione che sta sopra ai livelli dei microservizi e dei container. Tutto ciò prova che gli sviluppatori riescono a sfruttare con successo gli stili architetturali moderni, nonostante le complessità proprie dei sistemi distribuiti.

IL CLOUD È LA NORMA

Un altro tema pervasivo della conversazione, che ruota intorno a questa edizione, è di natura decisamente "nuvolosa". Mentre i fornitori di cloud stanno diventando più capaci e si avvicinano alla parità di feature, il modello basato sul cloud pubblico è ormai la norma in molte organizzazioni.

Invece di chiedersi "Perchè nel cloud?", quando partono nuovi progetti, molte aziende ora si chiedono "perché *non* nel cloud?" Certamente, alcuni tipi di software richiedono ancora sistemi on-premise, ma a mano a mano che i prezzi calano e le capacità si espandono, lo sviluppo cloud-native diventa sempre più facile.

È vero che i vendor più importanti forniscono tutti le stesse feature di base, ma ciascuno di essi offre anche caratteristiche uniche che lo differenziano, per certi specifici tipi di soluzioni. Così, vediamo che alcune aziende sfruttano diversi vendor in stile Polycloud, scegliendo le capacità specializzate di quelle piattaforme, che sono le più adatte per le esigenze dei loro clienti.

LA FIDUCIA NEL BLOCKCHAIN È DISTRIBUITA IN MANIERA PIÙ UNIFORME

Nonostante il caos che circonda le criptovalute nei mercati, molti dei nostri clienti stanno trovando soluzioni basate su blockchain per realizzare ledger distribuiti e smart contracts. Molti elementi del Radar dimostrano la maturità delle tecnologie basate su blockchain, con maniere sempre più interessanti di realizzare smart contracts, e una varietà di tecniche e linguaggi di programmazione.

Il Blockchain risolve l'antico problema della fiducia distribuita, e di ledger condivisi e indelebili. Le aziende oggi stanno aumentando la fiducia dei loro utenti nei meccanismi che sottostanno l'implementazione del blockchain. Molte industrie hanno segnatamente problemi che riguardano la fiducia distribuita; ci aspettiamo che le soluzioni basate su blockchain continuino a trovare il modo di risolverli.

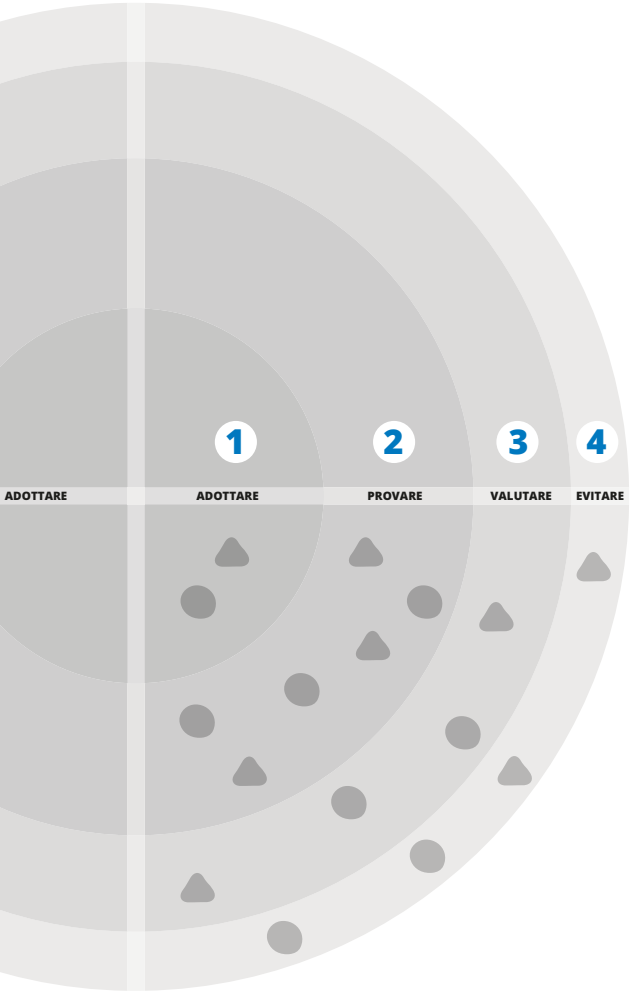
A PROPOSITO DEL RADAR

I ThoughtWorkers sono appassionati di tecnologia. La costruiamo, facciamo ricerca su di essa, la testiamo, ne apriamo il sorgente, ne scriviamo e cerchiamo costantemente di migliorarla, per tutti. La nostra missione è di essere i campioni dell'eccellenza nel software e di rivoluzionare l'IT. Creiamo e condividiamo il ThoughtWorks Technology Radar per supportare questa missione. Il Radar è creato dal ThoughtWorks Technology Advisory Board, un gruppo di leader tecnologici senior in ThoughtWorks. I leader si incontrano regolarmente per discutere la strategia tecnologica globale per ThoughtWorks e i trend tecnologici che impattano la nostra industria in modo significativo.

Il Radar cattura l'esito delle discussioni del Technology Advisory Board in un formato che fornisce valore a un ampio spettro di stakeholder, dagli sviluppatori ai CTO. Il contenuto del Radar ne è un sommario conciso.

Incoraggiamo chiunque a esplorare queste tecnologie per avere maggiori dettagli. Il Radar ha un formato grafico, e raggruppa gli elementi in tecniche, strumenti, piattaforme e linguaggi & framework. Quando elementi del Radar potrebbero apparire in più di un quadrante, scegliamo quello che ci sembra più appropriato. Raggruppiamo ulteriormente questi elementi in maniera da riflettere la nostra opinione corrente su di essi.

Per maggiori informazioni sul Radar, vedi thoughtworks.com/radar/faq



IL RADAR A COLPO D'OCCHIO

1 ADOTTARE

Siamo fortemente convinti che l'industria dovrebbe adottare questi elementi. Noi li usiamo nei nostri progetti.

2 PROVARE

Vale la pena di provare a usare queste tecnologie. È importante portare in casa la conoscenza. Vanno testate su progetti su cui ci si può permettere il rischio.

3 VALUTARE

Vale la pena di esplorare, con l'obiettivo di capire che impatto possono avere sulla vostra azienda.

4 EVITARE

Procedete a vostro rischio.

▲ NUOVI O CAMBIATI

● NON MODIFICATI

Gli elementi che sono nuovi o che hanno avuto modifiche significative dall'ulttimo Radar sono rappresentati da triangoli, mentre gli elementi che non si sono mossi sono rappresentati come cerchi

! Il nostro Radar guarda avanti. Per fare spazio per i nuovi elementi, lasciamo sbiadire gli elementi che non sono cambiati di recente. Questo non è perché non siano validi, ma per limiti di spazio.

IL RADAR

TECNICHE

ADOTTARE
1. Lightweight Architecture Decision Records

PROVARE
2. Applicare Product Management a piattaforme interne *NEW*
3. Funzione di fitness dell'architettura *NEW*
4. Autonomous bubble pattern *NEW*
5. Chaos Engineering *NEW*
6. Disaccoppiare la gestione dei segreti dal codice sorgente
7. DesignOps *NEW*
8. Legacy in scatola
9. Micro frontends
10. Pipelines per infrastructure as code *NEW*
11. Serverless architecture
12. TDD per container *NEW*

VALUTARE
13. Algorithmic IT operations *NEW*
14. Ethereum per applicazioni decentralizzate *NEW*
15. Event streaming come sorgente di verità *NEW*
16. Platform engineering product teams
17. Polycloud *NEW*
18. Service mesh *NEW*
19. Sidecar per endpoint security *NEW*
20. Le tre R della sicurezza *NEW*

EVITARE
21. Un solo server CI per tutti il team
22. Teatrino CI
23. Ambienti di integrazione condivisi per tutta l'azienda
24. Antipattern ESB con Kafka *NEW*
25. Generazione di codice basata su Spec

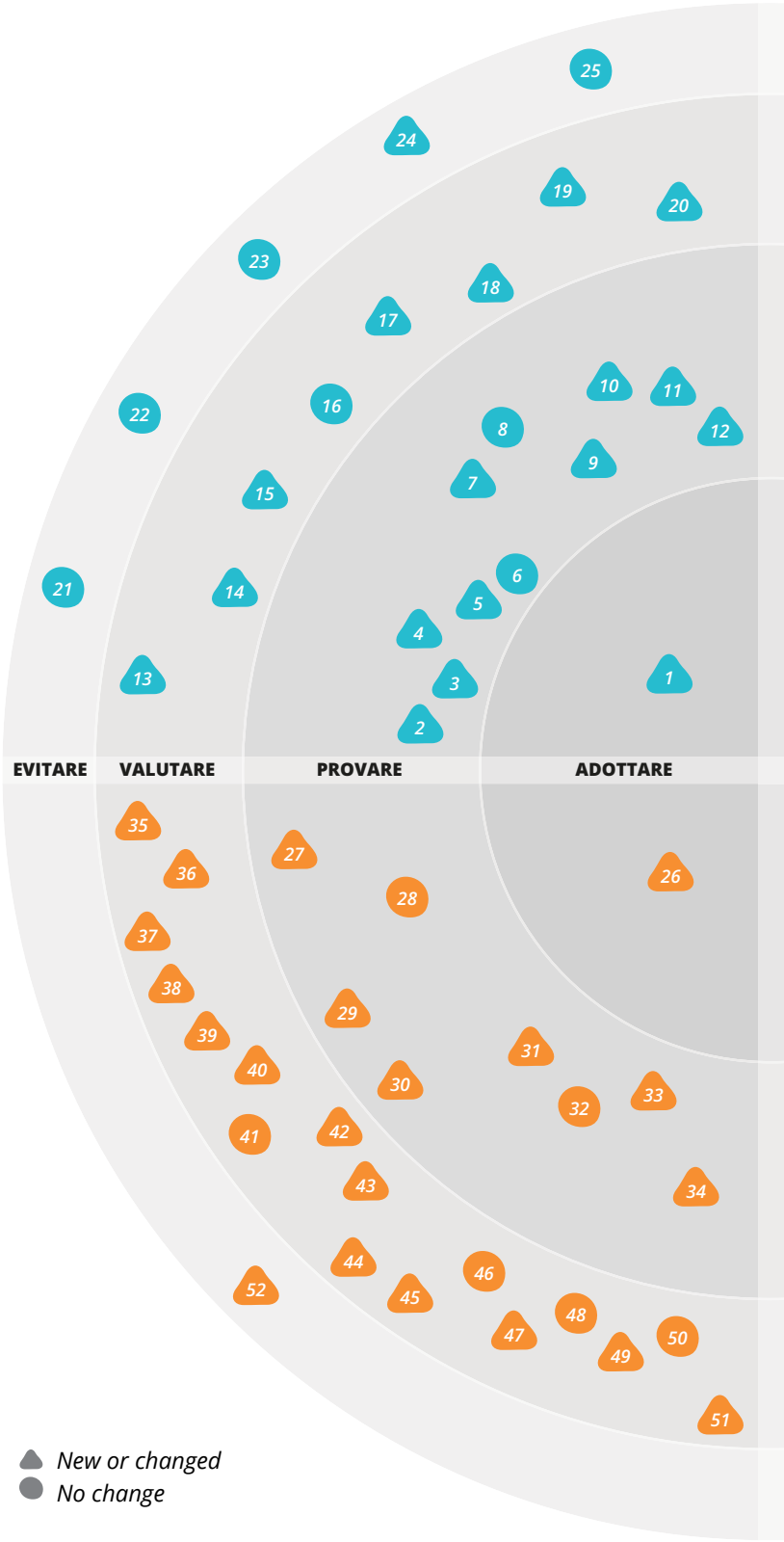
PIATTAFORME

ADOTTARE
26. Kubernetes

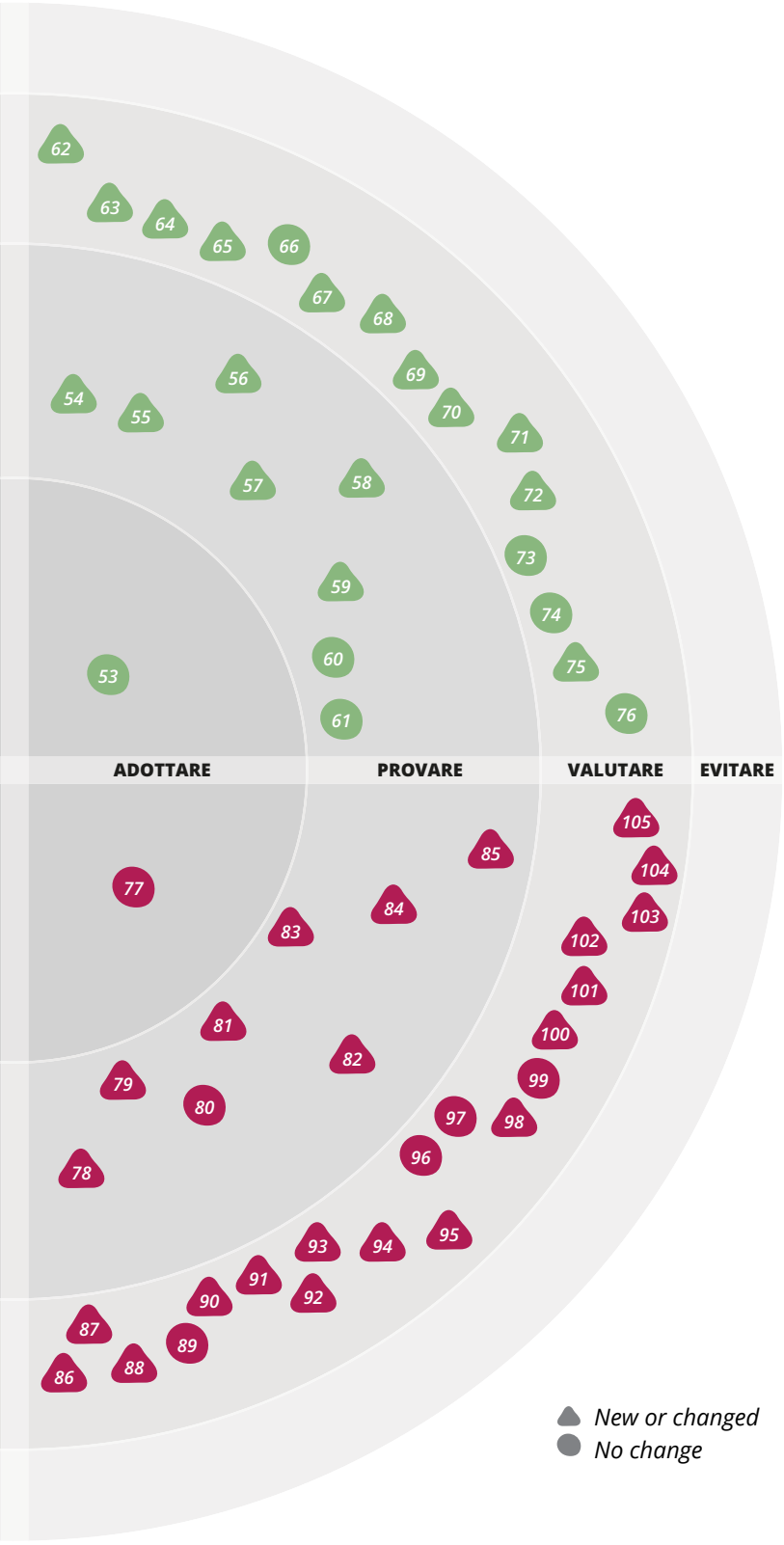
PROVARE
27. .NET Core
28. AWS Device Farm
29. Flood IO *NEW*
30. Google Cloud Platform *NEW*
31. Keycloak
32. OpenTracing
33. Unity al di là del gaming
34. WeChat *NEW*

VALUTARE
35. Azure Service Fabric *NEW*
36. Cloud Spanner *NEW*
37. Corda *NEW*
38. Cosmos DB *NEW*
39. DialogFlow
40. GKE *NEW*
41. Hyperledger
42. Kafka Streams
43. Language Server Protocol *NEW*
44. LoRaWAN *NEW*
45. MapD *NEW*
46. Mosquitto
47. Netlify *NEW*
48. PlatformIO
49. TensorFlow Serving *NEW*
50. Voice platforms
51. Windows Containers *NEW*

EVITARE
52. API gateways troppo ambiziosi



IL RADAR



STRUMENTI

ADOTTARE
53. fastlane

PROVARE
54. Buildkite *NEW*
55. CircleCI *NEW*
56. gopass *NEW*
57. Headless Chrome per test di front-end *NEW*
58. jsoniter *NEW*
59. Prometheus
60. Scikit-learn
61. Serverless Framework

VALUTARE
62. Apex *NEW*
63. assertj-swagger *NEW*
64. Cypress *NEW*
65. Flow *NEW*
66. InSpec
67. Jupyter *NEW*
68. Kong API Gateway *NEW*
69. kops *NEW*
70. Lighthouse *NEW*
71. Rendertron *NEW*
72. Sonobuoy *NEW*
73. spaCy
74. Spinnaker
75. Spring Cloud Contract *NEW*
76. Yarn

EVITARE

LINGUAGGI & FRAMEWORK

ADOTTARE
77. Python 3

PROVARE
78. Angular
79. AssertJ *NEW*
80. Avro
81. CSS Grid Layout *NEW*
82. CSS Modules *NEW*
83. Jest *NEW*
84. Kotlin
85. Spring Cloud

VALUTARE
86. Android Architecture Components *NEW*
87. ARKit/ARCore *NEW*
88. Atlas and BeeHive *NEW*
89. Caffe
90. Clara rules *NEW*
91. CSS-in-JS *NEW*
92. Digdag *NEW*
93. Druid *NEW*
94. ECharts *NEW*
95. Gobot *NEW*
96. Instana
97. Keras
98. LeakCanary *NEW*
99. PostCSS
100. PyTorch *NEW*
101. single-spa *NEW*
102. Solidity *NEW*
103. TensorFlow Mobile *NEW*
104. Truffle *NEW*
105. Weex *NEW*

EVITARE

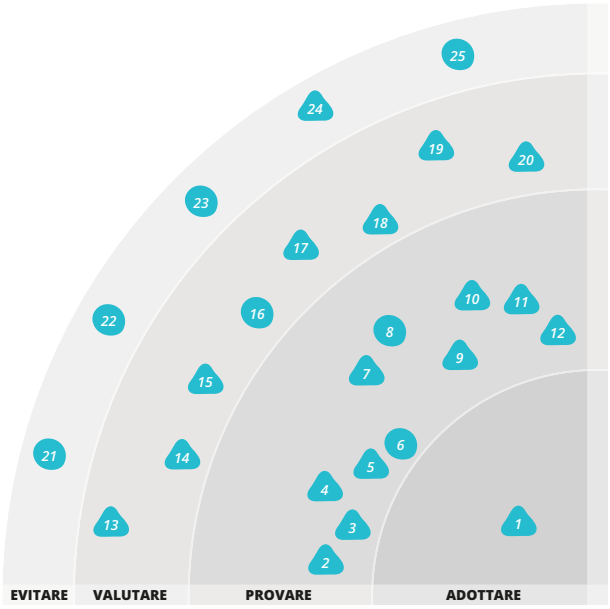
TECNICHE

ADOTTARE
1. Lightweight Architecture Decision Records

PROVARE
2. Applicare product management a piattaforme interne *NEW*
3. Funzione di fitness dell'architettura *NEW*
4. Autonomous bubble pattern *NEW*
5. Chaos Engineering *NEW*
6. Disaccoppiare la gestione dei segreti dal codice sorgente
7. DesignOps *NEW*
8. Legacy in scatola
9. Micro frontends
10. Pipelines per infrastructure as code *NEW*
11. Serverless architecture
12. TDD per container *NEW*

VALUTARE
13. Algorithmic IT operations *NEW*
14. Ethereum per applicazioni decentralizzate *NEW*
15. Event streaming come sorgente di verità *NEW*
16. Platform engineering product teams
17. Polycloud *NEW*
18. Service mesh *NEW*
19. Sidecar per endpoint security *NEW*
20. Le tre R della sicurezza *NEW*

EVITARE
21. Una sola istanza di CI per tutti i team
22. Teatrino CI
23. Ambienti di integrazione condivisi per tutta l'azienda
24. Antipattern ESB con Kafka *NEW*
25. Generazione di codice basata su Spec



Molta documentazione può essere sostituita dal codice stesso, purché sia molto leggibile, e dai test. Tuttavia, in un mondo di architetture evolutive, è importante annotare certe decisioni di design, sia a beneficio dei futuri membri del team, che per avere un parere esterno. **LIGHTWEIGHT ARCHITECTURE DECISION RECORDS** è una tecnica per tenere traccia delle decisioni architetturali importanti, insieme al loro contesto e alle loro conseguenze. Raccomandiamo di salvare questi dettagli nel sistema di controllo dei sorgenti, invece che in un wiki o in un sito web, perché così restano aggiornati con il codice stesso. Per la maggior parte dei progetti, pensiamo che non ci sia ragione di non usare questa tecnica.

Negli ultimi 12 mesi abbiamo visto un forte aumento di interesse per le piattaforme digitali . Le aziende che desiderano sviluppare rapidamente ed efficacemente nuove soluzioni digitali stanno costruendo piattaforme interne che offrono ai team l'accesso self-service alle API aziendali, agli strumenti, alla conoscenza e al supporto necessari per costruire e gestire le proprie soluzioni. Troviamo che queste piattaforme siano più efficaci quando vengono gestite con lo stesso rispetto di un prodotto esterno. **APPLICARE PRODUCT MANAGEMENT A PIATTAFORME INTERNE** significa stabilire empatia con i consumatori interni (leggi: gli sviluppatori) e collaborare con loro sul progetto. I Product Manager definiscono roadmap e assicurano che la piattaforma sia di valore per l'azienda e che migliori l'esperienza degli sviluppatori. Alcuni Platform Product Manager creano persino una brand identity per la piattaforma interna e la utilizzano per commercializzare i vantaggi ai propri colleghi. I PPM si occupano anche della qualità della piattaforma, di collezionare metriche di utilizzo e di migliorare in maniera continua la piattaforma nel tempo. Trattare la piattaforma come prodotto aiuta a creare un' ecosistema fiorente ed evita l'insidia di costruire l'ennesima architettura di servizi stagnante e inutilizzata.

Un concetto preso in prestito dall'evolutionary computing, la funzione di fitness viene usata per riassumere con un numero quanto una soluzione di design sia adatta a raggiungere il suo scopo preposto.

(funzione di fitness dell'architettura)

Un concetto preso in prestito dall'evolutionary computing, la funzione di fitness viene usata per riassumere con un numero quanto una soluzione di design sia adatta a raggiungere il suo scopo preposto. Quando si definisce un algoritmo evolutivo, il progettista cerca un altoritmo "migliore"; la funzione di fitness definisce che cosa si intenda per "migliore" in quel contesto. Una **FUNZIONE DI FITNESS DELL'ARCHITETTURA**, definita in Building Evolutionary Architectures, fornisce una valutazione oggettiva di alcune caratteristiche architetturali, che possono racchiudere vari criteri di valutazione, come unit testing, metriche, monitor e così via. Noi crediamo che gli architetti possano comunicare, validare e preservare le caratteristiche architetturali in maniera continua e automatizzata, il che è la chiave per costruire architetture evolutive.

Gli strumenti di CI e CD possono essere usati per testare la configurazione di server, la costruzione di immagini di server, il provisioning e l'integrazione di ambienti.

(Pipeline per infrastructure as code)

Molte organizzazioni con cui lavoriamo si sforzano di usare approcci architetturali moderni per realizzare nuove capability e feature, ma allo stesso tempo devono convivere con una coda lunga di sistemi legacy. Una vecchia strategia che, secondo la nostra esperienza, si è rivelata di crescente utilità in questi scenari è il pattern **AUTONOMOUS BUBBLE PATTERN** pubblicato da Eric Evans. Questo approccio si basa sulla creazione di un contesto nuovo per gli sviluppi applicativi nuovi, che sia schermato e protetto dal mondo legacy. Questo pattern è un passo oltre il semplice anticorruption layer. Fornisce al nuovo bubble context un controllo totale sui suoi dati propri, che vengono mantenuti aggiornati in maniera asincrona con il sistema legacy. Questo pattern richiede un po' di lavoro per proteggere i confini della bolla e mantenere entrambi i mondi consistenti, ma la maggiore autonomia e la riduzione della frizione fra il nuovo sviluppo e il legacy, forniscono agli sviluppatori un primo, potente passo avanti verso una cultura architetturale più moderna.

In precedenti edizioni del Radar, abbiamo parlato del Chaos Monkey di Netflix per testare come un sistema in esecuzione in produzione sia capace di affrontare interruzioni nella fornitura di corrente elettrica, disabilitando istanze a caso e misurando i risultati. **CHAOS ENGINEERING** è il termine neonato per un'applicazione generale di questa tecnica. Conducendo esperimenti su sistemi distribuiti in produzione, ci assicuriamo che tali sistemi funzionino come ci si aspetta in situazioni turbolente. Un buon punto di partenza per comprendere questa tecnica è il sito web Principles of Chaos Engineering.

Ispirato al movimento DevOps, **DESIGNOPS** è un cambiamento culturale e un insieme di pratiche, che consentono alle persone di un'organizzazione di riprogettare continuamente i prodotti senza compromettere la qualità, la coerenza dei servizi e l'autonomia del team. DesignOps sostiene la creazione e l'evoluzione di un'infrastruttura che minimizza lo sforzo necessario alla creazione di nuovi concetti e variazioni della UI e che permette cicli di feedback con gli utenti finali rapidi ed affidabili. Grazie a strumenti come lo Storybook che promuove una stretta collaborazione, la necessità di analisi approfondite e di specifiche prima di iniziare lo sviluppo si riducono al minimo. Con il DesignOps, il design evolve da pratica specifica a essere una componente del lavoro di tutto il team.

Abbiamo visto benefici significativi dall'introduzione di architetture a microservizi, che hanno permesso ai team di rilasciare e gestire servizi indipendenti gli uni dagli altri. Purtroppo abbiamo anche visto che molti team creano front-end monolitici, una singola, grande applicazione browser che si espande in modo incontrollato, sopra ai loro servizi di back-end. Il nostro approccio preferito (e collaudato) è quello di dividere il codice basato su browser in **MICRO FRONTENDS**. In questo approccio, l'applicazione web è suddivisa nelle sue funzionalità e ogni funzionalità dal frontend al backend è gestita da un team diverso. Questo assicura che ogni funzionalità sia sviluppata, testata e implementata indipendentemente da altre funzionalità. Esistono varie tecniche per ricomporre le funzionalità, a volte come pagine, a volte come componenti, in un'esperienza utente coesa.

L'utilizzo di pipeline di continuous delivery per orchestrare il processo di rilascio del software è diventato un concetto standard. Tuttavia, il test automatico delle modifiche al codice di infrastruttura non è altrettanto ben capito. Gli strumenti di Continuous integration (CI) e continuous delivery (CD) possono essere usati per testare: la configurazione di

server (per esempio: Chef cookbooks, Puppet modules, Ansible playbooks), la costruzione di immagini di server (per esempio: Packer), il provisioning di ambienti (per esempio: Terraform, CloudFormation) e l'integrazione di ambienti. L'utilizzo di **PIPELINE PER INFRASTRUCTURE AS CODE** consente di rilevare gli errori prima che le modifiche siano applicate agli ambienti operazionali, inclusi gli ambienti usati per lo sviluppo ed il test. In questo modo inoltre si garantisce che gli strumenti che generano l'infrastruttura vengano eseguiti in modo consistente da agenti di CI/CD, invece che da singole workstation. Rimangono comunque alcune difficoltà, come il lungo ciclo di feedback dovuto al tempo necessario a creare container e macchine virtuali. Ciononostante, abbiamo trovato che questa sia una tecnica valida.

L'impiego di **ARCHITETTURE SERVERLESS** è diventato molto rapidamente un approccio accettato dalle organizzazioni per installare applicazioni cloud, con una pletora di scelte disponibili per il deploy. Perfino organizzazioni tradizionalmente conservative stanno facendo un uso parziale di alcune tecnologie serverless. L'aspetto principale verte su Functions as a Service (per esempio: AWS Lambda, Google Cloud Functions, Azure Functions) mentre stanno ancora emergendo pattern d'uso appropriati. Installare funzioni serverless rimuove innegabilmente il lavoro non banale che tradizionalmente è fatto sul server e sulla configurazione ed orchestrazione del sistema operativo. Le funzioni serverless, però, non sono idonee per qualunque esigenza. Per il momento, bisogna essere preparati a tornare al deploy di container o addirittura di istanze di server per requisiti specifici. Tuttavia, gli altri componenti dell'architettura serverles, come Backend as a Service, sono diventati una scelta quasi scontata.

Per scrivere codice applicativo molti team di sviluppo hanno adottato pratiche di sviluppo guidate dai test (TDD) per via dei loro benefici. Altri hanno adottato i container per impacchettare ed installare i propri software, ed è una pratica accettata quella di usare script automatici per costruire tali container. Recentemente abbiamo osservato alcuni team combinare le due tendenze guidando la scrittura degli script per i container utilizzando dei test. Con framework come Serverspec e Goss, potete esplicitare le funzionalità desiderate per container isolati o orchestrati, con brevi cicli di feedback. Ciò significa che è possibile adottare gli stessi principi che abbiamo caldamente raccomandato per il codice nel **TDD PER CONTAINER**. La nostra esperienza iniziale nel fare questo è stata molto positiva.

I blockchain sono stati ampiamente reclamizzati come la panacea per tutti i problemi della tecnologia finanziaria, dal banking alle valute digitali, fino alla trasparenza nella supply chain.

(Ethereum per applicazioni decentralizzate)

La quantità di dati raccolti da operations in IT è in aumento da anni. Per esempio, il trend verso i microservizi implica che un numero crescente di applicazioni producono i propri dati di operations, e strumenti come Splunk, Prometheus, o lo stack ELK facilitano il salvataggio dei dati e la loro successiva elaborazione, per far luce sulle operations. In combinazione con gli strumenti di machine learning, ormai accessibili a tutti, è inevitabile che gli operatori comincino a utilizzare anche modelli statistici e algoritmi di classificazione basati su machine learning. Anche se questi algoritmi sono disponibili da anni, e ci siano stati vari tentativi di automatizzare la gestione dei servizi, solo ora cominciamo a capire come le macchine e gli esseri umani possano collaborare per identificare in anticipo i problemi in produzione, o identificare il punto di origine dei guasti. Sappiamo che c'è un rischio di eccedere nell'entusiasmo per **ALGORITHMIC IT OPERATIONS**, tuttavia i continui miglioramenti negli algoritmi di machine learning cambieranno inevitabilmente il ruolo degli esseri umani nella maniera di gestire i data center di domani.

I blockchain sono stati ampiamente reclamizzati come la panacea per tutti i problemi della tecnologia finanziaria, dal banking alle valute digitali, fino alla trasparenza nella supply chain. Nelle precedenti edizioni abbiamo parlato di Ethereum per via delle sue feature, fra cui gli smart contracts. Adesso stiamo assistendo a nuovi sviluppi che usano **ETHEREUM PER APPLICAZIONI DECENTRALIZZATE** in altri campi. Anche se questa tecnologia è molto giovane, troviamo incoraggiante che venga usata per costruire applicazioni decentralizzate, in campi diversi dalle criptovalute o dalle banche.

Con l'aumento della popolarità delle piattaforme di streaming, come Apache Kafka, molti le considerano solo una variante avanzata di code di messaggi, da usare solamente per trasmettere eventi. Anche quando vengono usate in questa maniera, l'event streaming ha i suoi vantaggi rispetto al message queueing tradizionale. Tuttavia, siamo più interessati a come alcuni usino **EVENT STREAMING COME SORGENTE DI VERITÀ** con piattaforme (particolarmente Kafka) dove lo storaggio primario dei dati è sotto forma di eventi immutabili. Un servizio basato su Event Sourcing, per esempio, può usare Kafka come event store. Gli eventi sono in questo modo disponibili per essere consumati da altri servizi. Questa tecnica promette di ridurre la duplicazione del lavoro fra persistenza e integrazione.

Una service mesh offre consistenza nel discovery, sicurezza, tracciabilità, monitoraggio e gestione dei guasti, senza richiedere un asset condiviso come un API gateway o un ESB.

(Service mesh)

I principali fornitori di cloud (Amazon, Microsoft e Google) sono impegnati in una battaglia serrata per mantenere la parità sulle capacità di base, mentre i loro prodotti si differenziano di poco. Questo fatto porta alcune organizzazioni ad adottare una strategia **POLYCLOUD**: invece che affidarsi totalmente ad un solo provider, affidano tipi diversi di workload a diversi fornitori, scegliendo le offerte più avanzate di ciascuno per quel tipo di applicazione. Per esempio, mettere servizi standard su AWS, ma usare Google per il machine learning, Azure per le applicazioni .Net che usano SQLServer, o magari usare il blockchain dell'Ethereum Consortium. Questa strategia è diversa dalla strategia cloud-agnostica di cercare la portabilità fra i vari provider, che è costosa e costringe a limitarsi al minimo comun denominatore di ciascuna piattaforma. Polycloud invece cerca di sfruttare il meglio che ciascun cloud può offrire.

In questo momento in cui le grandi organizzazioni stanno migrando verso team più autonomi, che rilasciano e gestiscono i loro propri microservizi, come ci si può assicurare la necessaria consistenza e compatibilità fra questi servizi, senza affidarsi ad una struttura centralizzata di hosting? Per quanto autonomi, i microservizi necessitano di alcuni standard organizzativi per collaborare in modo efficiente. Una **SERVICE MESH** offre consistenza nel discovery, sicurezza, tracciabilità, monitoraggio e gestione dei guasti, senza richiedere un asset condiviso come un API gateway o un ESB. Una tipica implementazione consiste in reverse-proxy leggeri installati a fianco di ogni servizio, a volte in un container separato. Questi proxy comunicano con registri di servizi, fornitori di identità, aggregatori di log, e così via. L'interoperabilità ed osservabilità dei servizi sono ottenute attraverso l'implementazione condivisa di questi proxy ma non attraverso la condivisione di una istanza runtime. Siamo stati a favore di un approccio decentralizzato alla gestione dei microservizi da tempo e siamo felici di vedere che questo pattern emerga in maniera consistente. I progetti open source come linkerd e Istio continueranno a maturare rendendo le mesh di servizi ancora più semplici da implementare.

Le architetture di microservizi, con un largo numero di servizi che espongono i propri asset e capacità attraverso API, e con un aumento della superficie attaccabile, richiedono un'architettura di sicurezza a fiducia zero — 'mai fidarsi, verificare sempre'. Tuttavia, il rinforzo dei controlli di sicurezza per la comunicazione fra un servizio e l'altro è spesso trascurato, a causa sia dell'aumento della complessità del codice del servizio stesso, sia della mancanza di librerie e di supporto per i vari linguaggi di programmazione in un ambiente poliglotta. Per aggirare tale complessità, alcuni team delegano la sicurezza ad un sidecar fuori-dal-processo — un processo o container che è installato e schedulato con ogni servizio, condividendone il contesto di esecuzione, host ed identità. I sidecar implementano funzionalità di sicurezza, come cifratura trasparente delle comunicazioni e TLS (Transport Layer Security), e anche autenticazione ed autorizzazione del servizio chiamante o dell'utente finale. Raccomandiamo di valutare l'utilizzo di Istio, linkerd or Envoy prima di implementare i vostri propri **SIDECAR PER ENDPOINT SECURITY**.

Gli approcci tradizionali alla sicurezza aziendale enfatizzano spesso il blocco ed il rallentamento del ritmo di cambiamento. Comunque, sappiamo che più tempo ha un attaccante per compromettere un sistema, maggiore è il danno potenziale. Le tre R della sicurezza enterprise — ruotare, riparare e ripavimentare — traggono vantaggio

dall'automazione dell'infrastruttura e dalla continuous delivery per eliminare opportunità di attacco. Ruotare le credenziali, applicare patch non appena disponibili e ricompilare i sistemi a partire da uno stato noto e sicuro, tutto nel giro di minuti o di ore, rende più difficile per gli attaccanti avere successo. La tecnica delle **TRE R DELLA SICUREZZA** è diventata fattibile con l'avvento delle moderne architetture basate sul cloud. Quando le applicazioni sono installate come container, e compilate e testate con una pipeline completamente atuomatizzata, una patch di sicurezza è solo l'ennesimo piccolo rilascio da mandare attraverso la pipeline con un click. Naturalmente, nel rispetto delle migliori pratiche relative ai sistemi distribuiti, gli sviluppatori hanno bisogno di progettare le proprie applicazioni affinché siano resilienti a indisponibilità improvvise dei server. Questo è simile all'impatto dell'implementazione di Chaos Monkey nel del vostro ambiente.

Kafka sta diventando molto popolare come soluzione di messaggistica, e Kafka Streams è sulla cresta dell'onda dell'interesse nelle architetture streaming. Sfortunatamente, quando le organizzazioni iniziano ad integrare Kafka nel cuore dei propri dati e delle proprie piattaforme applicative, stiamo osservando che in alcuni casi ricreano gli **ANTIPATTERN ESB CON KAFKA** attraverso la centralizzazione dell'ecosistema dei componenti Kafka, come connettori ed elaboratori di stream, invece di lasciare che questi componenti siano creati e rilasciati indipendentemente dai team di prodotto o di servizio. Questo ci riporta alla memoria quanto siano veramente problematici gli antipattern ESB, per cui sempre più logica, orchestrazione e trasformazione viene spinta in un ESB gestito in modo centralizzato, creando una significativa dipendenza su di un team centralizzato. Vogliamo attrarre l'attenzione su questo antipattern per dissuadere dal perseguirlo.

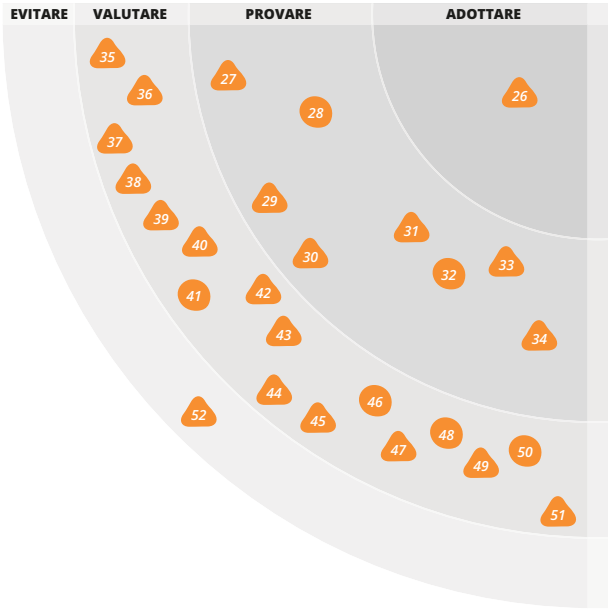
PIATTAFORME

ADOTTARE
26. Kubernetes

PROVARE
27. .NET Core
28. AWS Device Farm
29. Flood IO *NEW*
30. Google Cloud Platform *NEW*
31. Keycloak
32. OpenTracing
33. Unity al di là del gaming
34. WeChat *NEW*

VALUTARE
35. Azure Service Fabric *NEW*
36. Cloud Spanner *NEW*
37. Corda *NEW*
38. Cosmos DB *NEW*
39. DialogFlow
40. GKE *NEW*
41. Hyperledger
42. Kafka Streams
43. Language Server Protocol *NEW*
44. LoRaWAN *NEW*
45. MapD *NEW*
46. Mosquitto
47. Netlify *NEW*
48. PlatformIO
49. TensorFlow Serving *NEW*
50. Voice platforms
51. Windows Containers *NEW*

EVITARE
52. Overambitious API gateways



Dall'ultima volta che abbiamo parlato di **KUBERNETES** nel Radar, è diventato la soluzione di default per la maggior parte dei nostri clienti, quando installano software su container in un cluster. I prodotti alternativi non hanno convinto, e in alcuni casi i nostri clienti stanno cambiando i loro sistemi per adottare Kubernetes. Kubernetes è diventato la piattaforma per l'orchestrazione di container preferita sulle piattaforme cloud più importanti, compresi lo Azure Container Service di Microsoft e Google Cloud. (Si veda la voce GKE). Inoltre, ci sono molti prodotti utili che arricchiscono l'ecosistema di Kubernetes in continua crescita. Al contrario, le piattaforme che cercano di nascondere Kubernetes sotto uno strato di astrazione, non hanno ancora convinto.

L'adozione del framework cross-platform open source **.NET CORE**, è in aumento. Con .NET Core si può sviluppare e rilasciare applicazioni su Windows, macOS e Linux. Grazie alla pubblicazione di .NET Standard 2.0 , che aumenta il numero di API standard disponibili sulle varie piattaforme .NET, la migrazione verso .NET Core è diventata più agevole. I problemi legati alle applicazioni supportate su .NET Core stanno diminuendo, e abbiamo ormai a disposizione strumenti cross-platform di ottima qualità, questo rende agevole lo sviluppo su piattaforme non Windows. Vengono fornite immagini ufficiali di Docker per facilitare l'integrazione di servizi .NET in ambienti containerizzati. Reazioni positive nella comunità e feeback dai nostri progetti indicano che .NET Core è pronto per l'uso generalizzato.

Il load testing è diventato più facile, da quando abbiamo strumenti maturi come Gatling e Locust. Allo stesso tempo, l'elasticità dell'infrastruttura cloud rende possible simulare un gran numero di client. Siamo estasiati dal vedere Flood e altre piattaforme cloud che fanno progressi, facendo leva su queste tecnologie. **FLOOD IO** è un servizio di load testing SaaS che esegue script di test distribuiti su centinaia di server nel cloud. Secondo il nostro team, è facile eseguire i test su Flood, riutilizzando gli script di Gatling esistenti.

Con l'espansione della **GOOGLE CLOUD PLATFORM** (GCP) in termini di disponibilità in varie regioni geografiche e maturità dei servizi, i clienti possono ora considerarla seriamente per le proprie strategie cloud. In alcune aree, GCP ha raggiunto parità funzionale con il suo concorrente principale, Amazon Web Services, mentre in altre aree si è differenziata — in particolare con piattaforme accessibili di machine learning, strumenti di data engineering, ed un valido Kubernetes as a service (GKE). I nostri team non possono non elogiare l'esperienza di sviluppo con gli strumenti e le API della GCP.

Con l'espansione della Google Cloud Platform (GCP) in termini di disponibilità in varie regioni geografiche e maturità dei servizi, i clienti possono ora considerarla seriamente per le proprie strategie cloud.

(Google Cloud Platform)

In un'architettura a microservizi, o qualunque altra architettura distribuita, una delle necessità più comuni è mettere in sicurezza i servizi o l'API attraverso funzionalità di autenticazione ed autorizzazione. Questo è il contesto in cui **KEYCLOAK** entra in gioco. Keycloak è una soluzione open source per la gestione di identità e di accesso che rende semplice mettere in sicurezza applicazioni o microservizi con poco o nessun codice. Esso supporta single sign-on, social login e protocolli standard come OpenID Connect, OAuth 2.0 e SAML I nostri team stanno usando questo strumento e pianificano di continuare ad utilizzarlo nel prossimo futuro. Questo però richiede un po' di lavoro di configurazione. Infatti, poiché sia la configurazione che l'inizializzazione avvengono a runtime attraverso le API, è necessario scrivere script per garantire che l'installazione sia ripetibile.

Nei Radar precedenti, abbiamo menzionato che Unity è diventata la piattaforma di riferimento per lo sviluppo di applicazioni per VR e AR perché fornisce le astrazioni e il tooling di una piataforma matura, pur rimanendo più accessibile rispetto alla sua principale alternativa, l'Unreal Engine. Con la recente introduzione di ARKit per iOS e ARCore per Android, le due principali piattaforme mobile ora hanno SDK native e potenti per costruire applicazioni di realtà aumentata. Crediamo che molti team, specialmente quelli senza un'esperienza approfondita nella costruzioni di giochi, beneficeranno dall'utilizzo di un'astrazione come Unity, che è il motivo per cui la chiamiamo **UNITY AL DI LÀ DEL GAMING**. Questo consente agli sviluppatori poco familiari con la tecnologia di focalizzarsi su un unico SDK. È anche una soluzione all'enorme numero di dispositivi, specialmente lato Android, che non sono supportati dalle SDK native.

WECHAT, spesso visto come un' equivalente di WhatsApp, sta diventando la piattaforma di business per eccellenza in Cina. Non tutti lo sanno, ma WeChat è una delle piattaforme di pagamento online più popolari. Le piccole imprese conducono il loro e-commerce interamente su WeChat, grazie al CMS integrato e alla gestione degli utenti. Le grandi organizzazioni, invece, attraverso la funzione Account di servizio, possono interfacciare il proprio sistema interno ai propri dipendenti. Dato che oltre il 70% dei cinesi utilizza WeChat, è importante tenerlo in considerazione per le aziende che desiderano espandersi nel mercato cinese.

AZURE SERVICE FABRIC è una piattaforma per sistemi distribuiti pensata per microservizi e container. È paragonabile agli orchestratori di container come Kubernetes, ma funziona anche con semplici servizi vecchio stile. Può essere usato in una quantità impressionante di maniere diverse, a partire da semplici servizi realizzati in un qualsiasi linguaggio di vostra scelta, a container Docker oppure servizi costruiti a partire da un SDK. Da quando è stato pubblicato un paio di anni fa, è stato continuamente dotato di nuove feature, compreso il supporto per i container Linux. Kubernetes è stato l'orchestratore di container per antonomasia, ma Service Fabric è la scelta di default per le applicazioni .NET. Lo usiamo in alcuni progetti in ThoughtWorks e finora ne siamo rimasti soddisfatti.

CLOUD SPANNER è un servizio di database relazionale fully managed, con high availability e strong consistency, che non compromette i tempi di latenza. Per parecchio tempo Google ha lavorato su un database globalmente distribuito chiamato Spanner. Solo recentemente, il servizio è stato rilasciato al mondo esterno con il nome di Cloud Spanner. Si può scalare un'istanza di database da un nodo a migliaia di nodi distribuiti sul globo, senza preoccuparsi della consistenza dei dati. Cloud Spanner si basa su TrueTime, un orologio distribuito ad alta disponibilità, per fornire consistenza forte per read e

snapshot. Si può usare SQL standard per interrogare Cloud Spanner, ma per le scritture occorre usare la loro API RPC. Anche se non tutti i servizi richiedono un database distribuito su scala globale, il fatto che Cloud Spanner sia accessibile a tutti, è un grosso cambiamento nel modo in cui pensiamo ai database. Il suo design sta influenzando prodotti open source come CockroachDB.

Dopo una ricerca approfondita, R3, un giocatore importante nell'arena del blockchain, si è reso conto che il blockchain non rispondeva alle sue esigenze, e così ha creato **CORDA**. Corda è una piattaforma di ledger distribuito focalizzata sul campo finanziario. In R3 hanno una visione molto precisa e sanno che il loro problema richiede un approccio pragmatico alla tecnologia. Il che corrisponde alla nostra esperienza; le soluzioni blockchain correnti possono non essere una soluzione ragionevole per alcuni casi di business, a causa dei costi del mining e della inefficienza operazionale. Anche se l'esperienza di sviluppare su Corda finora non è stata delle più lisce, con API ancora instabili dopo la release v1.0, ci aspettiamo che lo spazio delle piattaforme di ledger distribuito maturi ulteriormente.

COSMOS DB è il servizio di database globalmente distribuito e multimodello di Microsoft, che è stato rilasciato al pubblico quest'anno. La maggior parte dei database NoSQL moderni offre livelli di consistenza configurabili, tuttavia Cosmos DB dà a questo concetto un'importanza primaria, e offre cinque diversi modelli di consistenza. Vale la pena notare che Cosmos DB supporta anche molteplici modelli: chiave-valore, document oriented, column e graph, che si mappano tutti sul suo data model interno, chiamato atom-record-sequence (ARS). Un aspetto interessante di Cosmos DB è che offre livelli di servizio dichiarati (SLA) sulla sua latenza, throughput, consistenza e disponibilità. Con questo ampio spettro di applicabilità, ha notevolmente alzato l'asticella per la competizione degli altri fornitori di cloud.

Parallelamente alla recente ondata di chatbot e piattaforme vocali, abbiamo osservato una proliferazione di strumenti e piattaforme che forniscono servizi per estrarre le intenzioni da contenuti testuali, e da gestori di flussi conversazionali ai quali è possibile agganciarsi. **DIALOGFLOW** (già API.ai), che è stata acquisita da Google, è una di delle offerte per la 'comprensione-del-linguaggio-naturale as a service' che fa concorrenza a wit.ai e Amazon Lex e ad altre offerte in questo settore.

Il mondo dello sviluppo software sta convergendo su Kubernetes come strumento di orchestrazione per i container; tuttavia, gestire cluster di Kubernetes resta una cosa complicata. **GKE** (Google Container Engine) è un servizio Kubernetes gestito per il deployment di soluzioni containerizzate, che allevia la difficoltà di manutenere cluster Kubernetes. L'esperienza dei nostri team con GKE è stata buona, perché la piattaforma fa gran parte del lavoro: applicare patch di sicurezza, monitorare e auto-riparare i nodi, gestire il networking multicluster e multiregione. Secondo la nostra esperienza, l'approccio di Google che espone le capacità della piattaforma mettendo le API al primo posto, e l'uso di tecnologie standard come OAuth per l'autorizzazione, migliora l'esperienza degli sviluppatori. E' importante ricordare che GKE è in rapido sviluppo, il che, nonostante i grandi sforzi dei suoi sviluppatori che cercano di nascondere i cambiamenti dietro le quinte, ci ha temporaneamente impattati in passato. Ci aspettiamo miglioramenti continui nel campo dell'Infrastructure as code con Terraform on GKE e strumenti simili.

I language server forniscono feature come autocompletamento, la ricerca delle righe che chiamano una funzione e il refactoring in un'API che permette a qualsiasi editor di lavorare con l'albero sintattico del linguaggio.

(Language Server Protocol)

KAFKA STREAMS è una libreria leggera per la costruzione di applicazioni streaming. Essa è stata progettata con l'obiettivo di semplificare l'elaborazione stream, al punto di renderla facilmente accessibile come principale modello di programmazione per servizi asincroni. Kafka Streams può essere una buona alternativa in scenari in cui si voglia applicare un modello di elaborazione stream al vostro problema, senza farsi carico della complessità di gestire un cluster (cosa di solito richiesta dai framework più articolati). Nuovi sviluppi includono una elaborazione 'exactly once' in un cluster Kafka. Ciò è stato raggiunto introducendo l'idempotenza nei produttori Kafka, consentendo scritture atomiche attraverso partizioni multiple ed usando la nuova Transactions API.

Gran parte della potenza degli IDE sofisticati proviene dalla loro capacità di trasformare un programma in un albero sintattico astratto (abstract syntax tree, AST), per poi usare quell'AST per manipolare e analizzare il programma. Si realizzano così feature come l'autocompletamento, la ricerca delle righe che chiamano una funzione e il refactoring. I server di linguaggio mettono questa capacità in un processo che permette a qualsiasi editor di accedere a un API per manipolare l'AST. Microsoft ha diretto la creazione del **LANGUAGE SERVER PROTOCOL** (LSP), ricavato dai loro progetti OmniSharp e TypeScript Server. Qualsiasi editor che usi questo protocollo può lavorare con qualsiasi linguaggio che disponga di un server LSP. Questo significa che possiamo continuare ad usare il nostro editor preferito, senza rinunciare alle capacità di editing evolute, per la felicità dei nostri appassionati di Emacs.

LORAWAN è una rete geografica a bassa potenza, progettata per bassi consumi e comunicazione su grandi distanze usando una bassa bitrate. Essa si occupa della comunicazione fra dispositivi e gateway, che possono inoltrare i dati a, per esempio, applicazioni o server. Un utilizzo tipico è per insiemi distribuiti di sensori, o per dispositivi Internet of Things (IoT), per i quali una lunga durata della batteria ed una comunicazione ad ampio raggio sono d'obbligo. LoRaWAN affronta due dei problemi chiave che sorgono nel tentativo di usare normali Wi-Fi per tali applicazioni: raggio e consumo di potenza. Ci sono varie implementazioni, degna di nota è The Things Network, una implementazione open source gratuita.

Spostandosi da un uso sperimentale ad un impiego in produzione, abbiamo bisogno di un modo affidablie per ospitare ed installare i modelli a cui si può accedere da remoto e che possono scalare in funzione del numero di clienti.

(TensorFlow Serving)

MAPD è un database analitico colonnare in-memoria, con supporto per SQL, che esegue su GPU. Abbiamo dibattuto se il carico di lavoro di un database sia più limitato da operazioni I/O, piuttosto che limitato dall'uso di CPU, ma ci sono casi in cui il parallelismo offerto da una GPU, combinato alla banda larga consentita da una VRAM, possa essere piuttosto utile. MapD gestisce in modo trasparente in VRAM i dati maggiormente usati (come colonne coinvolte in group-by, filtti, calcoli e condizioni di join) e memorizza il dati rimanenti nella memoria principale. Con questa configurazione della gestione della memoria, MapD raggiunge prestazioni significative senza il bisogno di indici. Sebbene ci siano altri produttori di database basati su GPU, MapD sta dominando il comparto con un recente rilascio open source del core del proprio database, e con la GPU Open Analytics Initiative. Se il vostro carico analitico è computazionalmente pesante, se potete sfruttare il parallelismo della GPU e se potete metterlo nella memoria principale, vi raccomandiamo di valutare MapD.

We like simple tools that solve one problem really well, and **NETLIFY** fits this description nicely. You can create static website content, check it into GitHub and then quickly and easily get your site live and available. There is a CLI available to control the process; content delivery networks (CDNs) are supported; it can work alongside tools such as Grunt; and, most importantly, Netlify supports HTTPS.

I modelli di machine learning iniziano ad insinuarsi nelle applicazioni aziendali usate quotidianamente. Quando sono disponibili sufficienti dati per l'apprendimento, questi algoritmi possono affrontare problemi che in passato avrebbero potuto richiedere complessi modelli statistici o euristici. Spostandosi da un uso sperimentale ad un impiego in produzione, abbiamo bisogno di un modo affidablie per ospitare ed installare i modelli a cui si può accedere da remoto e che possono scalare in funzione del numero di clienti. **TENSORFLOW SERVING** affronta parte di questo problema esponendo un'interfaccia remota gRPC ad un modello esportato; ciò consente ad un modello addestrato di essere installato in varie modalità. TensorFlow Serving accetta anche uno stream di modelli per incorporare continuamente aggiornamenti di apprendimento, e i suoi autori mantengono un file Docker per semplificare il processo di installazione. Presumibilmente, la scelta di gRPC deve essere consistente con il modello di esecuzione di TensorFlow; in generale, comunque, vi mettiamo in guardia dai protocolli che richiedono la generazione di codice e binding nativo.

Microsoft è entrata nel settore dei container con **WINDOWS CONTAINERS**. Al momento di questa pubblicazione, Microsoft fornisce due immagini del sistema operativo Windows come container Docker, Windows Server 2016 Server Core e Windows Server 2016 Nano Server. Sebbene ci siano margini di miglioramento per Windows Containers, per esempio nella riduzione della notevole dimensione dell'immagine, e nell'arricchimento dell'ecosistema di supporto e documentazione, i nostri team hanno iniziato ad usarlo in scenari in cui altri container stavano lavorando con successo, come agenti di compilazione.

Rimaniamo preoccupati riguardo all'implementazione nel middleware di logica e processi di orchestrazione, specialmente dove essa richieda forti competenze e tool complessi, creando al tempo stesso singoli punti di scalabilità e controllo. Nel mercato estremamente competitivo delle API gateway i venditori stanno continuando questa tendenza, aggiungendo funzionalità nel tentativo di differenziare i propri prodotti. Questo porta a **API GATEWAY TROPPO AMBIZIOSI** le cui funzionalità, che si aggiungono a quello che è essenzialmente un reverse proxy, incoraggia progetti che restano difficili da testare ed installare. Gli API gateways sono utili per risolvere specifiche problematiche, come l'autenticazione e la limitazione del numero di accessi concorrenti, ma tutta la logica di dominio dovrebbe risiedere in applicazioni o servizi.

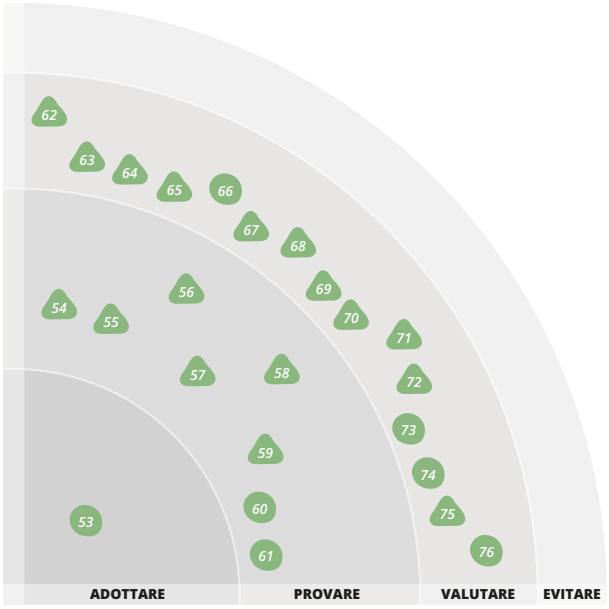
STRUMENTI

ADOTTARE
53. fastlane

PROVARE
54. Buildkite *NEW*
55. CircleCI *NEW*
56. gopass *NEW*
57. Headless Chrome per test di front-end *NEW*
58. jsoniter *NEW*
59. Prometheus
60. Scikit-learn
61. Serverless Framework

VALUTARE
62. Apex *NEW*
63. assertj-swagger *NEW*
64. Cypress *NEW*
65. Flow *NEW*
66. InSpec
67. Jupyter *NEW*
68. Kong API Gateway *NEW*
69. kops *NEW*
70. Lighthouse *NEW*
71. Rendertron *NEW*
72. Sonobuoy *NEW*
73. spaCy
74. Spinnaker
75. Spring Cloud Contract *NEW*
76. Yarn

EVITARE



I nostri team apprezzano veramente molto lo strumento CI/CD **BUILDKITE** per la sua semplicità e rapidità di confugurazione. Con Buildkite, fornite le vostre stesse macchine per l'esecuzione della compilazione — on premise o nel cloud — installando una leggera applicazione agent per connettere la compilazione al servizio in host. In molti casi, avere questo livello di controllo sulla configurazione dei propri agenti di compilazione è un vantaggio rispetto all'utilizzo di agenti as a service.

CIRCLECI è un sistema di continuous integration offerto come SaaS e on-premise. CircleCI è lo strumento CI standard SaaS per molti dei nostri team di sviluppo, quando si cerca uno strumento facile da configurare per realizzare pipeline di build e deployment. CircleCI 2.0 supporta i workflow, con flussi di fan-in e fan-out e approvazioni manuali. Supporta anche lo sviluppo mobile. Permette agli sviluppatori di eseguire le pipeline localmente e si integra con facilità con Slack e altri sistemi di notifica. Vi raccomandiamo di dare un'occhiata da vicino alle pratiche di sicurezza per CircleCI, come fareste per qualsiasi altro prodotto SaaS contenente asset proprietari della vostra azienda.

Un discendente di pass, gopass aggiunge funzionalità quali: supporto per la gestione destinatario e memorizzazione di password multiple in un singolo albero; ricerca interattiva; supporto per one-time password basate sul tempo; e memorizzazione di dati binari.

(gopass)

GOPASS è un gestore di password per team, basato su GPG e Git. È un discendente di pass ed aggiunge funzionalità quali: supporto per la gestione destinatario, supporto per la memorizzazione di password multiple in un singolo albero, ricerca interattiva, supporto per one-time password basate sul tempo (TOTP), e memorizzazione di dati binari. La migrazione del proprio archivio "pass" è abbastanza semplice, perché "Gopass" è largamente compatibile con i formati che usa "pass". Ciò significa anche che l'integrazione in workflow di provisioning può essere conseguita con una singola chiamata ad un segreto memorizzato.

Fin dalla metà del 2017, gli utenti di Chrome hanno la possibilità di eseguire il browser in modalità "headless". Questa feature è ideale per eseguire test automatici di front-end, senza pagare l'overhead dell'animazione delle azioni sullo schermo. In precedenza, questa cosa si faceva principalmente con PhantomJS, ma Headless Chrome sta rapidamente sostituendo il prodotto basato su WebKit. I test in Headless Chrome dovrebbero girare molto più velocemente, e comportarsi proprio come nel browser vero, anche se i nostri team hanno scoperto che richiede più memoria di PhantomJS. Con tutti questi vantaggi, **HEADLESS CHROME PER TEST DI FRONT-END** diventerà probabilmente uno standard di riferimento.

Se state cercando un encoder/decoder JSON con elevate prestazioni in Go e Java, date uno sguardo alla libreria open source **JSONITER**. ThLa libreria è compatibile con lo il pacchetto JSON standard in Go.

Abbiamo osservato sia miglioramenti continui che un incremento nell'adozione di Prometheus, lo strumento di monitoraggio e database di serie temporali, originariamente sviluppato da Soundcloud.

(Prometheus)

Abbiamo osservato sia miglioramenti continui che un incremento nell'adozione di **PROMETHEUS**, lo strumento di monitoraggio e database di serie temporali, originariamente sviluppato da Soundcloud. Prometheus supporta principalmente un modello HTTP basato su pull ma può anche supportare allarmi, caratteristica che lo rende una parte attiva della vostra strumentazione operazionale. Al momento in cui scriviamo, Prometheus 2.0 è in pre-release, e continua ad evolvere. Gli sviluppatori di Prometheus hanno concentrato i propri sforzi sul nucleo dei database di serie temporali e sulla varietà di metriche disponibili. Grafana è diventato lo strumento di visualizzazione d'elezione per gli utenti di Prometheus ed il supporto per Grafana è incluso con lo strumento. I nostri team trovano che il monitoraggio offerto da Prometheus completi elegantemente le capacità di indicizzazione e ricerca di un Elastic Stack.

APEX è uno strumento per creare, installare e gestire facilmente funzioni AWS Lambda. Con Apex, si possono scrivere funzioni in linguaggi che non sono ancora supportati in maniera nativa su AWS, compresi Golang, Rust e altri. Questo è reso possibile da un adapter scritto in Node.js, che crea un processo figlio che elabora gli eventi attraverso standard input e standard output. Apex ha molte features che migliorano l'esperienza dello svliuppatore; ci piace in particolare la possibilità di testare le funzioni localmente, e di eseguire un dry run delle modifiche prima che vengaono applicate alle risorse AWS.

ASSERTJ-SWAGGER è una libreria di AssertJ che permette di validare la corrispondenza dell'implementazione di una API rispetto al contratto della sua specifica. I nostri team usano assertj-swagger per accorgersi di quando un'implementazione cambia senza aggiornare la sua specifica Swagger, o di quando ci si dimentica di pubblicare la documentazione aggiornata.

La correzione dei fallimenti dei test end-to-end eseguiti su CI può essere un'esperienza frustrante, specialmente in modalità "headless". **CYPRESS** è uno strumento utile che semplifica agli sviluppatori la costruzione di test end-to-end e registra tutti i passi del test in un video MP4. Per risolvere un problema gli sviluppatori, invece che riprodurlo in modalità "headless", possono risolverlo guardando il video di test. Cypress non è solo un framework di test, ma anche una potente piattaforma. Al momento, abbiamo integrato la sua CLI con il nostro CI "headless" per i nostri prodotti.

FLOW è un controllore statico di tipo per JavaScript che vi consente di aggiungere incrementalmente il controllo di tipo al codice. Diversamente da Typescript, che è un linguaggio diverso, Flow può essere aggiunto incrementalmente ad un corpo di codice JavaScript pre-esistente, essendo supportate la 5a, 6a e 7a edizione di ECMAScript. Suggeriamo di aggiungere Flow alla vostra pipeline di continuous integration, partendo dal codice che vi preoccupa maggiormente. Flow aggiunge chiarezza al codice, incrementa l'affidabilità del refactoring ed intercetta presto i bug relativi ai tipi durante la compilazione.

Durante gli ultimi due anni, abbiamo notato un'ascesa decisa nella popolarità delle analytics notebook. Queste ultime sono applicazioni ispirate a Mathematica che combinano testo, visualizzazione e codice in documentazione vivente computazionale. In una precedente edizione abbiamo menzionato GorillaREPL, una variante Clojure di tali applicazioni. Ma l'aumento di interesse verso il machine learning, insieme all'emergere di Python come linguaggio di programmazione preferito dai praticanti in questo campo, ha focalizzato un'attenzione particolare sulle Python notebooks, fra le quali **JUPYTER** sembra stia guadagnando una maggior popolarità fra i team di ThoughtWorks.

Kong è un open source API gateway realizzato e sponsorizzato da Mashape, che fornisce anche un servizio enterprise che integra Kong con i loro strumenti proprietari per API analytics e con un portale per gli sviluppatori. Kong può essere configurato in varie maniere, come un API gateway esposto a Internet, oppure come API proxy interno. Si fonda su OpenResty, che grazie ai suoi moduli Nginx, costituisce una base robusta e performante, e si può estendere grazie ai plugin scritti in Lua. Kong può usare PostgreSQL per i deployment su una singola regione, oppure Cassandra per configurazioni multiregione. I nostri sviluppatori si sono avvalsi delle sue performance elevate, del suo approccio che mette le API al primo posto (il che permette di automatizzare la configurazione) e della facilità di deployment come container. **KONG API GATEWAY**, al contrario degli API gateway troppo ambiziosi, ha meno feature, ma implementa le cose essenziali che un API gateway deve avere, come il controllo del traffico, la security, il logging, il monitoraggio e l'autenticazione. Non vediamo l'ora di provare Kong in una configurazione sidecar in un prossimo futuro.

KOPS è uno strumento a linea di comando per la creazione e gestione in produzione di cluste Kubernetes ad alta disponibilità. Inizialmente pensato per AWS, ora ha un supporto sperimentale per altri fornitori. Con kops si è operativi velocemente e, sebbene alcune funzionalità (come i rolling upgrade) debbano ancora essere sviluppate completamente, siamo rimasti impressionati dalla comunità.

Un problema perenne per le applicazioni web pesantemente basate su JavaScript è come rendere disponibili ai motori di ricerca le parti dinamiche delle pagine. I bot che non renderizzano JavaScript possono essere rediretti a un server Rendertron che esegue il rendering per loro.

(Rendertron)

LIGHTHOUSE è uno strumento scritto da Google per valutare l'aderenza delle applicazioni web agli standard Progressive Web App. Il rilascio di Lighthouse 2.0 avvenuto quest'anno aggiunge le metriche di performance e le verifiche di accessibilità agli strumenti base preesistenti. Queste funzionalità addizionali sono state recentemente incorporate negli strumenti standard per gli sviluppatori di Chrome, sotto il tab "audit". Lighthouse 2.0 beneficia anch'esso della modalità headless di Chrome. Lighthouse è un'alternativa a Pa11y e a strumenti simili, per eseguire test di accessibilità in una pipeline di continuous integration, dato che lo strumento può essere eseguito anche dalla riga di comando, oppure da solo come una applicazione Node.js.

Un problema perenne per le applicazioni web pesantemente basate su JavaScript è come rendere disponibili ai motori di ricerca le parti dinamiche delle pagine. Storicamente, gli sviluppatori hanno escogitato una varietà di trucchi, incluso il rendering server-side con React, servizi esterni o contenuti pre-renderizzati. Ora la nuova modalità headless di Google Chrome aggiunge un nuovo 'trucco' all'armamentario: **RENDERTRON**, una soluzione di rendering con Chrome in modalità headless. Rendertron incapsula una istanza di Chrome headless all'interno di un container Docker, pronto per l'installazione come server HTTP standalone. I bot che non renderizzano JavaScript possono essere rediretti a questo server che esegue il rendering per loro. Sebbene gli sviluppatori possano sempre installarsi da soli i proxy Chrome headless con gli annessi server di instradamento, Rendertron semplifica la configurazione del processo di installazione, e fornisce codice middleware di esempio per rilevare ed instradare i bot.

SONOBUOY è uno strumento di diagnostica per l'esecuzione di test di conformità, in modo non distruttivo, su qualsiasi cluster Kubernetes. Il team di Heptio, che è stato fondato da due dei creatori del progetto Kubernetes, ha costruito questo strumento per garantire che grandi array di distribuzioni e configurazioni Kubernetes siano conformi alle best practice, mentre seguono la standardizzazione open source per l'interoperabilità dei cluster. Stiamo sperimentando Sonobuoy per l'esecuzione di parte della nostra pipeline di compilazione descritta come infrastructure as code ed anche per il monitoraggio continuo delle nostre installazionoi Kubernetes, per validare il comportamento e la salute dell'intero cluster.

Se state implementando servizi Java usando il framework Spring potreste voler considerare **SPRING CLOUD CONTRACT** per il test di contratti consumer-driven. Questo strumento supporta attualmente la verifica delle chiamate del client e del rispetto del contratto da parte del server. In confronto a Pact, che è un insieme di strumenti open source per il test di contratti consumer-driven, in Spring Cloud Contract manca la distribuzione dei contratti e il supporto per altri linguaggi di programmazione. Ad ogni modo si integra bene con l'ecosistema di Spring, per esempio con il routing di messaggi con Spring Integration.

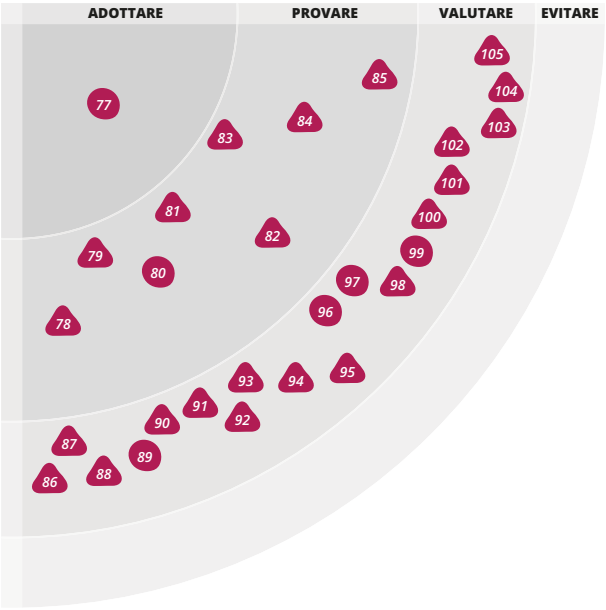
LINGUAGGI & FRAMEWORK

ADOTTARE
77. Python 3

PROVARE
78. Angular
79. AssertJ *NEW*
80. Avro
81. CSS Grid Layout *NEW*
82. CSS Modules *NEW*
83. Jest *NEW*
84. Kotlin
85. Spring Cloud

VALUTARE
86. Android Architecture Components *NEW*
87. ARKit/ARCore *NEW*
88. Atlas and BeeHive *NEW*
89. Caffe
90. Clara rules *NEW*
91. CSS-in-JS *NEW*
92. Digdag *NEW*
93. Druid *NEW*
94. ECharts *NEW*
95. Gobot *NEW*
96. Instana
97. Keras
98. LeakCanary *NEW*
99. PostCSS
100. PyTorch *NEW*
101. single-spa *NEW*
102. Solidity *NEW*
103. TensorFlow Mobile *NEW*
104. Truffle *NEW*
105. Weex *NEW*

EVITARE



Nelle edizioni precedenti del Radar, abbiamo esitato a dare una forte raccomandazione per **ANGULAR**, perché era un framework nuovo e nel complesso non particolarmente interessante, che condivideva con AngularJS solamente il nome (mentre AngularJS era stato molto apprezzato da noi in passato). Da allora Angular, ormai alla versione 5, è continuamente migliorato, restando sempre compatibile con le versioni precedenti. Diversi nostri team hanno applicazioni Angular in produzione e, secondo loro, è valido. Per questo, in questo Radar mettiamo Angular nell'anello "provare", per indicare che alcuni dei nostri team lo considerano ormai una buona scelta. La maggior parte dei nostri team, tuttavia, continua a preferire React, Vue o Ember ad Angular.

ASSERTJ è una libreria Java che fornisce una fluent interface per le asserzioni, che agevola la comunicazione di intenti all'interno di codice di test. AssertJ produce messaggi di errore leggibili, asserzioni soft, ed un supporto migliorato alle collezioni ed eccezioni. Stiamo osservando che vari team lo stanno adottando al posto della combinazione JUnit-Hamcrest.

CSS Grid Layout è un sistema basato su griglia bidimensionale che fornisce un meccanismo per dividere lo spazio disponibile per l'impaginazione in colonne e righe, usando un insieme di comportamenti ben definiti per il ridimensionamento.

(CSS Grid Layout)

CSS è la soluzione di riferimento per impaginare pagine web, anche se non forniva un supporto esplicito per la creazione di layout. Flexbox aiuta con i layout

monodimensionali più semplici, ma per i layout più complessi gli sviluppatori cercano di solito librerie e toolkit. **CSS GRID LAYOUT** è un sistema basato su griglia bidimensionale che fornisce un meccanismo per dividere lo spazio disponibile per l'impaginazione in colonne e righe, usando un insieme di comportamenti ben definiti per il ridimensionamento. Grid non richiede alcuna libreria a si affianca bene a Flexbox e ad altri elementi di visualizzazione CSS. Tuttavia, poiché esso è solo supportato parzialmente, in IE11, non è utile per gli utenti che dipendono ancora dal browser Microsoft su Windows 7.

La grandi basi di codice CSS richiedono schemi complessi di denominazione per evitare di conflitti di nomi nello spazio dei nomi globale. I **MODULI CSS** affrontano questi problemi creando uno scope locale per tutti i nomi di classi che appaiono in un singolo file CSS. Questo file viene importato in un modulo JavaScript, in cui le classi CSS sono identificate come stringhe. Quindi, nella pipeline di build (Webpack, Browserify, ecc.), i nomi delle classi vengono sostituiti con stringhe univoche generate automaticamente. Questo rappresenta un cambiamento significativo delle responsabilità. In precedenza, per evitare i conflitti di denominazione delle classi, una persona doveva gestire lo spazio dei nomi globali; ora quella responsabilità dipende dal build tooling. Un piccolo inconveniente che abbiamo incontrato con i moduli CSS è che i test funzionali sono solitamente fuori dall'ambito locale e quindi non possono fare riferimento ai nomi di classe definiti nel modulo CSS. Si consiglia di utilizzare al loro posto gli attributi ID o i gli attributi prefissati con "data-".

Jest is a 'zero configuration' front-end testing tool with out-of-the-box features such as mocking and code coverage, targeted at React and other JavaScript frameworks.

(Jest)

I nostri team sono deliziati dai risultati ottenuti con **JEST** per il test front-end. Esso fornisce un'esperienza 'zero-configurazione' ed ha funzionalità out-of-the-box come mocking e copertura di codice. Potete applicare questo framework di test non solo alle applicazioni React applications, ma anche ad altri framework JavaScript. Una delle funzionalità di Jest spesso pubblicizzate è il test UI snapshot. Il test snapshot sarebbe una buona aggiunta allo strato superiore della test pyramid, ma ricordate, i test di unità restano la base più solida.

L'annuncio di un supporto Android di prima classe ha dato una spinta aggiuntiva al rapido progresso del linguaggio **KOTLIN** e stiamo seguendo da vicino l'evoluzione di Kotlin/Native — la capacità basata su LLVM di compilare ottenendo eseguibili nativi. Null safety, classi di dati e la semplicità nel creare DSL sono alcuni dei benefici che ci sono piaciuti, insieme alla libreria Anko per lo sviluppo Android. Nonostante gli svantaggi di lenta compilazione iniziale e di dipendenza dall'IDE IntelliJ per un supporto di prima classe, raccomandiamo di dare un'opportunità a questo linguaggio moderno, fresco e conciso.

SPRING CLOUD continua ad evolvere ed aggiunge nuove interessanti funzionalità. Il supporto per il binding a Kafka Streams, per esempio, nel progetto spring-cloud-streams rende relativamente semplice costruire applicazioni message driven con connettori per Kafka ev RabbitMQ. I team che lo hanno usato apprezzano la semplicità che questo porta nell'usare infrastrutture a volte complesse, come ZooKeeper, e nel supporto per problemi comuni che affrontiamo quando costruiamo sistemi distribuiti, tracciando, per esempio, con spring-cloud-sleuth. Valgono le consuete precauzioni, ma lo stiamo utilizzando con successo in molti progetti.

Storicamente, la documentazione degli esempi Android di Google mancava di architettura e struttura. La situazione è cambiata con il rilascio di **ANDROID ARCHITECTURE COMPONENTS**, un insieme di librerie, che esprimono una visione particolare e che aiutano gli sviluppatori a creare applicazioni Android con architetture migliori. Esse affrontano aspetti mal trattati dallo sviluppo Android: gestione dei cicli di vita; paginazione; database SQLite; e la persistenza dei dati dopo i cambi di configurazione. Queste librerie non devono necessariamente essere usate insieme; potete prendere quelle che vi servono maggiormente ed integrarle nei vostri progetti.

Abbiamo visto un incremento di attività nella realtà aumentata per i dispositivi mobili, per la gran parte alimentata da **ARKIT AND ARCORE**, le librerie AR native usate rispettivamente da Apple e Google. Queste librerie portano le tecnologie mobile per la

realtà aumentata nel mainstream. Tuttavia, la sfida per le aziende è di trovare un campo di applicazione che vada al di là della curiosità per fornire soluzioni che genuinamente migliorino l'esperienza utente.

La strategia di realizzare più di una app è davvero controversa, soprattutto in un'epoca in cui sempre meno utenti scaricano nuove app. Per evitare di introdurre una nuova app e poi angustiarsi per l'esiguo numero dei download, le organizzazioni multiteam devono consegnare funzionalità per mezzo di una singola app che sia già installata da tanti utenti, e questo crea una difficoltà architetturale. **ATLAS E BEEHIVE** sono rispettivamente soluzioni di modularizzazione per applicazioni Android e iOS. Atlas e BeeHive consentono alle organizzazioni multiteam di lavorare su moduli isolati fisicamente, e poi riassemblare o caricare dinamicamente questi moduli in una app di facciata. Entrambi sono progetti open source di Alibaba, dato che Alibaba ha incontrato proprio questo problema del numero di download che diminuisce e della difficoltà architetturale della singola app..

La nostra prima regola quando si tratta di scegliere un motore di regole, normalmente è: non hai bisogno di un motore di regole. Abbiamo visto troppa gente legarsi a sistemi di regole black-box, inflessibili e difficili da testare, per ragioni artificiose; quando invece scrivere codice ad-hoc sarebbe stata una soluzione migliore. Detto questo, abbiamo avuto successo con **CLARA RULES** in scenari in cui usare un motore di regole ha senso. Ci piace il fatto che usi semplice codice Clojure per esprimere e valutare le regole, il che significa che sia possibile rifattorizzarle, testarle e versionarle come sorgenti. Non corre dietro all'illusione che le persone di business debbano scrivere le regole direttamente, ma incoraggia la collaborazione fra gli esperti di business e gli sviluppatori.

CSS IN JS iè una tecnica di scrittura di stili CSS in linguaggio JavaScript, che incoraggia un pattern che consiste nel mettere il testo che descrive lo stile insieme al componente JavaScript a cui si applica, collocando insieme logica e stile. I nuovi prodotti, fra cui JSS, emotion e styled-components si basano su strumenti che traducono il codice CSS-in-JS in stylesheet CSS separati, adatti all'uso da parte del browser. Questo è un approccio di seconda generazione alla scrittura di CSS in JavaScript, e al contrario degli approcci precedenti, non si basa sugli stili in-line. In questo modo può supportare tutte le caratteristiche di CSS, condividendo il CSS con l'ecosistema npm e utilizzando i componenti su più di una piattaforma. I nostri team hanno trovato che gli styled-components coesistono bene con i framework basati sui componenti, come React, e facilitano i test di unità del CSS con jest-styled-components. Questo spazio è nuovo e cambia rapidamente; l'approccio richiede un po' di lavoro per debuggare manualmente nel browser i nomi delle classi che sono generati automaticamente, e potrebbe non essere adatto ad alcuni progetti in cui l'architettura di front-end non si presta al riuso di componenti e richiede una stilizzazione globale.

Gli Android Architecture Components sono un insieme di librerie opinionate che aiutano gli sviluppatori a creare applicazioni Android con un'architettura migliore.

(Android Architecture Components)

DIGDAG è uno strumento per compilare, eseguire, schedulare e monitorare pipeline di dati complessi nel cloud. Potete definire queste pipeline in YAML usando il ricco insieme di operatori predefiniti oppure costruendo le vostre tramite le API. Digdag ha molte delle funzionalità tipiche delle soluzioni di pipeline di dati: gestione delle dipendenze, workflow modulare per promuovere il riuso, gestione sicura dei segreti e supporto multilingua. La funzionalità della quale siamo più entusiasti è il supporto al polycloud, che vi consente di spostare ed unire dati attraverso AWS RedShift, S3, e Google BigQuery. Visto che sempre più fornitori di cloud offriranno soluzioni competitive di elaborazione dati, pensiamo che Digdag (e strumenti simili) saranno utili per individuare l'opzione migliore per svolgere un certo compito.

DRUID è un pool di connessioni JDBC con ricche funzionalità di monitoraggio. Ha un parser SQL built-in, che fornisce un monitoraggio semantico degli statement SQL eseguiti nel database. Statement SQL iniettati o sospetti sono bloccati e loggati dallo strato JDBC. Inoltre, le query possono essere fuse in base alla loro semantica. Questo è un progetto open source di Alibaba, e riflette le lezioni che Alibaba ha imparato lavorando sui propri database.

ECHARTS è una libreria per disegnare grafici con un ricco supporto per vari tipi di grafici ed interazioni. Poiché ECharts è basato interamente sulla Canvas API, ha prestazioni incredibili perfino quando lavora con oltre 100.000 data point, ed è anche ottimizzato per l'utilizzo mobile. Insieme al suo progetto fratello, ECharts-X, può supportare la stampa 3D. ECharts è un progetto open source di Baidu.

La capacità del linguaggio di programmazione Go di compilare in linguaggio macchina ha sollevato interesse fra gli sviluppatori che utilizzano il linguaggio per sistemi embedded. **GOBOT** è un framework per robotica, computazione fisica, e Internet of Things, scritto nel linguaggio di programmazione Go e supporta una varietà di piattaforme. Abbiamo usato il framework per progetti robotici sperimentali in cui la risposta in tempo reale non era un requisito, ed abbiamo creato dei piloti software open source con Gobot. Le API HTTP di Gobot consentono una semplice integrazione hardware con dispositivi mobili per creare applicazioni più ricche.

I nostri team che sviluppano applicazioni mobile sono entusiasti di **LEAKCANARY**, uno strumento per rilevare fastidiose memory leak (perdite di memoria) in Android e Java. LeakCanary e' semplice da installare e fornisce notifiche con una chiara traccia (trace-back) della causa della perdita. Aggiungere questo strumento al vostro toolkit vi farà risparmiare noiose ore passate ad eliminare gli errori di memoria (out-of-memory) su più dispositivi.

PYTORCH è la riscrittura completa da Lua a Python del framework di machine learning Torch. Sebbene sia abbastanza nuovo ed immaturo rispetto a Tensorflow, i programmatori trovano PyTorch molto più semplice da usare. Grazie al suo orientamento agli oggetti ed alla sua implementazione nativa in Python, i modelli possono essere espressi più chiaramente e succintamente, e possono essere debuggati durante l'esecuzione. Sebbene molti di questi framework siano emersi di recente, PyTorch ha il supporto di Facebook e di una vasta gamma di organizzazioni affiliate, inclusa NVIDIA, che dovrebbe garantire un supporto continuo all'architettura CUDA. I team di ThoughtWorks trovano che PyTorch sia utile per sperimentare e sviluppare modelli ma si affidano ancora alle prestazioni di TensorFlow per addestramento e classificazione a livello di produzione.

SINGLE-SPA è un meta-framework JavaScript che ci consente di costruire micro frontends utilizzando differenti framework che possono coesistere in una singola applicazione. In generale, non raccomandiamo l'utilizzo di più di un framework per una applicazione, ma ci sono situazioni in cui non possiamo evitarlo. Per esempio, single-spa può essere piuttosto utile quando lavorate con una applicazione legacy e desiderate sperimentare sviluppando una nuova funzionalità, usando o una nuova versione del framework esistente oppure usando una versione completamente diversa. Considerato il breve periodo di vita di molti framework JavaScript, osserviamo la necessità di una soluzione che consenta futuri cambi di framework, e sperimentazioni localizzate, senza compromettere l'intera applicazione. single-spa sembra essere un buon inizio in questa direzione.

La programmazione di smart contracts richiede un linguaggio più espressivo di un sistema di scripting per transazioni. **SOLIDITY** è il più popolare fra i nuovi linguaggi di programmazione progettati per i contratti smart. Solidity è un linguaggio orientato ai contratti e staticamente tipizzato la cui sintassi è simile a JavaScript. Esso fornisce astrazioni per la scrittura di business logic auto-rinforzante nei contratti smart. L'ecosistema attorno a Solidity sta crescendo rapidamente. Oggigiorno, Solidity è la prima scelta sulla piattaforma Ethereum Data la natura immutabile dei contratti smart una volta rilasciati, è sottointeso che test rigosi e verifiche delle dipendenze sono di vitale importanza.

TENSORFLOW MOBILE rende possibile agli sviluppatori l'inclusione di un'ampia gamma di tecniche di comprensione e classificazione nelle proprie applicazioni iOS o Android. Ciò è particolarmente utile considerata la gamma di sensori disponibili sui dispositivi mobili. Modelli TensorFlow pre-addestrati possono essere caricati in una applicazione mobile ed applicati agli input come immagini di video in diretta, testo o parlato. I dispositivi mobili costituiscono una piattaforma sorprendentemente adatta per l'implementazione di tali modelli computazionali. I modelli TensorFlow sono esportati e caricati come file protobuf, il che può presentare alcuni inconvenienti per gli implementatori. Il formato dei binari protobuf può rendere difficile esaminare i modelli e richiede di installare la corretta versione di libreria protobuf nell'applicazione. Ma l'esecuzione locale del modello

offre un'alternativa allettante al TensorFlow Serving senza i ritardi dovuti ad esecuzioni remote.

Abbiamo avuto successo con Clara rules in scenari in cui un motore di regole ha senso. Ci piace che usi semplice codice Clojure per esprimere e valutare le regole, il che rende possibile rifattorizzarle, testarle e versionarle.

(Clara rules)

TRUFFLE è un framework di sviluppo che porta una moderna esperienza di sviluppo web nella piattaforma Ethereum. Si inserisce nell'ambito della compilazione di smart contract, del linking di libreria e deployment, ed anche della gestione degli artefatti in reti blockchain. Una delle ragioni per le quali amiamo Truffle è che incoraggia le persone a scrivere test per i propri smart contract. Dovete veramente prendere i test in seria considerazione perché la programmazione smart contract è spesso legata al denaro. Con il suo framework di test built-in e l'integrazione con TestRPC, Truffle rende possibile la scrittura di smart contract nello stile TDD. Ci aspettiamo di vedere più tecnologie simili a Truffle per promuovere la continuous integration nell'area blockchain.

WEEX è un framework per costruire app mobile multipiattaforma usando la sintassi dei componenti di Vue.js component syntax. For those who prefer the simplicity of Vue.js, Weex is a viable option for native mobile apps, but it also works very well for more complicated apps. Per chi preferisce la semplicità di Vue.js, Weex è un'opzione possibile per mobile app native, ma funziona molto bene anche per app più complicate. Su questo framework notiamo molti successi relativamente ad app abbastanza complicate, incluse TMall and Taobao, due delle più popoòari mobile app in Cina. Weex è stato sviluppato da Alibaba, ed ora è un Apache incubator project.

Sii il primo a sapere quando il Technology Radar viene pubblicato, e mantieniti aggiornato con seminari e contenuti esclusivi.

ABBONATI ORA

thght.works/Sub-EN

ThoughtWorks®

ThoughtWorks è una società di consulenza tecnologica e una comunità di individui appassionati e guidati da uno scopo. Aiutiamo i nostri clienti a mettere la tecnologia al centro del loro business, e insieme creiamo il software più importante per loro. Siamo votati al cambiamento sociale: la nostra missione è di migliorare l'umanità con il software, e siamo partner di molte organizzazioni che si impegnano nella stessa direzione.

Fondata oltre 20 anni fa, ThoughtWorks è cresciuta fino a diventare un'azienda di più di 4500 persone, compresa una divisione prodotti che realizza strumenti pionieristici per team di sviluppo software. ThoughtWorks ha 42 uffici in 15 paesi: Australia, Brasile, Canada, Cile, Cina, Ecuador, Germania, India, Italia, Singapore, Sud Africa, Spagna, Turchia, il Regno Unito e gli Stati Uniti d'America.

thoughtworks.com