ThoughtWorks®

TECHNOLOGY RADAR *VOL.16*

Le nostre idee sulla tecnologia
e sulle tendenze che
daranno forma al futuro

thoughtworks.com/radar

#TWTechRadar

HANNO CONTRIBUITO

Il Technology Radar viene redatto dal
ThoughtWorks Technology Advisory Board, che include:

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

CHE NOVITÀ CI SONO?

Temi alla ribalta in questa edizione:

INTERFACCE UTENTE COLLOQUIALI ED ELABORAZIONE DEL LINGUAGGIO NATURALE

guarda il video *(thght.works/ConUI)*

La conversazione, una nuova maniera di interagire con le applicazioni, ha sconvolto l'ecosistema con strumenti come Siri, Cortana e Allo, e poi si è spostata all'interno delle case con dispositivi come Amazon Echo e Google Home.

Costruire interfacce utente colloquiali e in linguaggio naturale presenta nuove sfide, ma ha anche ovvi benefici. Il team che ha progettato Echo ha volutamente omesso lo schermo, costringendosi così a ripensare molte interazioni uomo-macchina.

Il trend colloquiale non è limitato solo alla voce; da quando le app di messaging sono arrivate a dominare sia sui telefoni che sul lavoro, vediamo che alle conversazioni con gli esseri umani si affiancano i chatbot intelligenti. Quando queste piattaforme miglioreranno, impareranno a capire il contesto e l'intento delle conversazioni, rendendo le interazioni più realistiche e più convincenti.

L'esplosione di interesse nel mercato e nei media in generale porta una corrispondente crescita dell'interesse degli sviluppatori per questa nuova maniera di interagire con le macchine.

INTELLIGENCE AS A SERVICE

guarda il video *(thght.works/IntSer)*

Una nuova famiglia di piattaforme, che chiamiamo "intelligence as a service" è apparsa recentemente sulla scena. Queste piattaforme comprendono una grande varietà di servizi sorprendentemente potenti, dall'elaborazione della voce alla comprensione del linguaggio naturale, al riconoscimento di immagini, e al deep learning.

Capacità che alcuni anni fa sarebbero state molto costose da sviluppare, ora appaiono come piattaforme open source o SaaS. Sembra che le "guerre del cloud" si siano spostate dalla competizione sullo storage e la computazione, alle capacità cognitive, come si evince dalla volontà di aprire il sorgente di strumenti che solo poco tempo fa erano considerati fattori differenzianti, come Kubernetes e Mesos.

Tutti i big del settore competono in questa arena, insieme a interessanti operatori di nicchia che vale la pena di considerare. Anche se abbiamo ancora delle riserve sulle implicazioni etiche e di privacy di questi servizi, vediamo grandi promesse nell'utilizzo di questi strumenti in maniere innovative. I nostri clienti stanno già investigando quali nuovi orizzonti si possano dischiudere, grazie alla combinazione di servizi di cognizione che sono ormai commodity, con la conoscenza e la comprensione della loro propria area di business.

L'ESPERIENZA DELLO SVILUPPATORE COME NUOVO FATTORE DIFFERENZIANTE

guarda il video *(thght.works/DevExp)*

La progettazione dell'esperienza utente è stata per anni un fattore chiave di differenziazione per le aziende di prodotti tecnologici. Oggi sta avvenendo qualche cosa di simile, con al centro questa volta gli sviluppatori. La rapida crescita della gamma di strumenti a disposizione, assieme alla difficoltà di trovare talento ingegneristico, portano ad una estrema attenzione all'esperienza degli sviluppatori stessi.

Sempre più le organizzazioni valutano le offerte cloud basandosi su quanto queste riducano l'attrito con la funzione ingegneristica, trattano le API come un prodotto, creano nuovi team dedicati a migliorare la produttività ingegneristica. In ThoughtWorks siamo

sempre stati ossessionati dal promuovere pratiche ingegneristiche efficienti ed abbiamo spinto per l'utilizzo di strumenti e piattaforme che possano rendere la vita dello sviluppatore più facile. Pertanto ci fa molto piacere vedere che l'industria nel suo complesso stia iniziando ad intraprendere questa strada.

Tra le tecniche chiave ci piace citare: trattare l'infrastruttura interna come un prodotto che deve essere in grado di competere per efficacia con le offerte di mercato, focalizzarsi sul self-service, capire l'ergonomia per lo sviluppatore delle API che si mettono a disposizione, confinare le applicazioni legacy "in una scatola", ed impegnarsi in una continua ricerca empatica per migliorare l'esperienza degli sviluppatori che utilizzano gli strumenti loro offerti.

L'ASCESA DELLE PIATTAFORME

guarda il video *(thght.works/RiseOTP)*

I temi del Radar emergono da osservazioni e conversazioni svolte durante il processo di selezione. Recentemente abbiamo notato un numero sempre maggiore di nuove entrate nel quadrante delle Piattaforme. Pensiamo che ciò sia indicativo di una tendenza più ampia nell'ecosistema dello sviluppo software.

Famose aziende della Silicon Valley hanno mostrato come la costruzione di piattaforme facili da utilizzare possa portare notevoli benefici. Parte del loro successo deriva dalla capacità di trovare un bilanciamento ottimale fra incapsulamento e funzionalità offerte. Sempre più il pensare a livello di "piattaforma" trova spazio all'interno dell'ecosistema, a partire da funzionalità avanzate messe in luce nel Radar quali il linguaggio naturale, fino alle piattaforme infrastrutturali come Amazon.

Le aziende stanno iniziando a pensare in termini di piattaforma quando espongono le proprie API, ormai ispirate a veri e propri prodotti. I team di sviluppo pensano in termini di piattaforma da integrare ed esperienza sviluppatore da migliorare. Sembra che l'industria stia finalmente trovando una ragionevole combinazione di pacchettizzazione, facilità d'uso ed utilità.

Una definizione che ci piace è che le piattaforme dovrebbero esporre API pronte all'uso in modalità self-service, facili da configurare e fornite di ambienti dedicati ai team di sviluppo; concetti questi in linea con un'altro tema emergente, quello dell'esperienza sviluppatore come nuovo fattore differenziante. Ci aspettiamo di vedere ulteriori raffinamenti sia nella definizione che nelle capacità offerte dalle piattaforme nel prossimo futuro.

PYTHON OVUNQUE

guarda il video *(thght.works/PerPyt)*

Python è un linguaggio che sempre più spesso appare in posti interessanti. La sua facilità d'uso e le solide basi che fornisce al calcolo matematico e scientifico hanno storicamente fatto di questo linguaggio uno dei più adottati nelle comunità accademiche e scientifiche. La recente diffusione di applicazioni che utilizzano Intelligenza Artificiale, unitamente alla maturità di Python 3, hanno contribuito a far guadagnare al linguaggio nuovi utilizzatori.

Questa edizione del Radar riporta alcune librerie Python che hanno aiutato a far crescere l'ecosistema, fra cui Scikitlearn nel dominio del machine learning, TensorFlow, Keras ed Airflow per la costruzione di grafi di flussi dati intelligenti e spaCy che implementa logiche di processo del linguaggio naturale, da utilizzare nella costruzione di API per la gestione di interazioni colloquiali. Vediamo sempre più spesso Python avvicinare, all'interno delle organizzazioni, gli scienziati e gli ingegneri, riducendo i reciproci pregiudizi nei confronti degli strumenti favoriti dai due gruppi.

Approcci architetturali quali microservizi e container hanno reso più facile l'esecuzione di codice Python in ambienti di produzione. Gli ingegneri ora possono facilmente distribuire ed integrare codice Python specializzato, creato dagli scienziati, utilizzando API agnostiche dal punto di vista sia del linguaggio che della tecnologia sottostante. Questa fluidità rappresenta un gran passo in avanti verso la creazione di un ecosistema coerente composto da ingegneri e ricercatori, in contrasto con le pratiche ancora molto presenti nell'industria, che vedono spesso la traduzione da linguaggi specializzati come R in altri linguaggi per gli ambienti di produzione.

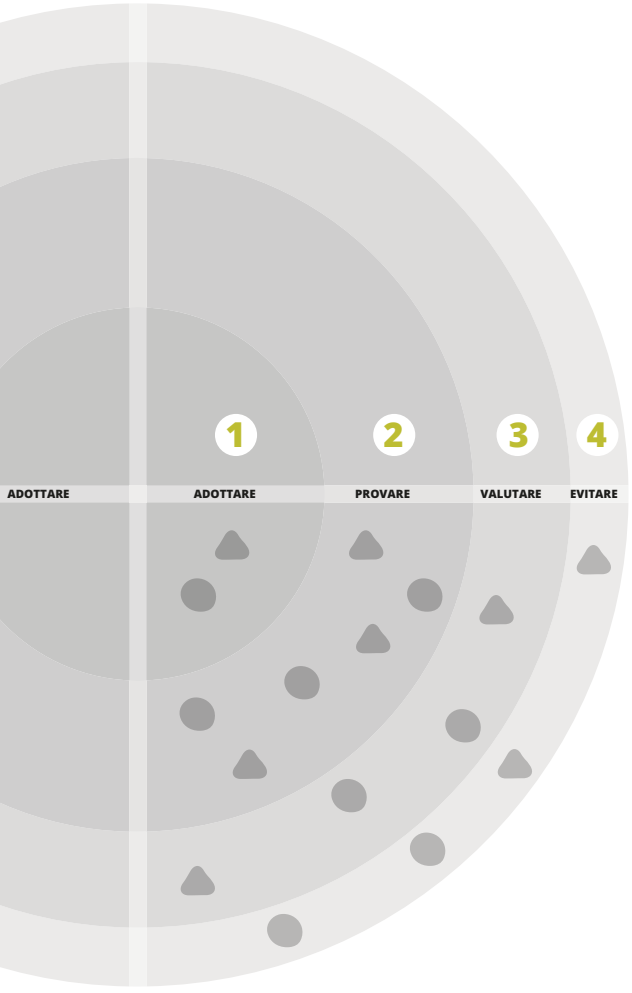
A PROPOSITO DEL RADAR

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 maggiorni 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 portarne in casa la conoscenza. Vanno testare su progetti che possono permettersi di accettarne il rischio.

3 VALUTARE

Vale la pena di esplorare, con l'obiettivo di capire che impatto può 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. Pipelines as code

PROVARE
2. API come prodotto
3. Disaccoppiare i segreti dal codice *NUOVO*
4. Conservare i dati sensibili nella UE
5. Legacy in una scatola *NUOVO*
6. Lightweight Architecture Decision Records
7. Progressive Web Applications *NUOVO*
8. Prototipare con InVision e Sketch *NUOVO*
9. Architetture serverless

VALUTARE
10. Client-directed query
11. Container security scanning
12. API colloquialmente consapevoli *NUOVO*
13. Differential privacy
14. Micro frontends
15. Platform engineering product teams *NUOVO*
16. Analisi sociale del codice *NUOVO*
17. Realtà virtuale al di là del codice

EVITARE
18. Una sola istanza di CI per tutti i team
19. Anemic REST
20. Big Data envy
21. Teatrino CI *NUOVO*
22. Ambienti di test di integrazione a livello aziendale *NUOVO*
23. Spec-based codegen *NUOVO*

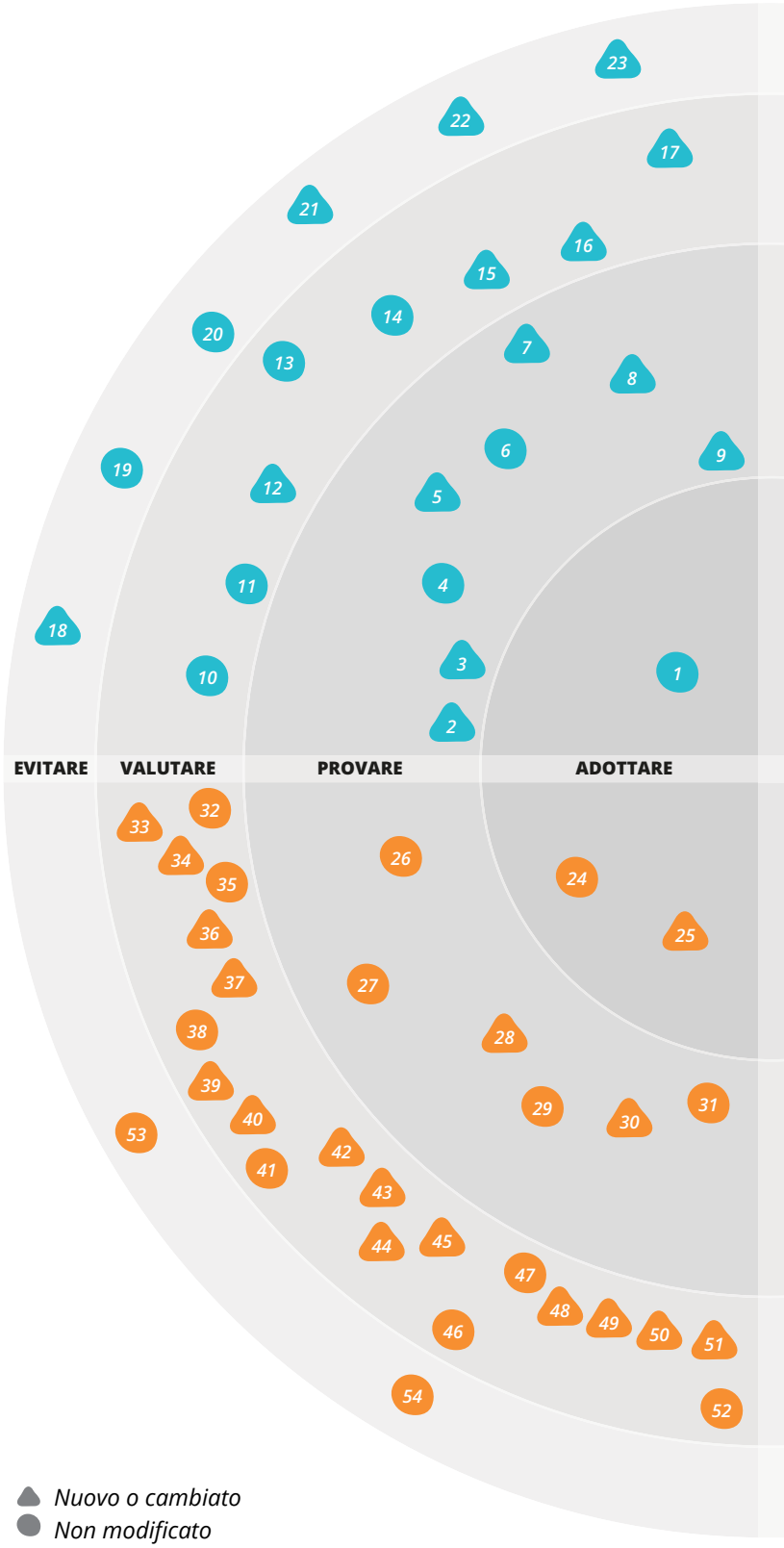
PIATTAFORME

ADOTTARE
24. HSTS
25. Linux Security Modules

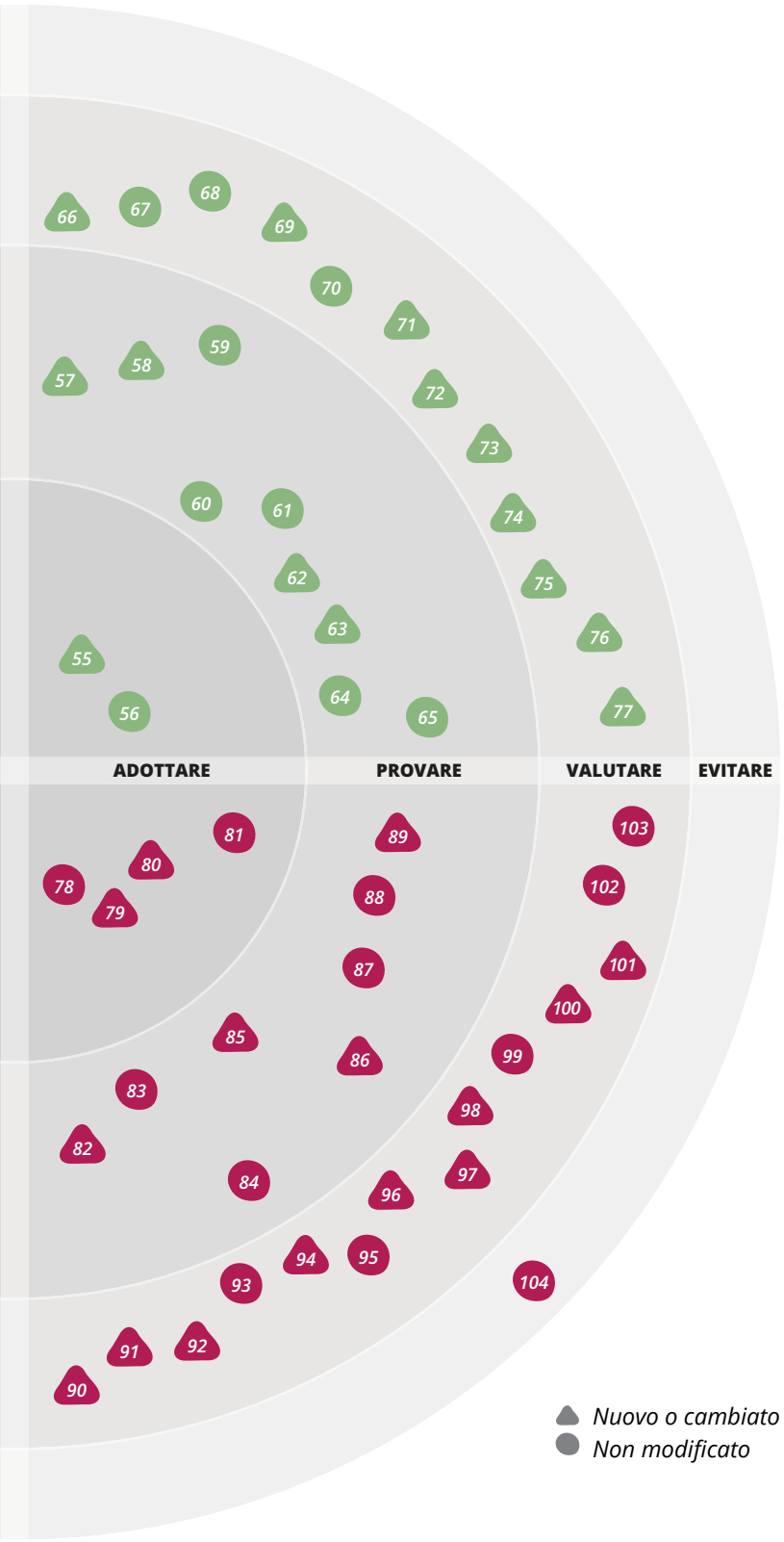
PROVARE
26. Apache Mesos
27. Auth0
28. AWS Device Farm *NUOVO*
29. AWS Lambda
30. OpenTracing *NUOVO*
31. Unity beyond gaming

VALUTARE
32. .NET Core
33. Amazon API Gateway
34. api.ai *NUOVO*
35. Cassandra carefully
36. Comprensione delle immagini nel cloud *NUOVO*
37. DataStax Enterprise Graph *NUOVO*
38. Electron
39. Ethereum
40. Hyperledger *NUOVO*
41. IndiaStack
42. Kafka Streams *NUOVO*
43. Keycloak *NUOVO*
44. Mesosphere DCOS
45. Mosquitto *NUOVO*
46. Nuance Mix
47. OpenVR
48. PlatformIO *NUOVO*
49. Tango *NUOVO*
50. Voice platforms *NUOVO*
51. WebVR *NUOVO*
52. wit.ai

EVITARE
53. CMS as a platform
54. Overambitious API gateways



IL RADAR



STRUMENTI

ADOTTARE
55. fastlane
56. Grafana

PROVARE
57. Airflow *NUOVO*
58. Cake and Fake *NUOVO*
59. Galen
60. HashiCorp Vault
61. Pa11y
62. Scikit-learn
63. Serverless Framework *NUOVO*
64. Talisman
65. Terraform

VALUTARE
66. Amazon Rekognition *NUOVO*
67. Android-x86
68. Bottled Water
69. Claudia *NUOVO*
70. Clojure.spec
71. InSpec *NUOVO*
72. Molecule *NUOVO*
73. Spacemacs *NUOVO*
74. spaCy *NUOVO*
75. Spinnaker *NUOVO*
76. Testinfra *NUOVO*
77. Yarn *NUOVO*

EVITARE

LINGUAGGI & FRAMEWORK

ADOTTARE
78. Ember.js
79. Python 3
80. ReactiveX
81. Redux

PROVARE
82. Avro *NUOVO*
83. Elixir
84. Enzyme
85. Hangfire *NUOVO*
86. Nightwatch *NUOVO*
87. Phoenix
88. Quick and Nimble
89. Vue.js

VALUTARE
90. Angular 2 *NUOVO*
91. Caffe *NUOVO*
92. DeepLearning.scala *NUOVO*
93. ECMAScript 2017
94. Instana *NUOVO*
95. JuMP
96. Keras *NUOVO*
97. Knet.jl *NUOVO*
98. Kotlin *NUOVO*
99. Physical Web
100. PostCSS *NUOVO*
101. Spring Cloud *NUOVO*
102. Three.js
103. WebRTC

EVITARE
104. AngularJS

TECNICHE

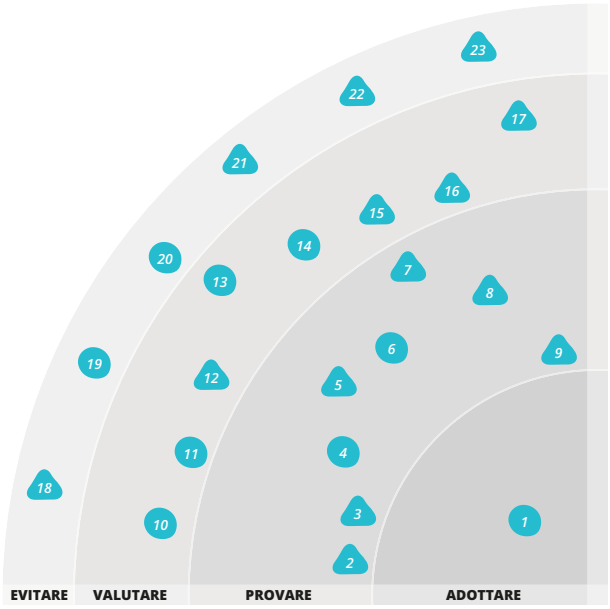
TECNICHE

ADOTTARE
1. Pipelines as code

PROVARE
2. API come prodotto
3. Disaccoppiare i segreti dal codice *NUOVO*
4. Conservare i dati sensibili nella UE
5. Legacy in una scatola *NUOVO*
6. Lightweight Architecture Decision Records
7. Progressive Web Applications *NUOVO*
8. Prototipare con InVision e Sketch *NUOVO*
9. Architetture serverless

VALUTARE
10. Client-directed query
11. Container security scanning
12. API colloquialmente consapevoli *NUOVO*
13. Differential privacy
14. Micro frontends
15. Platform engineering product teams *NUOVO*
16. Analisi sociale del codice *NUOVO*
17. Realtà virtuale al di là del codice

EVITARE
18. Una sola istanza di CI per tutti i team
19. Anemic REST
20. Big Data envy
21. Teatrino CI *NUOVO*
22. Ambienti di test di integrazione a livello aziendale *NUOVO*
23. Spec-based codegen *NUOVO*



Le aziende hanno largamente adottato le API (Application Programming Interface) come maniera di esporre funzionalità di business a sviluppatori sia interni che esterni. Le API promettono di poter sperimentare rapidamente nuove idee di business, ricombinando insieme funzionalità di base. Ma che cosa differenzia un API da un semplice servizio di enterprise integration? Una differenza sta nel trattare le **API COME PRODOTTO,** anche nei casi in cui il consumatore di questa API sia un sistema interno o un collega sviluppatore. I team che realizzano API devono comprendere i bisogni dei loro clienti, per fare in modo che il prodotto sia utile e conveniente. I test di usabilità e la ricerca UX possono portare a un miglior design e a una migliore comprensione di come la API viene usata; tutto ciò aiuta ad arrivare a un modo di pensare orientato al prodotto per le API. Le API, come in generale i prodotti, devono essere attivamente manutenute e supportate, e devono essere facili da usare. Dovrebbero avere un "owner", che prenda le parti del cliente e spinga per un miglioramento continuo. Nella nostra esperienza, l'orientamento al prodotto è l'ingrediente mancante che fa la differenza fra la solita integrazione enterprise e un business agile, costruito su una piattaforma di API.

Nei precedenti numeri del Radar abbiamo citato strumenti come git-crypt e Blackbox che ci consentono di salvare segreti in maniera sicura all'interno del codice sorgente. **DISACCOPPIARE I SEGRETI DAL CODICE** è il nostro modo di ricordare agli esperti di tecnologia che ci sono altre maniere di memorizzare i segreti. Per Esempio, HashiCorp vault, i server di CI e gli strumenti di gestione della configurazione forniscono meccanismi per memorizzare segreti che non sono collegati al codice sorgente di una applicazione. Sono percorribili entrambi gli approcci e vi raccomandiamo di usarne almeno uno nei vostri progetti.

I team che realizzano API devono comprendere i bisogni dei loro clienti, per fare in modo che il prodotto sia utile e conveniente. I test di usabilità e la ricerca UX possono portare a un miglior design e a una migliore comprensione di come la API viene usata; tutto ciò aiuta ad arrivare a un modo di pensare orientato al prodotto per le API.

— API come prodotto

Lavorare con codice legacy, specialmente con grandi monoliti, è una delle esperienze più insoddisfacenti ed irritanti per gli sviluppatori. Sebbene mettiamo in guardia dall'estendere e mantenere attivamente monoliti legacy, questi ultimi continuano a rappresentare delle dipendenze nel nostro ambiente, e gli sviluppatori spesso sottostimano il costo ed il tempo necessario per sviluppare rallentati da queste dipendenze. Per aiutare a ridurre l'attrito gli sviluppatori hanno adottato immagini di macchine virtuali, o immagini containerizzate con Docker per creare immagini immutabili di sistemi legacy e delle loro relative configurazioni. L'idea è di confinare il **LEGACY IN UNA SCATOLA** affinché gli sviluppatori possano eseguirlo localmente, rimuovendo il vincolo della sua ricompilazione, riconfigurazione o condivisione di ambienti. In uno scenario ideale, i team che detengono un sistema legacy generano, attraverso le loro pipeline di build, la corrispondente immagine legacy in una scatola, così gli sviluppatori possono eseguire ed orchestrare tali immagini nelle loro sandbox in modo più affidabile. Sebbene questo approccio abbia ridotto il tempo complessivo impiegato da ogni singolo sviluppatore, ha avuto un successo limitato quando i team, che detengono le dipendenze a valle, si sono dimostrati riluttanti nel creare immagini "containerizzate" ad uso di altri.

L'aumento delle **PROGRESSIVE WEB APPLICATIONS** (PWA) è l'ultimo tentativo di ristabilire il mobile web come risposta alla "fatica da app" percepita dagli utenti. Le PWA, proposte in origine da Google nel 2015, sono applicazioni web che sfruttano i vantaggi delle ultime tecnologie per combinare il meglio delle applicazioni web e mobile. Usando un insieme di standard tecnologici aperti, come i service workers, gli app manifest, e le API di cache e push, possiamo creare applicazioni che sono indipendenti dalla piattaforma ed erogano una esperienza utente simile a una app. Ciò comporta un'equiparazione delle applicazioni web e mobile ed aiuta gli sviluppatori di applicazioni mobili a raggiungere utenti fuori dal recinto degli app store. Si pensino le PWA come dei siti web che sembrano app native.

L'uso combinato di InVision e Sketch ha cambiato il modo in cui alcuni team affrontano lo sviluppo di applicazioni web. Sebbene questi siano tool, ciò che in realtà rende questo elemento rilevante è la tecnica di **PROTOTIPARE CON INVISION E SKETCH**. Creare prototipi ricchi ed interattivi (cliccabili) come primo passo nell'implementazione dei comportamenti applicativi, sia di front end che di back end, permette di velocizzare lo sviluppo e ridurre i ricicli nelle fasi realizzative. L'utilizzo combinato di questi strumenti permette un buon bilanciamento tra l'elaborazione anticipata di dettagli visuali e la possibilità di catturare feedback utente sull'esperienza interattiva.

Un approccio basato su **ARCHITETTURE SERVERLESS** sostituisce le macchine virtuali in esecuzione per lunghi periodi, con potenza di calcolo effimera, che appare su richiesta e scompare immediatamente dopo il suo utilizzo. L'approccio serverless piace ai nostri team; per le nostre esigenze funziona bene e lo consideriamo una valida scelta architetturale. Si noti che l'adozione dell'approccio severless non deve necessariamente essere tutto-o-niente: alcuni dei nostri team hanno rilasciato una nuova parte dei loro sistemi usando serverless, mantenendo un approccio architetturale tradizionale per le altre parti. Sebbene AWS Lambda sia quasi sinonimo di serverless, tutti gli altri maggiori fornitori di cloud propongono offerte simili, e raccomandiamo di valutare anche operatori di nicchia come webtask.

Le tecnologie come Amazon Alexa, Google Voice e Siri hanno reso molto più semplice scrivere software per interazione vocale. Tuttavia, uno stile di interazione più discorsivo (vocale o testuale) può essere ancora difficile da realizzare con molte delle API esistenti

— API colloquialmente consapevoli

Le tecnologie come Amazon Alexa, Google Voice e Siri hanno reso molto più semplice scrivere software per interazione vocale. Tuttavia, uno stile di interazione più discorsivo (vocale o testuale) può essere ancora difficile da realizzare con molte delle API esistenti, specialmente quando si desideri uno stile di interazione stateful, dove la successiva interazione ha bisogno di essere consapevole dell'intero contesto conversazionale. In questo stile di interazione ci piacerebbe, per esempio, chiedere i treni da Manchester a Glasgow e poi essere in grado di domandare "A che ora è la prima partenza?" senza dover fornire nuovamente il contesto della conversazione. Normalmente questo contesto sarebbe presente nella risposta iniziale che abbiamo restituito ad un browser, ma nel caso delle interfacce vocali abbiamo bisogno di gestire tale contesto altrove. Le **API COLLOQUIALMENTE CONSAPEVOLI** possono essere un esempio di backend for front-end in cui il front-end è una piattaforma voce o chat. Questo tipo di API può trattare le specificità di questo stile di interazione gestendo lo stato della conversazione, chiamando al tempo stesso servizi retrostanti per conto del front end vocale.

L'adozione del cloud e di pratiche DevOps ha aumentato la produttività dei team, che ora possono operare più rapidamente, avendo minori dipendenze dai team centralizzati di operations ed infrastrutture. Allo stesso tempo, questi cambiamenti possono rappresentare un problema per quei team che mancano delle competenze per gestire in maniera end-to-end una applicazione, il suo stack tecnologico ed i suoi processi operativi. Alcune organizzazioni hanno affrontato questo tema creando dei **PLATFORM ENGINEERING PRODUCT TEAMS**. Questi team gestiscono una piattaforma interna, che permette ai team di delivery di rilasciare e gestire i sistemi di propria competenza in modalità self-service, con una riduzione sia dei tempi di attesa che della complessità dello stack. In questo caso l'enfasi è posta sugli strumenti di supporto e self-service basati su API; i team di delivery rimangono responsabili per supportare ciò che rilasciano sulla piattaforma. Le organizzazioni interessate a considerare questa opzione devono porre attenzione nel non creare involontariamente un team di DevOps separato, né semplicemente rinominare come "piattaforma" la loro attuale struttura di operations.

L'**ANALISI SOCIALE DEL CODICE** migliora la nostra comprensione della qualità del codice, arricchendo l'analisi strutturale del codice con l'analisi del comportamento dello sviluppatore. Utilizza dati estratti dai sistemi di controllo di versione, come la frequenza e l'istante di una modifica, e la persona che la ha eseguita. Si può scegliere di scriversi da soli gli script per analizzare tali dati, oppure utilizzare strumenti come CodeScene. CodeScene può aiutare ad acquisire una miglior comprensione dei sistemi software, identificando punti critici e sottosistemi complessi e difficili da manutenere: dall'accoppiamento temporale fra sottosistemi distribuiti, alla proiezione della legge di Conway nella organizzazione. Alla luce delle tendenze tecnologiche come i sistemi distribuiti, i microservizi e i team distribuiti, crediamo che la dimensione sociale del codice rivesta un ruolo vitale per la comprensione olistica della salute dei sistemi.

L'idea della realtà virtuale è in giro da più di 50 anni, e con i successivi progressi nella tecnologia, molte idee sono state reclamizzate ed esplorate. Crediamo di aver ormai raggiunto un punto di non ritorno. L'anno scorso sono arrivati sul mercato occhiali VR a prezzi ragionevolmente accessibili per il consumatore, e le schede grafiche di oggi sono sufficientemente potenti per creare delle vere esperienze di immersione. Questi occhiali sono per lo più destinate agli appassionati di videogiochi, ma siamo convinti che apriranno le porte a molte possibilità per la **REALTÀ VIRTUALE AL DI LÀ DEI GIOCHI**. I team che non hanno esperienza nella realizzazione di videogiochi faranno bene a non sottovalutare l'impegno e le conoscenze necessarie per creare buoni modelli 3D e texture convincenti.

L'idea della realtà virtuale è in giro da più di 50 anni, e con i successivi progressi nella tecnologia, molte idee sono state reclamizzate ed esplorate. Crediamo di aver ormai raggiunto un punto di non ritorno.

— Realtà virtuale al di là dei giochi

Siamo costretti a consigliare prudenza, un'altra volta, nei confronti della creazione di **UNA SOLA ISTANZA DI CI** per tutti i team. Sebbene in teoria sia una bella idea consolidare e centralizzare l'infrastruttura di CI, in pratica non vediamo una maturità sufficiente negli strumenti e nei prodotti in quest'ambito per conseguire i risultati desiderati. I team di sviluppo che devono usare un'offerta di CI centralizzata hanno regolarmente notevoli ritardi, che dipendono dal team centrale, per eseguire attività minori di configurazione, o per risolvere problemi nell'infrastruttura condivisa e nel tooling. In questa fase, continuiamo a raccomandare alle organizzazioni di limitare gli investimenti in centralizzazione alla sola definizione di pattern, linee guida e supporto, perché i team di sviluppo possano gestirsi da soli la propria infrastruttura di CI.

Siamo da molto tempo sostenitori della continuous integration (CI), e siamo stati pionieri nella realizzazione di server di CI per eseguire build dei progetti automaticamente dopo un commit. Usati bene, questi programmi monitorano il ramo principale del progetto, sul quale gli sviluppatori contribuiscono codice quotidianamente. Il server di CI compila il progetto ed esegue test esaustivi per assicurare che l'intero sistema software sia sempre integrato in uno stato rilasciabile, soddisfando perciò il principio di continuous delivery. Sfortunatamente, molti sviluppatori si limitano ad installare un server di CI assumendo erroneamente di "fare CI" quando, in realtà, non ne ottengono alcun beneficio. Cause di fallimento tipiche includono: eseguire CI su un ramo principale condiviso ma con rari commit, per cui l'integrazione non è veramente continua; eseguire il build con una copertura scadente dei test, lasciando il build rosso per lunghi periodi; oppure eseguire CI su feature branch, che si traduce in continuous isolation. Un tale "**TEATRINO CI**" potrà anche far contento qualcuno, ma fallirebbe qualsiasi test di certificazione CI credibile.

Quando i rilasci mensili o trimestrali che coinvolgevano l'intera azienda erano considerati una best practice, era necessario mantenere un ambiente completo per eseguire i cicli di test prima di rilasciare in produzione. Questi **AMBIENTI DI TEST DI INTEGRAZIONE A LIVELLO AZIENDALE**, spesso chiamati System Integration & Testing (SIT) o Staging, oggi sono un collo di bottiglia per la continuous delivery. Questi ambienti sono fragili e costosi da mantenere, e spesso hanno componenti che richiedono una configurazione manuale da parte di un apposito team di gestione dell'ambiente. Testare nell'ambiente di staging fornisce un feedback lento e inaffidabile, e il lavoro di test eseguito in questo ambiente è un duplicato dei test che possono essere eseguiti sui componenti isolati. Raccomandiamo che le organizzazioni creino incrementalmente una percorso indipendente di delivery in produzione per i componenti più importanti. Alcune tecniche utili sono: contract testing, decoupling deployment from release, focus on mean time to recovery e testing in production.

Sfortunatamente, molti sviluppatori si limitano ad installare un server di CI assumendo erroneamente di "fare CI" quando, in realtà, non ne ottengono alcun beneficio. Un tale "Teatrino CI" potrà anche far contento qualcuno, ma fallirebbe qualsiasi test di certificazione CI credibile.

— Teatrino CI

Ai tempi in cui SOAP dominava l'industria del software enterprise, la pratica di generare codice lato client dalle specifiche WSDL era accettata e perfino incoraggiata. Sfortunatamente, il codice risultante era spesso complesso, non testabile, difficile da modificare e poco portabile fra differenti piattaforme. Con l'avvento di REST, troviamo che sia meglio far evolvere i client con il pattern tolerant reader, per leggere ed elaborare solo i campi necessari. Recentemente abbiamo osservato un ritorno allarmante alle vecchie abitudini, con sviluppatori che generano codice dalle specifiche di API scritte con Swagger o RAML, una pratica che chiamiamo **SPEC-BASED CODEGEN**. Sebbene questi strumenti siano molto utili per guidare la progettazione delle API e per estrarre documentazione, mettiamo in guardia dalla facile scorciatoia di generare codice client da queste specifiche. I rischi sono che tale codice sia difficile da testare e manutenere.

PIATTAFORME

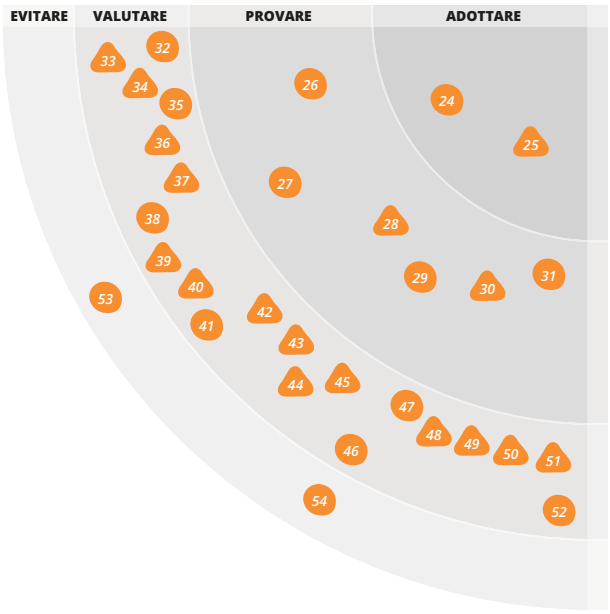
PIATTAFORME

ADOTTARE
24. HSTS
25. Linux Security Modules

PROVARE
26. Apache Mesos
27. Auth0
28. AWS Device Farm *NUOVO*
29. AWS Lambda
30. OpenTracing *NUOVO*
31. Unity beyond gaming

VALUTARE
32. .NET Core
33. Amazon API Gateway
34. api.ai *NUOVO*
35. Cassandra carefully
36. Comprensione delle immagini nel cloud *NUOVO*
37. DataStax Enterprise Graph *NUOVO*
38. Electron
39. Ethereum
40. Hyperledger *NUOVO*
41. IndiaStack
42. Kafka Streams *NUOVO*
43. Keycloak *NUOVO*
44. Mesosphere DCOS
45. Mosquitto *NUOVO*
46. Nuance Mix
47. OpenVR
48. PlatformIO *NUOVO*
49. Tango *NUOVO*
50. Voice platforms *NUOVO*
51. WebVR *NUOVO*
52. wit.ai

EVITARE
53. CMS as a platform
54. Overambitious API gateways



Il Principio del Privilegio Minimo ci incoraggia a imporre che i componenti software possano accedere solo alle risorse di cui hanno bisogno. Tuttavia, un processo Linux normalmente può fare tutto quello che può fare l'utente che lo esegue, dall'aprire porte TCP arbitrarie al lanciare nuove shell. Il framework **LINUX SECURITY MODULES** (LSM) framework, che consente di inserire estensioni di sicurezza nel kernel, è stato usato per implementare il Mandatory Access Control su Linux. SELinux e AppArmor sono le più note e diffuse implementazioni compatibili con LSM e sono distribuite con il kernel. Raccomandiamo che i team imparino ad usare uno di questi framework di sicurezza (ed è per questo che li abbiamo messi nella cerchia "Adopt"). Questi framework aiutano i team a porsi domande su chi ha accesso a quali risorse su host condivisi, compresi i servizi in essi contenuti. Questo approccio prudente alla gestione degli accessi aiuterà i team a rendere la sicurezza una parte integrante del loro ciclo di sviluppo.

AMAZON API GATEWAY permette di pubblicare API su Internet. Comprende i servizi tipici di un API gateway, come la gestione del traffico, il monitoraggio, autenticazione e autorizzazione. I nostri team hanno avuto esperienze positive nell'uso di questo servizio come front end per AWS Lambda, per implementare architetture serverless. D'altra parte, abbiamo avuto problemi quando l'abbiamo usato come API gateway di uso generale, come front end di servizi HTTP/HTTPS su EC2. In questi casi, siamo stati frustrati dalla scarsa interoperabilità con le VPC e dalla difficoltà di eseguire l'autenticazione mediante client certificates. Per via di queste differenti esperienze, consigliamo ai team di sperimentare l'uso di AWS API Gateway con Lambda, ma di prestare attenzione alla sua efficacia per l'uso più generale.

L'enorme numero di dispositivi mobili rende quasi impossibile per le aziende testare le proprie app mobili su tutti i dispositivi. Ma oggi c'è **AWS DEVICE FARM**, un servizio di test per app che consente di eseguire ed interagire con le app Android, iOS e web su di un'ampia gamma di dispositivi fisici che sono simultaneamente ospitati nel cloud. Durante ogni singola esecuzione sono generati log dettagliati, grafici di prestazione ed istantanee delle videate, allo scopo di fornire un riscontro sia generale che specifico per quel particolare dispositivo. Il servizio offre molta flessibilità, consentendo di alterare lo stato e la configurazione di ogni dispositivo al fine di riprodurre scenari di test molto specifici. I nostri team stanno usando AWS Device Farm per eseguire test end-to-end sui dispositivi maggiormente diffusi.

Da quando si stanno sostituendo le applicazioni monolitiche con ecosistemi più complessi basati su microservizi, tracciare le richieste attraverso molteplici servizi sta diventando la norma.

— OpenTracing

Da quando si stanno sostituendo le applicazioni monolitiche con ecosistemi più complessi basati su (micro)servizi, tracciare le richieste attraverso molteplici servizi sta diventando la norma. Fortunatamente **OPENTRACING** sta rapidamente affermandosi come lo standard de facto per il tracciamento distribuito. Sviluppato da Uber, Apple, Yelp ed altri fra i maggiori attori, supporta più tracciatori come Zipkin, Instana, e Jaeger. Attualmente OpenTracing è implementato in sei linguaggi: Go, JavaScript, Java, Python, Objective-C and C++, in maniera indipendente da un fornitore specifico.

Parallelamente alla recente ondata di chatbot e piattaforme vocali, abbiamo osservato una proliferazione di strumenti e piattaforme come **API.AI**, che forniscono un servizio che estrae l'intento dello scrivente da un testo e gestisce il flusso di una conversazione. Recentemente acquisita da Google, questa offerta di "comprensione-del-linguaggio-naturale as a service" è in concorrenza con altri attori in questo mercato come wit.ai e Amazon Lex.

La comprensione delle immagini era un tempo un'arte oscura che richiedeva un team di data scientist sul posto. Negli ultimi anni, invece, siamo arrivati molto vicini alla soluzione di problemi come la classificazione e categorizzazione di immagini e volti, il confronto di volti, l'identificazione dei punti facciali ed il riconoscimento facciale. La **COMPRENSIONE DELLE IMMAGINI NEL CLOUD** fornisce accesso a funzionalità di machine-learning per mezzo di servizi come Amazon Rekognition, Microsoft Computer Vision API e Google Cloud Vision API, che possono integrare applicazioni AR e qualsiasi altra cosa che comporti il tagging e la classificazione di foto.

Abbiamo ottenuto i primi successi con **DATASTAX ENTERPRISE GRAPH** (DSE Graph) per trattare grandi database di grafi. Basato su Cassandra, DSE Graph DSE Graph si colloca nell'ambito di grandi data set dove Neo4j, il nostro preferito da molto tempo, inizia a mostrare alcuni limiti. Tale ordine di grandezza ha i suoi compromessi; per esempio, si perdono le transazioni ACID e l'indipendenza dallo schema di Neo4j; tuttavia, l'accesso alle tabelle di Cassandra sottostanti, l'integrazione di Spark per lavori analitici e il potente linguaggio di query TinkerPop/Gremlin rendono questa soluzione un'opzione che vale la pena considerare.

Il clamore intorno ai blockchain e alle crittovalute sembra aver raggiunto l'apice, come si vede dal diradarsi degli annunci un tempo torrenziali in quest'area, e ci aspettiamo che alcuni dei progetti più speculativi si estinguano con il tempo. Uno dei blockchain, **ETHEREUM**, anche se non è il più popolare fra i gli esperti di blockchain, viene usato in un numero crescente di nuove iniziative. Ethereum è un blockchain

pubblico con un linguaggio di programmazione incorporato, che permette agli sviluppatori di scrivere "smart contracts", che sono movimenti algoritmici di "etere" (la crittovaluta di Ethereum), in risposta ad attività in corso sul blockchain. R3CEV, il consorzio che costruisce tecnologia blockchain per le banche, ha rilasciato i suoi primi proof of concept su Ethereum. Ethereum è stato usato per costruire una organizzazione autonoma distribuita (DAO, Distributed Autonomous Organization), una delle prime "corporation algoritmiche", anche se un recente furto di 150 milioni di dollari in "etere" prova che i blockchain e le crittovalute sono ancora il Far West del mondo tecnologico.

HYPERLEDGER è una piattaforma costruita attorno alle tecnologie blockchain che consiste di un'implementazione blockchain, chiamata "Fabric", e di altri tool associati. Trascurando il clamore che circonda blockchain, i nostri team hanno trovato facile iniziare ad usare questi tool. Al nostro entusiasmo per Hyperledger si aggiunge anche il fatto che esso sia una piattaforma open source supportata da Linux Foundation.

KAFKA STREAMS è una libreria leggera per applicazioni streaming. È stata disegnata con l'obiettivo di semplificare la gestione dei processi basati su stream, per avere un modello di programmazione facilmente utilizzabile nella costruzione di servizi asincroni. È una opzione attraente nei casi in cui si voglia applicare il modello di programmazione basato su stream, senza aggiungere la complessità della gestione di un cluster (cosa di solito richiesta dai framework più completi dedicati ai processi streaming).

In un'architettura a microservizi o in una qualsiasi altra architettura distribuita, una delle più comuni necessità è la messa in sicurezza dei servizi o delle API attraverso autenticazione e autorizzazione. Questo è l'ambito in cui si colloca **KEYCLOAK**. Keycloak è una soluzione open source di gestione dell'identità e dell'accesso, che rende semplice mettere in sicurezza applicazioni e microservizi con poco o nessun codice. Pronto all'uso c'è il supporto al single sign-on, al social login, ed ai protocolli standard come OpenID Connect, OAuth2 e SAML..

MESOSPHERE DCOS è una piattaforma basata su Mesos che astrae l'infrastruttura sottostante, per applicazioni containerizzate ed anche per quelle applicazioni che non si desideri eseguire all'interno di Docker. Per installazioni più modeste ciò potrebbe sembrare esagerato, ma stiamo iniziando ad osservare successi sia con la versione commerciale che con la versione open source. Ci piace particolarmente il fatto che agevoli la portabilità fra diversi fornitori di cloud e di hardware dedicato, e che per lavori containerizzati non vincoli ad un framework di orchestrazione di container. Sebbene gli aggiornamenti possano essere un po' troppo complessi rispetto a come ci piacerebbe che fossero, l'intero stack si sta stabilizzando bene.

Nella nostra esperienza con soluzioni Internet of Things (IoT), in cui molti dispositivi comunicano l'uno con l'altro e/o con un hub di dati centrale, il protocollo di connettività MQTT ha dato buona prova di sé. Abbiamo cominciato ad apprezzare anche l'MQTT broker **MOSQUITTO**. Non soddisfa tutte le esigenze, specialmente quelle di scalabilità, ma la sua natura compatta e la semplice installazione lo rendono ideale per sviluppo e test.

PLATFORMIO fornisce un ricco ecosistema per lo sviluppo IoT attraverso processi di compilazione cross-platform, gestione delle librerie ed una buona integrazione con gli IDE esistenti. Il completamento del codice intelligente ed il code linter con un terminale ed un monitor di porta seriale in dotazione migliorano enormemente l'esperienza di sviluppo. PlatformIO organizza e mantiene anche migliaia di librerie e fornisce una chiara gestione delle dipendenze con versionamento semantico per semplificare lo sviluppo IoT. Abbiamo iniziato ad usare PlatformIO in alcuni progetti IoT e ci è veramente piaciuto per la sua semplicità ed ampio supporto di piattaforme and schede.

L'anno scorso la realtà aumentata (AR) e la realtà mista (MR) sono entrate nella norma insieme alla realtà virtuale (VR), quest'ultima caratterizzata da una barriera di ingresso relativamente più alta, sia a causa di requisiti hardware che dello sforzo di creazione di mondi virtuali. Inoltre Pokémon Go ha dimostrato che gli ordinari smartphone sono sufficienti per creare esperienze AR/MR convincenti. **TANGO** è una nuova tecnologia di sensori hardware per telefoni cellulari che migliora ulteriormente le possibilità per la AR/MR sui cellulari. Tango consente alle app di acquisire misure 3D dettagliate dell'ambiente circostante l'utente, cosicché gli oggetti virtuali possano essere piazzati e resi sulla videocamera in modo più convincente. I primi cellulari con tecnologia Tango sono già disponibili.

Le **PIATTAFORME VOCALI** come Amazon Alexa e Google Home stanno cavalcando la cresta dell'onda promozionale; alcuni addirittura annunciano l'ubiquità delle interfacce vocali conversazionali. Stiamo già integrando UI conversazionali all'interno di prodotti e stiamo vedendo l'impatto di questa nuova interazione nel modo in cui progettiamo le interfacce. Nello specifico, Alexa è stata costruita fin dall'inizio senza schermo e considera la UI conversazionale di primaria importanza. Ma è ancora troppo presto per credere agli annunci iperbolici, e ci aspettiamo che altri grandi attori entrino sul mercato.

Stiamo già integrando UI conversazionali all'interno di prodotti e stiamo vedendo l'impatto di questa nuova interazione nel modo in cui progettiamo le interfacce.

— Piattaforme vocali

WEBVR è una API JavaScript sperimentale che consente l'accesso a dispositivi di realtà virtuale attraverso il browser. Ha conquistato il supporto della comunità ed è disponibile in build giornaliere e anche in versioni stabili. Se stai progettando di costruire esperienze VR nel browser, questo è un eccellente punto di partenza. Questa tecnologia, insieme a strumenti complementari come Three.js, A-Frame, ReactVR, Argon.js e Awe.js, porta l'esperienza della realtà aumentata nel browser. Il fiorire di strumenti in questo spazio, contemporaneamente agli standard della commissione Internet, possono aiutare a promuovere una maggiore adozione di AR e VR.

STRUMENTI

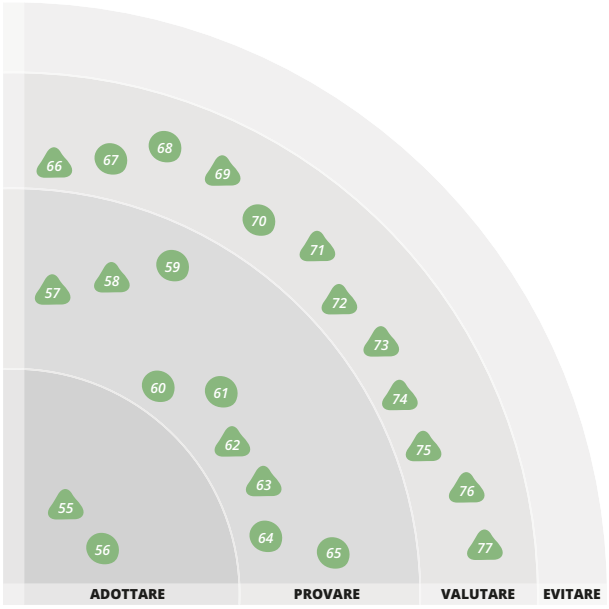
STRUMENTI

ADOTTARE
55. fastlane
56. Grafana

PROVARE
57. Airflow *NUOVO*
58. Cake and Fake *NUOVO*
59. Galen
60. HashiCorp Vault
61. Pa11y
62. Scikit-learn
63. Serverless Framework *NUOVO*
64. Talisman
65. Terraform

VALUTARE
66. Amazon Rekognition *NUOVO*
67. Android-x86
68. Bottled Water
69. Claudia *NUOVO*
70. Clojure.spec
71. InSpec *NUOVO*
72. Molecule *NUOVO*
73. Spacemacs *NUOVO*
74. spaCy *NUOVO*
75. Spinnaker *NUOVO*
76. Testinfra *NUOVO*
77. Yarn *NUOVO*

EVITARE



Gli sviluppatori di applicazioni web hanno gioco facile quando si parla di semplificare ed automatizzare diversi workflow di sviluppo; possono scegliere fra una varietà di soluzioni per automatizzare il loro processo di rilascio. Quando si sviluppa su mobile, invece, abbiamo due sistemi operativi con due diverse maniere di compilare, testare, distribuire, generare screenshot, firmare e distribuire applicazioni. Per mitigare il problema, i nostri team hanno adottato **FASTLANE** per automatizzare il processo di rilascio per applicazioni iOS e Android. Con semplici configurazioni e pipeline multiple, possono ottenere la continuous delivery per lo sviluppo mobile.

AIRFLOW è uno strumento per creare programmaticamente, schedulare e monitorare pipeline di dati. Airflow incoraggia pipeline di dati manutenibili, versionabili e testabili, trattando i grafi diretti aciclici (Direct Acyclic Graphs, DAG) come codice. Abbiamo sfruttato questa configurazione nei nostri progetti e siamo stati in grado di creare pipeline dinamiche che hanno prodotto workflow di dati chiari ed espliciti. Airflow rende semplice definire i vostri operatori ed esecutori, ed estendere la libreria cosicché soddisfi il livello di astrazione che meglio si adatta al vostro ambiente.

Per mitigare il problema, i nostri team hanno adottato fastlane per automatizzare il processo di rilascio per applicazioni iOS e Android.

— fastlane

MSBuild è stato il sistema di build principale nell'ecosistema .NET fin dal suo debutto nel 2005; tuttavia, ha sempre gli stessi punti deboli che abbiamo segnalato in Maven. La comunità .NET ha cominciato a sviluppare alternative a MSBuild che sono più semplici da manutenere e più flessibili, e possono evolvere in maniera più fluida al crescere delle dimensioni del progetto. Due di queste alternative sono **CAKE** E **FAKE**. Cake usa un DSL scritto in C#, mentre Fake usa un DSL in F#. Entrambi hanno visto una crescita significativa nel corso dell'ultimo anno e si sono dimostrati una valida alternativa a MSBuild, per orchestrare i normali task di compilazione in progetti .NET.

La comunità .NET ha cominciato a sviluppare alternative a MSBuild che sono più semplici da manutenere e più flessibili, e possono evolvere in maniera più fluida al crescere delle dimensioni del progetto.

— Cake e Fake

SCIKIT-LEARN non è uno strumento nuovo (sta per compiere dieci anni); quello che è nuovo è il tasso di adozione di strumenti e tecniche di machine learning al di fuori dell'accademia e delle grandi aziende tecnologiche. Scikit-learn fornisce un robusto insieme di modelli e un ricco insieme di funzionalità, e gioca un ruolo importante nel rendere concetti e capacità di machine learning più accessibili a un pubblico più ampio e spesso non specialista.

Il popolare **SERVERLESS FRAMEWORK** fornisce strumenti per creare ed installare applicazioni serverless, usando principalmente AWS Lambda e altri servizi di AWS. Il Serverless Framework consente di creare applicazioni in JavaScript, Python, Java e C#, e ha una comunità attiva che contribuisce plugin che estendono il framework. Il framework supporta anche il progetto OpenWhisk (nell'incubatore Apache) come alternativa ad AWS Lambda.

AMAZON REKOGNITION è uno degli strumenti cloud di analisi delle immagini che abbiamo menzionato altrove in questo Radar. Quello che ci piace è che Amazon ha adottato un approccio relativamente nuovo (usando le GUID) per mantenere l'anonimità dei volti nei confronti di AWS, tenendo così conto dei problemi di privacy correlati al riconoscimento facciale.

La combinazione di AWS Lambda con Amazon API Gateway ha avuto un grosso impatto nella maniera di distribuire servizi e API. Tuttavia, anche lavorando in serverless, la quantità di configurazioni necessarie per collegare insieme i pezzi non è banale. **CLAUDIA** è uno strumento che automatizza l'installazione di funzioni AWS Lambda scritte in JavaScript e della configurazione API Gateway associata. I default che fornisce sono convenienti, e i nostri team hanno trovato che permette di iniziare a lavorare velocemente con microservizi basati su Lambda.

Come può un'azienda concedere autonomia ai team di sviluppo, e allo stesso tempo garantire che le soluzioni da loro consegnate siano sicure e in linea con gli standard? Come si può garantire che i server, una volta installati, restino sicuri e in linea con gli standard per tutta la durata della loro vita? Questi sono i problemi che **INSPEC** cerca di risolvere. InSpec è uno strumento di test dell'infrastruttura ispirato a Serverspec, ma con differenze che lo rendono più utile per i professionisti della sicurezza, che hanno bisogno di garantire l'aderenza a uno standard di migliaia di server. I test individuali possono essere combinati in profili di sicurezza completi, ed eseguiti remotamente dalla riga di comando. InSpec è utile durante lo sviluppo, ma anche per testare l'infrastruttura di produzione in maniera continua, nella direzione della QA in produzione.

MOLECULE aiuta nello sviluppo e nel test di ruoli di Ansible. Ci risparmia dal dover configurare manualmente il nostro ambiente di test, perché costruisce un'impalcatura per eseguire i test dei ruoli di Ansible. Molecule si avvantaggia di Vagrant, Docker, e OpenStack per gestire macchine virtuali e container, e supporta Serverspec, Testinfra, o Goss per eseguire i test. La sequenza dei passi eseguiti di default comprende: gestione delle macchine virtuali, lint di codice Ansible, test di idempotenza e test di convergenza. Anche se è un progetto abbastanza giovane, vediamo in Molecule un grande potenziale.

Come qualsiasi fan di Emacs dirà, Emacs è più di un text editor; è una piattaforma per applicazioni basate su caratteri. Negli ultimi anni, c'è stata un'esplosione di nuovi sviluppi su questa piattaforma, ma pensiamo che **SPACEMACS** meriti un'attenzione speciale. Spacemacs fornisce un'introduzione alla piattaforma Emacs, con una nuova interfaccia utente basata sulla tastiera, i layer di customizzazione semplificati e una distribuzione curata di pacchetti Emacs. Uno degli obiettivi di questo progetto è di essere il meglio dei due mondi, combinando l'interfaccia utente di Vim con la riprogrammabilità interna di Emacs. Noi pensiamo che gli strumenti di produttività dei programmatori siano un componente vitale dello sviluppo software efficace, e se è passato un po' di tempo dall'ultima volta che hai considerato Emacs, ti consigliamo di dare un'occhiata a come Spacemacs ripensa questa classica piattaforma di sviluppo.

SPACY è una libreria di Natural Language Processing (NLP) scritta in Python. È una libreria ad alta performance, pensata per l'uso in produzione, e applica modelli NLP adatti all'elaborazione di testi che contengano emoticon e segni di punteggiatura inconsistenti. A differenza di altri framework NLP, spaCy è una libreria che può essere inserita in un progetto e non una piattaforma; è destinata ad applicazioni in produzione, invece che all'addestramento di modelli per la ricerca. Si accompagna bene a TensorFlow e al resto dell'ecosistema dell'Intelligenza Artificiale in Python. Abbiamo usato spaCy in un contesto enterprise per costruire un motore di ricerca che accetta domande in linguaggio naturale ed aiuta gli utenti a prendere decisioni di business.

Netflix ha reso open source **SPINNAKER**, la sua piattaforma per la continuous delivery (CD) di microservizi. In confronto ad altre piattaforme CI/CD, Spinnaker implementa la gestione di cluster e il rilascio sul cloud di immagini customizzate (es. AMI) come caratteristica principale. Supporta di default l'installazione e la gestione di cluster per diversi fornitori di cloud, come Google Cloud Platform, AWS e Pivotal Cloud Foundry. Si può integrare Spinnaker con Jenkins per eseguire un job di compilazione su Jenkins. A noi piace l'approccio opinionato di Spinnaker al rilascio di microservizi sul cloud, tranne il fatto che le pipelines di Spinnaker sono create con un'interfaccia utente e non possono essere configurate come codice.

Dato l'ampio utilizzo di strumenti per la gestione della infrastruttura, non deve sorprendere che il concetto di "infrastructure as code" abbia guadagnato molto spazio. Questa tendenza porta con sé la necessità di testare automaticamente tale codice. **TESTINFRA** permette di testare lo stato dei server, sia che essi siano configurati manualmente che tramite strumenti quali Ansible, Puppet and Docker. Testinfra è un plugin del test engine Pytest e mira ad essere equivalente a Serverspec.

Dato l'ampio utilizzo di strumenti per la gestione della infrastruttura, non deve sorprendere che il concetto di "infrastructure as code" abbia guadagnato molto spazio.

— Testinfra

YARN è un nuovo package manager che sostituisce l'attuale workflow del client npm, rimanendo compatibile con il registry di npm. Utilizzando il client npm è possibile ottenere strutture ad albero (sotto la directory node_modules) differenti a seconda dell'ordine in cui le dipendenze sono installate. Questa natura non deterministica del client npm può causare problemi tipo "sulla mia macchina funziona". Yarn risolve questo problema spezzando l'installazione in più passi (risoluzione delle dipendenze, scaricamento dei pacchetti e linking) ed utilizzando algoritmi deterministici e lock-files, in modo da garantire installazioni ripetibili. Noi abbiamo inoltre rilevato, nei nostri ambienti di continuous integration (CI), tempi di build molto più rapidi, resi possibili dal caching dei pacchetti scaricati.

LINGUAGGI & FRAMEWORK

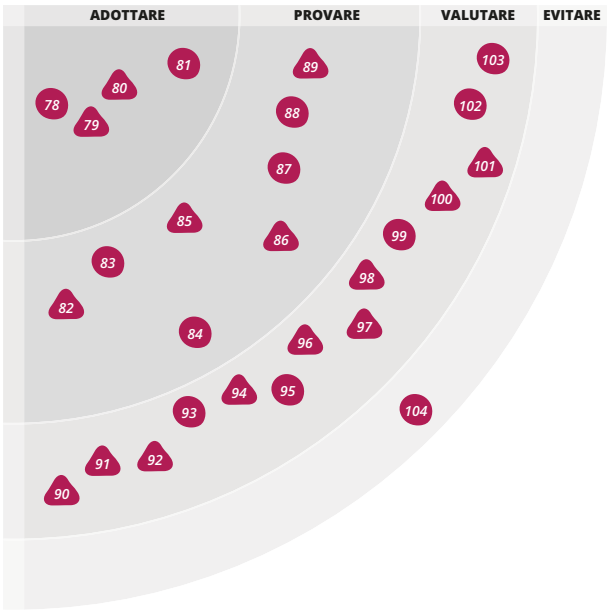
LINGUAGGI & FRAMEWORK

ADOTTARE
78. Ember.js
79. Python 3
80. ReactiveX
81. Redux

PROVARE
82. Avro *NUOVO*
83. Elixir
84. Enzyme
85. Hangfire *NUOVO*
86. Nightwatch *NUOVO*
87. Phoenix
88. Quick and Nimble
89. Vue.js

VALUTARE
90. Angular 2 *NUOVO*
91. Caffe *NUOVO*
92. DeepLearning.scala *NUOVO*
93. ECMAScript 2017
94. Instana *NUOVO*
95. JuMP
96. Keras *NUOVO*
97. Knet.jl *NUOVO*
98. Kotlin *NUOVO*
99. Physical Web
100. PostCSS *NUOVO*
101. Spring Cloud *NUOVO*
102. Three.js
103. WebRTC

EVITARE
104. AngularJS



PYTHON 3 ha introdotto molti elementi utili, che tuttavia non sono retrocompatibili con Python 2.x. Ha anche rimosso diverse componenti di Python 2.x che erano state mantenute per retrocompatibilità, rendendo Python 3 più facile da imparare ed utilizzare, oltre che complessivamente più coerente. La nostra esperienza nell'utilizzo di Python 3 negli ambiti di machine learning e sviluppo di applicazioni web mostra che sia il linguaggio che la gran parte delle librerie a supporto sono maturati e sono pronti per essere adottati. Abbiamo risolto piccoli problemi in alcune librerie tramite fork e patch, oppure abbiamo evitato di usare librerie Python 2.x non più supportate. Il nostro suggerimento per chi sviluppa in Python è quindi di utilizzare Python 3.

I sistemi distribuiti spesso utilizzano multi-threading, meccanismi di comunicazione basati su eventi e I/O non bloccante al fine di migliorare l'efficienza complessiva. Queste tecniche di programmazione comportano difficoltà quali: gestione dei thread a basso livello, thread safety, sincronizzazione, strutture dati concorrenti ed appunto operazioni di I/O non bloccanti. La libreria open source **REACTIVEX** fornisce una elegante astrazione di questi concetti, oltre che efficaci strumenti applicativi per gestirli, ed estende il pattern observable agli stream di eventi asincroni. Attorno a ReactiveX inoltre si è sviluppata una comunità attiva di sviluppatori ed il numero di linguaggi supportati è in continua espansione (l'ultimo entrato è RxSwift). La libreria inoltre è integrata con le maggiori piattaforme mobile e desktop.

La libreria open source ReactiveX fornisce una elegante astrazione di questi concetti: thread safety, sincronizzazione, strutture dati concorrenti ed I/O non bloccante; fornisce efficaci strumenti applicativi ed estende il pattern observable agli stream di eventi asincroni.

— ReactiveX

AVRO è un framework per la serializzazione dei dati. Per il fatto che salva lo schema insieme al contenuto dei dati, facilita l'evoluzione dello schema. I produttori di dati possono modificare i nomi dei campi, aggiungere campi o cancellare campi esistenti e Avro garantisce che i clienti continueranno a consumare i messaggi. Avere uno schema consente di scrivere ciascun dato senza overhead, il che significa codifica più compatta e tempi di elaborazione ridotti. Anche se lo scambio di messaggi privi di struttura fra produttore e consumatore è flessibile, abbiamo visto che alcuni team hanno avuto il problema di messaggi non processati rimasti in coda durante i rilasci. Abbiamo usato Avro in parecchi porgetti e raccomandiamo di usarlo, invece di mandare messaggi non strutturati.

Hangfire è facile da usare e al tempo stesso flessibile, e si basa su uno stile funzionale. È particolarmente interessante la sua capacità di salvare lo stato di un task, cosicché possa ripartire quando l'applicazione riparte dopo un crash o un arresto.

— Hangfire

Un problema comune nello sviluppo di applicazioni è come schedulare task che devono eseguire al di fuori del processo principale, periodicamente o al verificarsi di certe condizioni. Il problema diventa più complicato quando accadono eventi inattesi, come l'arresto dell'applicazione. Il framework **HANGFIRE** può fare questo e molto altro, nell'ambiente .NET, come hanno scoperto i nostri team. Hangfire è facile da usare e al tempo stesso flessibile, e si basa su uno stile funzionale. È particolarmente interessante la sua capacità di salvare lo stato di un task, cosicché possa ripartire quando l'applicazione riparte dopo un crash o un arresto.

NIGHTWATCH è un framework che permette di creare test di accettazione automatizzati in JavaScript per applicazioni che eseguono nel browser, e di eseguirli in Node.js. I test di Nightwatch si scrivono con una API fluente che può poi essere eseguita tramite un server Selenium/WebDriver. Nel caso di una single page app o comunque di pagine pesantemente basate su JavaScript, questo permette di scrivere ed eseguire test nello stesso linguaggio ed ambiente del grosso del codice.

Nel mondo in continua evoluzione dei framework Javascript per front end, **VUE.JS** appare meritevole di particolare attenzione. Vue.js è una alternativa più semplice e leggera ad AngularJS. È una libreria molto flessibile, e meno opinionata, che offre strumenti per la costruzione di interfacce web interattive basandosi su concetti di modularità, componenti e reactive data flow. Ha una curva di apprendimento dolce, il che lo rende particolarmente interessante per sviluppatori junior e principianti. È da notare tuttavia che Vue.js non è un framework completo; si focalizza soltanto sul livello "view" e si integra quindi facilmente con altre librerie o progetti esistenti.

Nella precedente edizione del Radar avevamo collocato AngularJS, nella cerchia "Evitare" (dove rimane tuttora). Quando si parla di **ANGULAR 2**, però, riceviamo indicazioni contrastanti. Nel corso dell'ultimo anno alcuni team di ThoughtWorks hanno utilizzato con successo Angular2 e lo considerano un framework solido. Tuttavia Angular2 è una riscrittura e non una evoluzione di AngularJS, e passare da AngularJS ad Angular2 non è molto diverso dal passare da AngularJS ad un qualsiasi altro framework. Dato che, nella nostra esperienza, esistono soluzioni migliori come React.js, Ember.js and Vue.js, non ci sentiamo ancora pronti per raccomandare in maniera forte Angular2. Vogliamo allo stesso tempo sottolineare che che Angular2 non rappresenta una scelta sbagliata, specialmente per team ed organizzazioni che già utilizzano TypeScript.

CAFFE è una libreria open-source creata per il deep-learning dal Berkeley Vision and Learning Center. È specializzata sulle reti convoluzionali per applicazioni di computer vision. Caffe è una soluzione solida e molto

utilizzata per compiti di computer vision; sono inoltre disponibili diversi modelli sviluppati da utilizzatori di Caffe, pronti all'uso e già utilizzati con successo, che possono essere scaricati dal Caffe Model Zoo. Come Keras, anche Caffe è un API basata su Python. In Keras, tuttavia, i modelli ed i componenti sono oggetti creati direttamente con codice Python, mentre i modelli Caffe sono descritti tramite file di configurazione Protobuf. Ambedue gli approcci hanno pro e contro; la conversione fra i due è comunque possibile.

DEEPLEARNING.SCALA è una libreria open source per il deep learning scritta in Scala, creata da alcuni nostri colleghi in ThoughtWorks. Riteniamo che questo progetto sia estremamente interessante, perché utilizza tecniche di programmazione funzionale differenziabile per creare e comporre reti neurali; gli sviluppatori devono solo scrivere codice in Scala utilizzando la tipizzazione statica. DeepLearning.Scala al momento supporta tipi base quali float, double, vettori multidimensionali con accelerazione GPU così come tipi di dati algebrici. Le prossime versioni dovrebbero supportare le funzioni di ordine superiore e il training distribuito su Spark.

INSTANA è un nuovo strumento che si aggiunge ai tanti esistenti per la gestione delle performance applicative. Si differenzia da molti dei suoi concorrenti per il fatto che è costruito fin dall'inizio per architetture cloud native. Le sue caratteristiche comprendono la discovery automatica, il tracciamento distribuito e la salute del servizio, e fornisce inoltre la possibilità di "traslare nel tempo" la vista dell'infrastruttura fino al momento in cui un incidente è avvenuto. Resta da vedere se questo prodotto potrà imporsi nei confronti delle alternative basate su una combinazione di progetti open source, come Consul, Prometheus e le implementazioni di OpenTracing che fanno la stessa cosa; tuttavia, può essere utile considerarlo ove ci sia bisogno di una soluzione preconfezionata.

KERAS è un'interfaccia di alto livello in Python per costruire reti neurali. Keras è stata creata da un ingegnere di Google, è open source e gira su TensorFlow oppure su Theano. Fornisce un'interfaccia semplicissima per creare algoritmi potenti di deep-learning, da addestrare su CPU o GPU. Keras è ben progettata, in termini di modularità, semplicità ed estendibilità. Al contrario di librerie come Caffe, Keras supporta architetture di reti più generali, come le recurrent networks, il che la rende alla fine più utile per analisi dei testi, elaborazione del linguaggio naturale e machine learning generico. Per applicazioni di computer vision, o qualsiasi altra branca specializzata del machine learning, Caffe può essere una scelta più appropriata. Ma per chi stia cercando un framework semplice e al tempo stesso potente, Keras dovrebbe essere la prima scelta.

KNET.JL è il framework di deep-learning della Koç University implementato in Julia da Deniz Yuret e i suoi collaboratori. Si differenzia dai compilatori che generano gradienti come Theano and TensorFlow, che forzano gli utenti ad usare un mini-linguaggio ristretto, perché Knet consente di definire ed addestrare modelli di machine-learning usando la piena potenza ed espressività di Julia. Knet usa grafi computazionali dinamici generati a runtime per differenziare automaticamente la maggior parte dei costrutti di Julia. Apprezziamo molto il supporto delle operazioni GPU dato dal tipo KnetArray, e in caso non si abbia accesso a una macchina GPU, il team di Knet mantiene una Immagine di macchina Amazon preconfigurata (AMI) in modo da poterlo valutare nel cloud.

Il linguaggio di programmazione **KOTLIN** è quest'anno nella lista di cose da valutare di molti nostri sviluppatori, e qualcuno lo ha già usato in produzione con successo. Kotlin è un linguaggio open source per la JVM ideato da JetBrains. Piace ai nostri sviluppatori per dispositivi mobili che usano Swift, perché e sintatticamente vicino a Swift ed ugualmente conciso. Ai nostri sviluppatori Java è piaciuta la sua fluida interoperabilità con il linguaggio e gli strumenti Java, e l'hanno trovato più semplice da imparare rispetto a Scala. Kotlin supporta concetti di programmazione funzionale, ma in maniera

meno completa rispetto a Scala. I nostri sviluppatori che apprezzano la tipizzazione statica, con il compilatore che rileva difetti di puntatori null, hanno osservato che si scrivono meno test boilerplate.

I team che costruiscono sistemi composti da microservizi hanno la necessità di pensare a tecniche di coordinamento come scoperta dei servizi, bilanciamento del carico, interruzione di circuito, health check.

— Spring Cloud

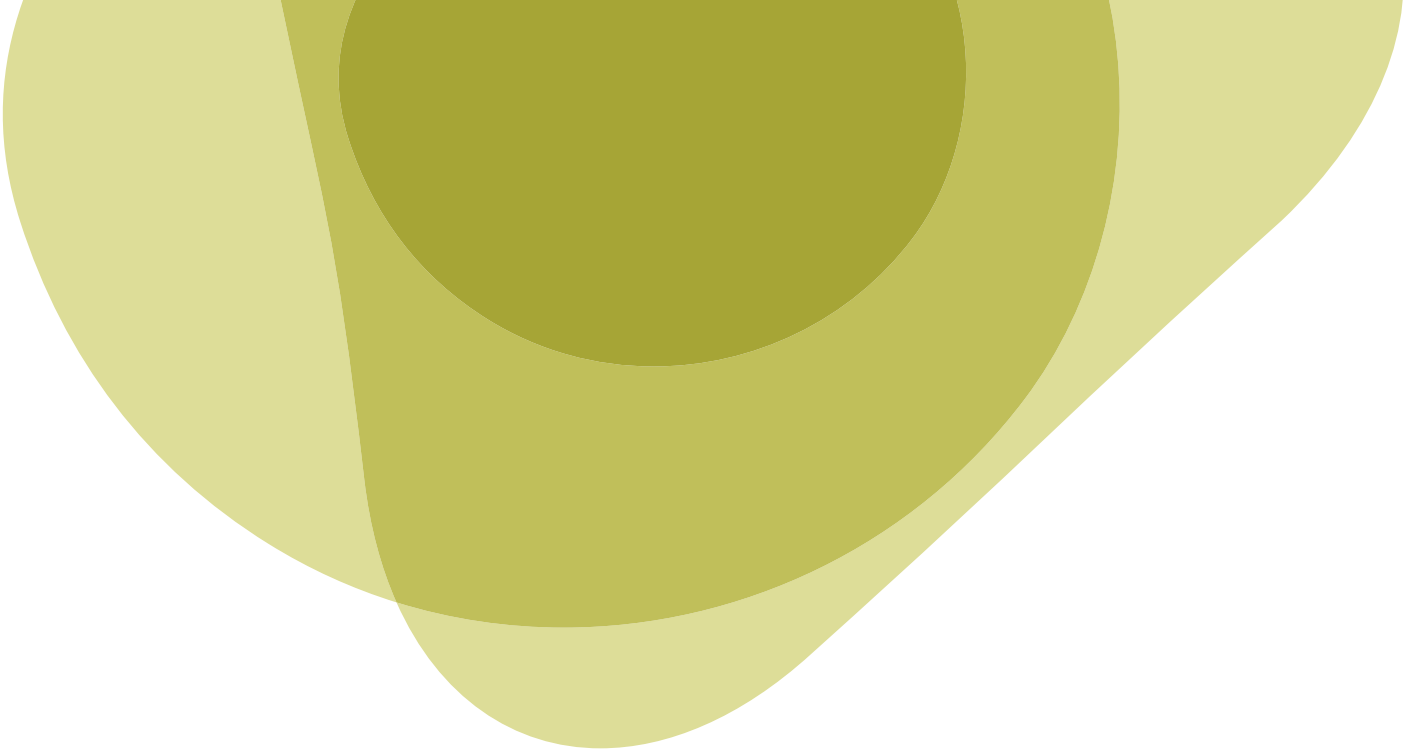
POSTCSS è un framework JavaScript che si fonda su Node.js e che consente, tramite una sintassi astratta basata su alberi, di operare su documenti CSS tramite un ricco ecosistema di plugin. La vera forza di PostCSS, spesso erroneamente considerato alla stregua di un preprocessore come SaaS o Less, risiede nel numero di cose che possono essere fatte con il ricco insieme di plugin che include: linting (il plugin stylelint), cross-compilation (il plugin sugarss), decorazione dei nomi per evitare collisioni di selettori (il plugin modules), generazione di CSS boilerplate (il plugin autoprefixer), minificazione e molto altro. Nonostante il differente livello di maturità dei vari plugin, PostCSS rimane un framework semplice e potente per trattare CSS come un linguaggio maturo per lo sviluppo front-end.

I team che costruiscono sistemi composti da microservizi hanno la necessità di pensare a tecniche di coordinamento come scoperta dei servizi, bilanciamento del carico, interruzione di circuito, health check. Molte di queste tecniche richiedono ai team l'installazione, che non sempre è banale, di opportuni strumenti. Il progetto **SPRING CLOUD** fornisce strumenti per gli sviluppatori affinché possano utilizzare tali tecniche di coordinamento nell'ambiente familiare di Spring. Questi strumenti supportano Consul, ZooKeeper e lo stack completo di Netflix OSS, tutti strumenti che ci piacciono. Semplificando, Spring Cloud rende semplice fare la cosa giusta con questi insiemi di strumenti. Nonostante rimanga la nostra solita preoccupazione con Spring, cioè che nasconda troppo la complessità, dovreste considerare Spring Cloud se lavorate in tale ecosistema e avete necessità di risolvere questi problemi.

Sii il primo a sapere quando uscirà il prossimo Technology Radar e mantieniti aggiornato con contenuti e webinar esclusivi.

ABBONATI

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 4000 persone, compresa una divisione prodotti che realizza strumenti pionieristici per team di sviluppo software. ThoughtWorks ha 40 uffici in 14 paesi: Australia, Brasile, Cile, Cina, Ecuador, Germania, India, Italia, Singapore, Sud Africa, Spagna, Turchia, il Regno Unito e gli Stati Uniti d'America.

thoughtworks.com