ThoughtWorks®

# TECHNOLOGY RADAR *VOL.16*

Informació sobre la tecnologia
i les tendències que estan
modelant el futur

*thoughtworks.com/radar*

***#TWTechRadar***

# COL·LABORADORS

*El Radar Tecnològic ha sigut creat pel Gabinet d'Assessors de ThoughtWorks Technology format per:*


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

# QUÈ HI HA DE NOU?

*Aquests són els temes principals d'aquesta edició:*

## INTERFÍCIE D'USUARI CONVERSACIONAL I PROCESSAMENT DE LLENGUATGE NATURAL

*mireu el vídeo (thght.works/ConUI)*

La nova manera d'interactuar amb les aplicacions utilitzant el llenguatge va ser un èxit rotund gràcies a eines com Siri, Cortana i Allo, i ha arribat a les cases gràcies a aparells com el Amazon Echo o el Google Home.

Crear interfícies d'usuari conversacionals i que utilitzen llenguatge natural, tot i presentar molts reptes, té clars beneficis. L'equip de disseny de l'Echo va ometre intencionalment la pantalla per a forçar-se a replantejar les interaccions humà-màquina.

Aquesta tendència conversacional no es limita a la veu. A mida que les aplicacions de missatgeria han crescut fins a dominar tant els telèfons com els llocs de treball, veiem com les converses amb humans s'estan canviant per bots conversacionals intel·ligents. A mida que aquestes plataformes evolucionin, aprendran a entendre el context i la intenció de la conversa, la qual cosa farà les interacciones més reals i interessants.

El gran interès del mercat i dels medis de comunicació dominants fa que hi hagi un increment proporcional en el desenvolupament d'aquest tipus d'interacció personal.

## INTEL·LIGÈNCIA COM A SERVEI

*mireu el vídeo (thght.works/IntSer)*

Recentment, han aparegut en escena una sèrie de plataformes que anomenem intel·ligència com a servei. Aquestes plataformes engloben una gran varietat d'utilitats sorprenentment poderoses que van des del processament de veu fins a la comprensió del llenguatge natural i passant pel reconeixement d'imatges i l'aprenentatge profund.

Capacitats que haguessin consumit cars recursos només fa uns anys, apareixen ara com a plataformes de codi obert o SaaS. Sembla que "la guerra dels núvols" ha passat de lluitar per l'emmagatzematge i la computació cap a les capacitats cognitives, com ho demostra la disposició de fer que eines que antigament eren clau, com ara Kubernetes i Mesos, ara siguin de codi obert.

En aquest espai, tots els grans tenen la seva oferta a la vegada que també hi ha jugadors especialitats que val la pena d'avaluar. Tot i que encara tenim les nostres reserves pel que fa a les implicacions ètiques i de privacitat d'aquests serveis, veiem un gran potencial al fet d'utilitzar aquestes poderoses eines de manera innovadora. Els nostres clients ja estan investigant quins nous horitzons se'ls poden obrir combinant aquestes eines amb informació sobre la seva pròpia empresa.

## EXPERIÈNCIA DE DESENVOLUPADOR COM A NOU DIFERENCIADOR

*mireu el vídeo (thght.works/DevExp)*

El disseny de l'experiència d'usuari ha sigut, durant anys, un diferenciador clau per a les empreses de productes de tecnologia. Ara, però, el ràpid creixement d'eines i productes enfocats als desenvolupadors i la falta de talent en enginyeria, fa que hi hagi un interès similar per l'experiència dels desenvolupadors.

Cada vegada hi ha més organitzacions que avaluen els serveis al núvol tenint en compte la fricció que estalvien, que tracten les APIs com a productes i que redistribueixen els equips per a maximitzar la productivitat. A ThoughtWorks, sempre ens ha obsessionat la productivitat i hem promogut eines i plataformes que fan la vida dels desenvolupadors més fàcil, per la qual cosa ens agrada veure com la industria està començant a adoptar aquest enfocament.

Les tècniques clau inclouen: tractar la infraestructura interna com un producte que ha de ser prou captivador com per lluitar amb les ofertes externes,

centrar-se en l'autoservei, entendre l'ergonomia de les APIs que es produeixen, contenir els "sistemes heretats envasats" i comprometre's a dur a terme un estudi dels desenvolupadors que utilitzen els vostres serveis.

## LA PUJADA DE LES PLATAFORMES

*mireu el vídeo (thght.works/RiseOTP)*

Els temes del Radar surten de l'observació i les converses que es duen a terme durant el procés d'investigació. Fa poc, mentre quadràvem el Radar, ens vam adonar del nombre de noves entrades en el quadrant de les plataformes i creiem que és un indicador d'una nova tendència en l'ecosistema del desenvolupament de programari.

Grans companyies de Sillicon Valley han demostrat com el fet de crear una plataforma adequada pot tenir grans beneficis. Part del seu èxit es deu a la seva capacitat de trobar un nivell útil d'encapsulament i capacitats. A més, el "pensament en plataformes" té cada vegada més pes en l'ecosistema: des de capacitats avançades que es troben en el Radar com ara el llenguatge natural, fins a plataformes d'infraestructura com ara Amazon.

Les empreses estan començant a pensar en les plataformes quan exposen capacitats a través d'APIs basades en els productes. Els equips de desenvolupament pensen més en la creació de plataformes per a la integració i la millor experiència de desenvolupament. Sembla que la industria ha trobat, per fi, una combinació adequada entre l'embalatge, la conveniència i la utilitat.

Una definició de plataforma que ens agrada és que haurien d'exposar una API d'autoservei i ser fàcils d'abastar i de configurar per un equip de desenvolupadors. Això, a més, està relacionat amb un altre tema emergent: l'experiència dels desenvolupadors com a nou diferenciador. Esperem veure encara més millores tant en la definició com en les capacitats de les plataformes en el futur.

## GENERALITZACIÓ DE PYTHON

*mireu el vídeo (thght.works/PerPyt)*

Python és un llenguatge que no para d'aparèixer en llocs interessants. La seva facilitat d'ús com a llenguatge de programació general, combinada amb la seva gran base matemàtica i de computació científica, ha fet que, històricament, s'hagi adoptat tant per la comunitat acadèmica com per la comunitat investigadora. Més recentment, la tendència de la industria sobre la generalització i les aplicacions de la intel·ligència artificial, combinada amb la maduresa de Python 3, ha fet que altres comunitats s'interessin per Python.

Aquesta adició del Radar compta amb algunes llibreries de Python que han ajudat a estimular l'ecosistema. Algunes d'aquestes llibreries són: Scikit-learn, pel que fa al domini de l'aprenentatge automàtic; Tensorflow, Keras i Airflow, per a gràfics de dades intel·ligents; i spaCy, que implementa el processament de llenguatge natural per a potenciar la creació d'APIs conversacionals. A més, veiem com, cada vegada més, Python ajuda a acostar els científics i els enginyers dintre d'una organització, la qual cosa ajuda a perdre els favoritismes cap a la seva eina preferida.

Alguns enfocaments arquitecturals, com ara els microserveis i els contenidors, han facilitat l'execució de Python en entorns de producció. Els enginyers ara poden implementar i integrar codi Python creat per científics amb APIs independents del llenguatge i la tecnologia. Aquesta fluïditat és un gran pas cap a un ecosistema consistent entre els investigadors i els enginyers, a diferència de la pràctica de traduir llenguatge especialitzat, com ara R, als entorns de producció.

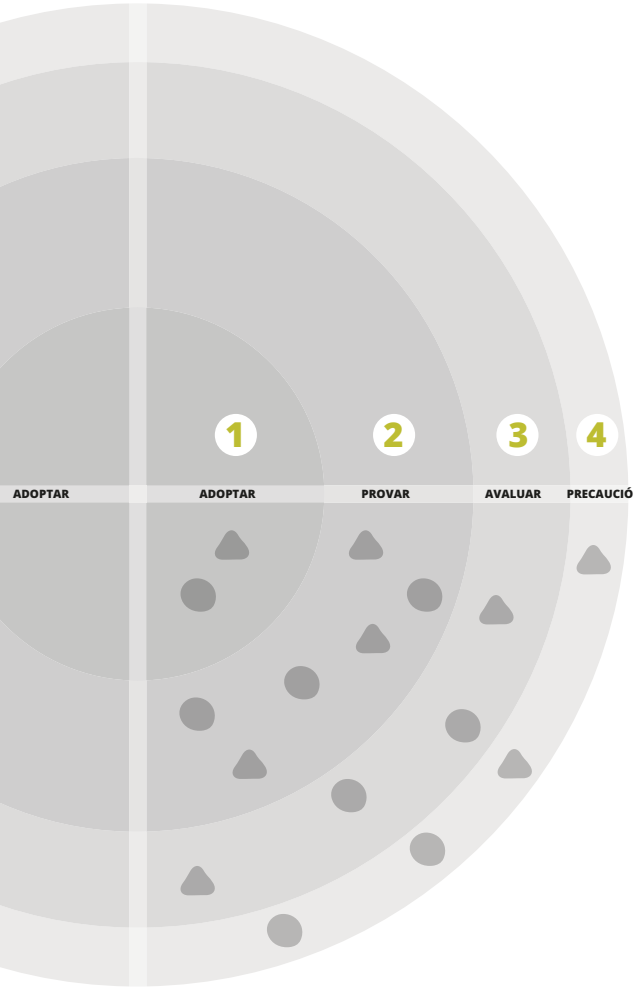
# SOBRE EL RADAR

Als treballadors de ThoughtWorks els apassiona la tecnologia. La creem, l'investiguem, la posem a prova, la compartim gràcies al codi obert, escrivim sobre ella i sempre la volem millorar per a beneficiar a tothom. La nostra missió és arribar a l'excel·lència i revolucionar la informàtica. Amb aquesta missió en ment, creem i compartim el Radar Tecnològic. El Radar el crea el gabinet d'assessors de ThoughtWorks Technology, un grup de líders sèniors de ThoughtWorks. Es reuneixen periòdicament per a parlar sobre l'estratègia global de ThoughtWorks i les tendències tecnològiques que tenen un impacte significatiu sobre la nostra industria. .

Aquest Radar captura l'essència de les xerrades del gabinet d'assessor i la comparteix en un format adequat per a una gran varietat d'actors, des de gerents de sistemes (CIO) fins a desenvolupadors. El contingut del Radar s'ha pensat com un resum concís d'aquestes tecnologies.

Recomanem que s'explorin més detalladament aquestes tecnologies. El Radar té una naturalesa més aviat gràfica i agrupa els elements en tècniques, eines, plataformes i llenguatges i entorns de treball. Si algun element pot aparèixer en més d'un quadrant, escollim el que ens sembla més adient. Agrupem aquests elements en 4 anells per a reflectir el que en pensem.

Per a més informació sobre el radar, visiteu thoughtworks.com/radar/faq




## RADAR D'UN COP D'ULL

**1 ADOPTAR**

Creiem fermament que la industria hauria d'adoptar aquests elements. Nosaltres els utilitzem quan ho creiem oportú en els nostres projectes.

**2 PROVAR**

Val la pena seguir investigant. És important entendre com desenvolupar aquesta habilitat. Les empreses haurien de provar aquesta tecnologia en un projecte en el que es pogués permetre el risc.

**3 AVALUAR**

Val la pena explorar-la per a poder entendre com afectaria l'empresa.

**4 PRECAUCIÓ**

Continuar amb precaució.

**NOU O CANVIAT**

**SENSE CANVIS**

Els elements nous o que han canviat de manera significativa des de l'últim Radar es representen amb un triangle mentre que els elements que no s'han mogut es representen amb un cercle.

! El nostre radar té visió de futur. Per a deixar espai per a nous elements, deixem que elements que no s'han mogut recentment s'esvaeixin, la qual cosa no és un reflex del seu valor, sinó més bé una limitació del nostre Radar.

# EL RADAR

## TÈCNIQUES

**ADOPTAR**
1. Pipelines com a codi

**PROVAR**
2. APIs com a producte
3. Desacoblament de secrets del codi font *NOU*
4. Allotjar dades PII a la UE
5. Sistemes heretats envasats *NOU*
6. Registre de decisions d'arquitectura de baix pes
7. Aplicacions web progressives *NOU*
8. Fer prototips amb InVision i Sketch *NOU*
9. Arquitectura sense servidor

**AVALUAR**
10. Recerca orientada al client
11. Estudi de la seguretat dels contenidors
12. APIs conversacionals *NOU*
13. Privacitat diferencial
14. Micro-interfícies
15. Equips de producció de plataformes *NOU*
16. Anàlisi social del codi *NOU*
17. Realitat Virtual més enllà dels videojocs

**PRECAUCIÓ**
18. Instància única d'IC per a tots els equips
19. REST anèmic
20. Enveja pel Big Data
21. Teatre d'IC *NOU*
22. Entorns de proves d'integració a tota l'empresa *NOU*
23. Generació de codi basat en les especificacions *NOU*

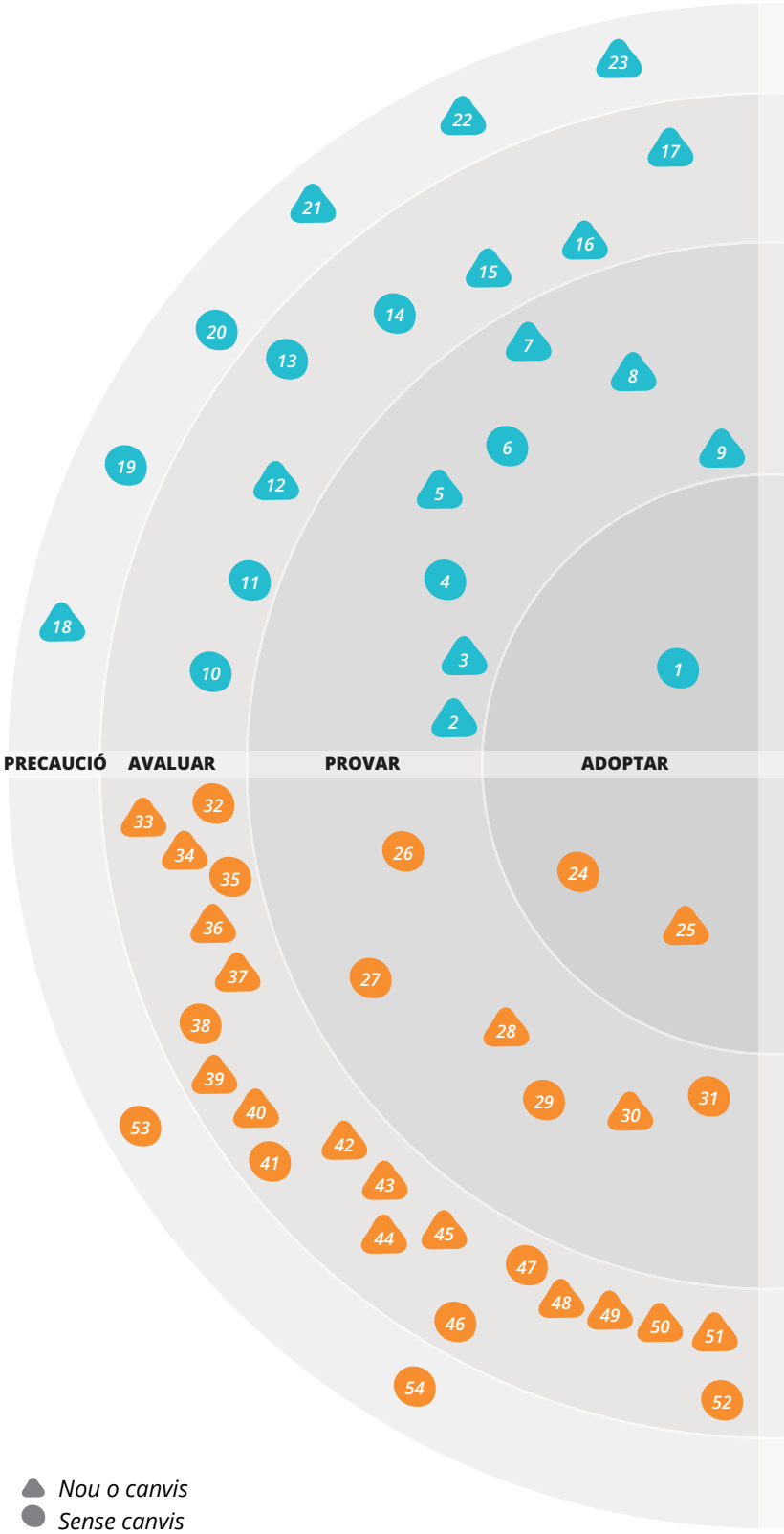
## PLATAFORMES

**ADOPTAR**
24. HSTS
25. Linux Security Modules

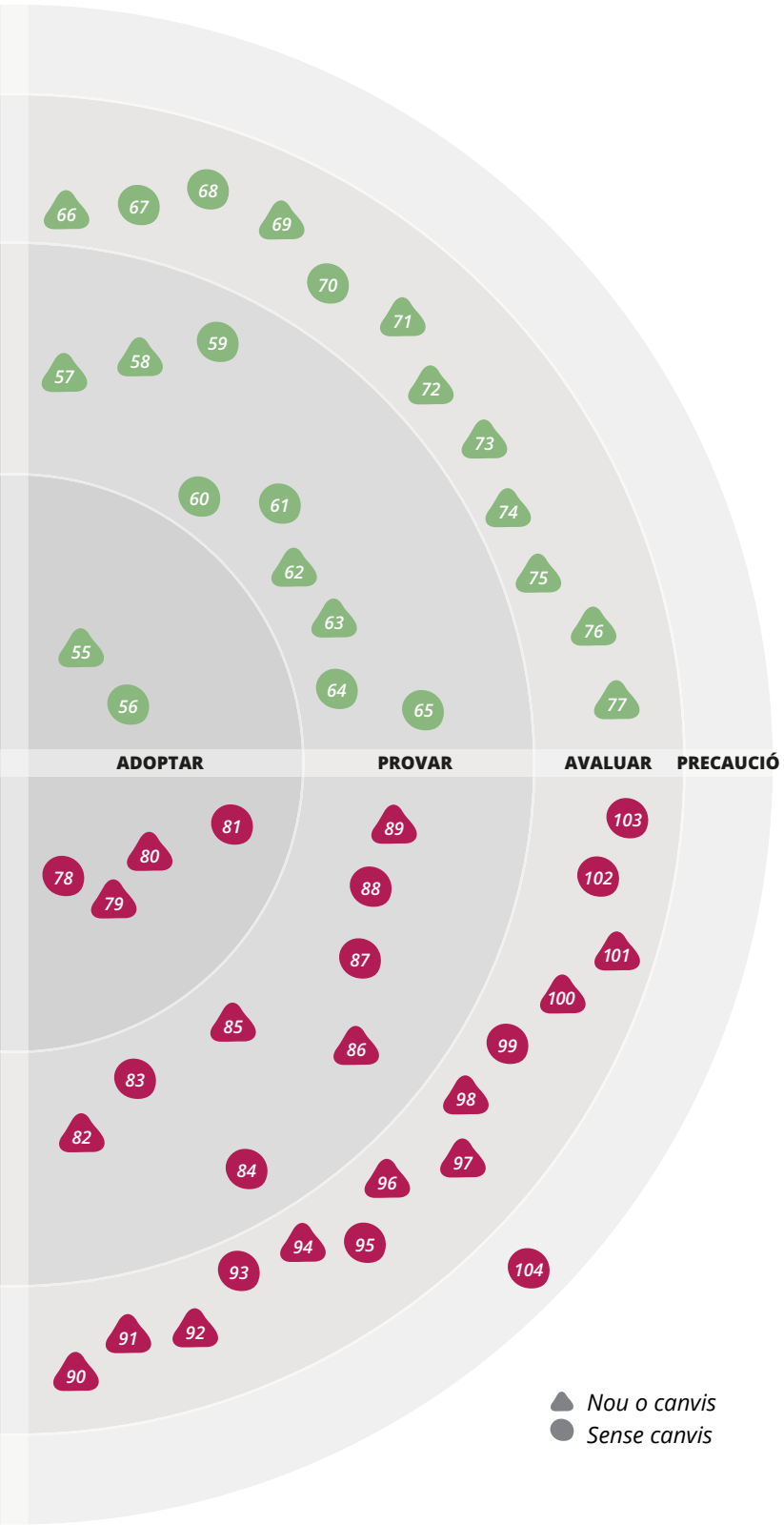
**PROVAR**
26. Apache Mesos
27. Auth0
28. AWS Device Farm *NOU*
29. AWS Lambda
30. OpenTracing *NOU*
31. Unity més enllà dels videojocs

**AVALUAR**
32. .NET Core
33. Amazon API Gateway
34. api.ai *NOU*
35. Cassandra carefully
36. Reconeixement d'imatges al núvol *NOU*
37. DataStax Enterprise Graph *NOU*
38. Electron
39. Ethereum
40. Hyperledger *NOU*
41. IndiaStack
42. Kafka Streams *NOU*
43. Keycloak *NOU*
44. Mesosphere DCOS
45. Mosquitto *NOU*
46. Nuance Mix
47. OpenVR
48. PlatformIO *NOU*
49. Tango *NOU*
50. Voice platforms *NOU*
51. WebVR *NOU*
52. wit.ai

**PRECAUCIÓ**
53. CMS com a plataforma
54. API gateways massa ambicioses



# EL RADAR




## EINES

**ADOPTAR**
55. fastlane
56. Grafana

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

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

**PRECAUCIÓ**

## LLENGUATGES I ENTORNS DE TREBALL

**ADOPTAR**
78. Ember.js
79. Python 3
80. ReactiveX
81. Redux

**PROVAR**
82. Avro *NOU*
83. Elixir
84. Enzyme
85. Hangfire *NOU*
86. Nightwatch *NOU*
87. Phoenix
88. Quick i Nimble
89. Vue.js

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

**PRECAUCIÓ**
104. AngularJS

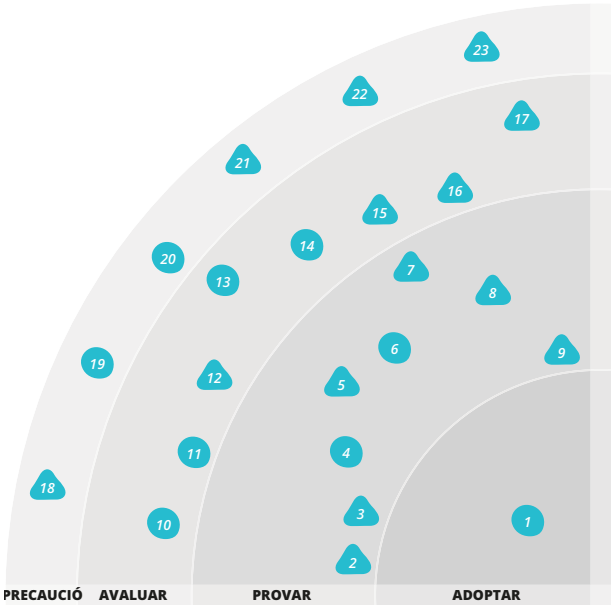
# TÈCNIQUES

**ADOPTAR**
1. Pipelines com a codi

**PROVAR**
2. APIs com a producte
3. Desacoblament de secrets del codi font *NOU*
4. Allotjar dades PII a la UE
5. Sistemes heretats envasats *NOU*
6. Registre de decisions d'arquitectura de baix pes
7. Aplicacions web progressives *NOU*
8. Fer prototips amb InVision i Sketch *NOU*
9. Arquitectura sense servidor

**AVALUAR**
10. Recerca orientada al client
11. Estudi de la seguretat dels contenidors
12. APIs conversacionals *NOU*
13. Privacitat diferencial
14. Micro-interfícies
15. Equips de producte d'enginyeria de plataformes *NOU*
16. Anàlisi de codi social *NOU*
17. Realitat Virtual més enllà dels videojocs

**PRECAUCIÓ**
18. Instància única d'integració continua per a tots els equips
19. REST anèmic
20. Enveja pel Big Data
21. Teatre d'integració continua *NOU*
22. Entorns de proves d'integració a tota l'empresa *NOU*
23. Generació de codi basat en les especificacions *NOU*




Les empreses han adoptat massivament les API com a la manera de demostrar les capacitats de l'empresa tant per a desenvolupadors interns com externs. Les API permeten experimentar sense esforços amb noves idees de l'empresa recombinant els elements essencials. Aleshores, però, què diferencia una API d'un servei ordinari d'integració d'empresa (EIS)? Una de les diferències és que es tracta les **APIS COM A PRODUCTE**, fins i tot quan el consumidor és un sistema intern. Els equips que creen APIs haurien d'entendre les necessitats dels seus clients i fer que el producte sigui adequat per a ells. Les proves d'usabilitat i l'anàlisi de l'experiència de l'usuari poden traduir-se en un millor disseny i en una millor comprensió dels patrons d'ús de les APIs i, així, aportar-hi el pensament de producte. Les APIs, com els productes, s'haurien de mantenir activament i haurien de ser fàcils d'utilitzar. Haurien de tenir un propietari que busca constantment la seva millora i està del costat del consumidor. Segons la nostra experiència, aquest tractament de producte és el que marca la diferència entre una empresa amb una integració ordinària i un negoci àgil construït sobre una base d'APIs.

A Radars anteriors hem mencionat eines, com ara git-crypt i blackbox, que ens permeten mantenir secrets amb seguretat al codi font. **EL DESACOBLAMENT DE SECRETS DEL CODI OBERT** és la nostra manera de recordar als tecnòlegs que hi ha altres opcions per a emmagatzemar secrets. Per exemple, HashiCorp vault, servidors d'integració contínua i eines de gestió de configuració proporcionen mecanismes per a emmagatzemar secrets sense que estiguin enllaçats amb el codi font d'una aplicació. Ambdues opcions són vàlides i nosaltres recomanem que se n'utilitzi com a mínim una en els projectes.

*Els equips que creen APIs haurien d'entendre les necessitats dels seus clients i fer que el producte sigui adequat per a ells. Les proves d'usabilitat i l'anàlisi de l'experiència de l'usuari poden traduir-se en un millor disseny i en una millor comprensió dels patrons d'ús de les APIs i, així, aportar el pensament de producte a les APIs.*

— APIs com a producte

Treballar amb codi heretat, especialment quan és un gran sistema monolític, és una de les experiències més insatisfactòries i pesades per als desenvolupadors. Tot i que recomanem no estendre ni mantenir activament sistemes monolítics heretats, aquests segueixen sent dependències en els nostres entorns i sovint els desenvolupadors subestimen el cost i el temps necessari per a desenvolupar contra aquestes dependències. Per a ajudar a reduir aquesta fricció, els desenvolupadors han utilitzat imatges de màquines virtuals o imatges de contenidors amb contenidors Docker per a crear imatges immutables de sistemes heretats i les seves configuracions. La intenció és contenir els **SISTEMES HERETATS ENVASATS** per a què els desenvolupadors treballin localment i no necessitin reconstruir, reconfigurar o compartir entorns. En un escenari ideal, els equips que tenen sistemes heretats generarien les imatges corresponents a través dels seus canals i els desenvolupadors podrien fer anar i orquestrar aquestes imatges en els seus entorns de proves de manera més fiable. Tot i que aquest enfocament ha reduït el temps de feina de cada desenvolupador, ha tingut un èxit limitat en els casos en què els equips propietaris d'aquestes dependències no han volgut crear contenidors per que els puguin utilitzar els altres.

L'augment d'**APLICACIONS WEB PROGRESSIVES** (PWAs per les seves sigles en anglès) és l'últim intent per a recuperar la web mòbil com a resposta a la "fatiga d'aplicacions" dels usuaris. Originalment proposat per Google l'any 2015, les PWAs són aplicacions web que treuen partit de les últimes tecnologies per a combinar el millor de la web i de les aplicacions natives per a mòbils. Utilitzant una sèrie de tecnologies d'estàndard obert com ara treballadors de serveis, el manifest de l'aplicació i APIs de cache i de tramesa automàtica (push), podem crear aplicacions que funcionen a totes les plataformes i que tenen una experiència d'usuari igual a la d'una aplicació nativa. Això aporta igualtat entre les aplicacions natives i les web i ajuda a què els desenvolupadors puguin arribar als usuaris sense haver de passar per les botigues d'aplicacions. Penseu en les PWAs com a webs que funcionen i semblen aplicacions natives.

L'ús combinat de InVision i Sketch ha canviat la manera com certes persones enfoquen el desenvolupament d'aplicacions web. Tot i que són eines, realment és la tècnica de **FER PROTOTIPS AMB INVISION I SKETCH** el que fa significatiu aquest senyal. Crear prototips rics i que es poden clicar com a punt de partida per a la implementació de comportaments d'interfícies ajuda a accelerar el desenvolupament i a eliminar restes en la implementació. Aquest ús combinat d'aquestes eines permet un equilibri entre l'elaboració prematura dels detalls visuals i el fet d'aconseguir comentaris dels usuaris sobre l'experiència interactiva.

Una **ARQUITECTURA SENSE SERVIDOR** és un enfocament pel qual es canvien les màquines virtuals en funcionament constant per un mode de processament efímer que apareix quan se'l demana i desapareix tant bon punt s'ha deixat d'utilitzar. Als nostres equips els agrada l'arquitectura sense servidor ja que ens funciona bé i la considerem una opció d'arquitectura vàlida. S'ha de tenir en compte, però, que "sense servidor" no té perquè ser un enfocament de "o tot o res". Alguns dels nostres equips han implementat algunes parts dels seus sistemes sense servidor mentre que han seguit utilitzant arquitectures més tradicionals per a d'altres parts. Tot i que AWS Lambda és gairebé sinònim de sense servidor, els principals proveïdors de serveis al núvol tenen ofertes similars i també recomanem avaluar altres actors especialitzats com webtask.

*Tecnologies com Amazon Alexa, Google Voice i Siri han abaixat significativament el llistó pel que fa a interaccions per veu amb programari. No obstant això, pot ser difícil d'aconseguir una interacció més conversacional (ja sigui per escrit o parlat) amb les APIs existents.*

— APIs conversacionals

Tecnologies com Amazon Alexa, Google Voice i Siri han abaixat significativament el llistó pel que fa a interaccions per veu amb programari. No obstant això, pot ser difícil d'aconseguir una interacció més conversacional (ja sigui per escrit o parlat) amb les APIs existents, especialment quan es vol un tipus d'interacció

dinàmica on les següents interaccions han de tenir en compte el context conversacional global. En aquest tipus d'interacció, per exemple, ens agradaria fer algunes preguntes sobre trens que van de Manchester a Glasgow i poder preguntar "a quina hora surt el primer?" sense haver de repetir el context una altra vegada. Normalment, aquest context ja estaria en la resposta del navegador, però, en el cas de les interfícies de veu, aquest context s'ha de tractar en alguna altra banda. Les **APIS CONVERSACIONALS** poden ser un exemple de patró d'interfície (BFF) on la interfície és la veu d'una plataforma de xat. Aquest tipus d'API pot gestionar els detalls d'aquest tipus d'interacció gràcies a la gestió dels estats de la conversa mentre contacta serveis subjacents en nom de la interfícies de veu.

L'adopció del núvol i de DevOps, tot i que incrementa la productivitat dels equips que poden moure's més ràpidament i no depenen tant d'equips o infraestructures centralitzades, també ha limitat els equips que no tenen prou coneixements per a gestionar ells mateixos una aplicació sencera i les seves operacions. Algunes organitzacions han abordat aquest repte creant **EQUIPS DE PRODUCCIÓ DE PLATAFORMES**. Aquests equips operen una plataforma interna que permet que els equips de producte puguin implementar i operar sistemes sense ajuda i, a la vegada, reduir el temps de producció i la complexitat de la memòria. En aquest aspecte, es dóna especialment importància a l'autoservei i a les eines de suport centrades en APIs mentre que els equips de producte segueixen sent responsables del que s'implementa a la plataforma. Les organitzacions que considerin establir una plataforma així haurien d'anar amb molt de compte de no crear un equip separat de DevOps, així com tampoc haurien de reetiquetar la seva estructura d'allotjament i d'operacions existent com a plataforma.

**L'ANÀLISI SOCIAL DEL CODI** enriqueix el nostre coneixement sobre la qualitat del codi superposant el comportament del desenvolupador a l'anàlisi estructural del codi. Utilitza dades de sistemes de control de versions, com ara la freqüència i el moment del canvi o la persona que el va fer. En aquest aspecte, es pot escollir entre escriure el seu propi script o utilitzar eines com CodeScene. CodeScene ens pot ajudar a comprendre millor els sistemes de programari gràcies a la identificació de problemes i de subsistemes difícils de mantenir. Ens permet acoblar subsistemes distribuïts a través de l'acoblament temporal, així com veure la llei de Conway de la nostra organització. Creiem que amb tendències tecnològiques com els sistemes distribuïts, els microserveis i els equips distribuïts, la dimensió social del nostre codi és un aspecte vital en la comprensió holística de la salut del nostre sistema.

La idea de la realitat virtual fa més de 50 anys que ens acompanya i, amb els avenços en la tecnologia dels ordinadors, s'ha explorat moltes idees. Creiem que hem arribat a un punt crític. L'any passat es van posar a la venda cascos de Realitat Virtual raonablement assequibles per al públic i les targetes gràfiques tenen prou potència com per a crear experiències immersives amb ells. Els cascos estan pensats, principalment, per a fanàtics dels videojocs, però nosaltres creiem que obriran la porta per una gran quantitat de possibilitats per a la **REALITAT VIRTUAL MÉS ENLLÀ DELS VIDEOJOCS**. Els equips que no tenen experiència en la creació de videojocs no haurien de subestimar l'esforç i les habilitats necessàries per a crear bons models 3D i textures convincents.

*La idea de la realitat virtual fa més de 50 anys que ens acompanya i, amb els avenços en la tecnologia dels ordinadors, s'ha explorat moltes idees. Creiem que hem arribat a un punt crític.*

— Realitat Virtual més enllà dels videojocs

Ens veiem obligats, una altra vegada, a recomanar precaució pel que fa a la creació d'una **INSTÀNCIA ÚNICA D'INTEGRACIÓ CONTÍNUA** per a tots els equips. Tot i que, en teoria, sembla una bona idea el consolidar i centralitzar una infraestructura IC, en la pràctica no creiem que les eines i productes d'aquest espai siguin prou madures com per a aconseguir els resultats desitjats. Els equips de creació de programari que han d'utilitzar IC centralitzada tenen, sovint, llargues esperes degut a un equip central que ha de dur a terme petites tasques de manteniment o que ha de solucionar problemes en la infraestructura i les eines compartides. Seguim recomanant que les organitzacions limitin l'ús de IC centralitzada i que inverteixin els seus esforços en crear patrons, directrius i suport per a que els seus equips operin la seva pròpia infraestructura de IC.

Fa temps que som defensors de la integració contínua (IC) i som pioners en la creació de programes de servidors IC per a crear projectes automàticament cada vegada que es pugen. Si s'utilitzen adequadament, aquests programes funcionen com a processos daemon en la part principal d'un projecte compartit, el qual els desenvolupadors pugen diàriament. El servidor IC crea el projecte i dur a terme proves completes que asseguren que tot el sistema estigui integrat i estigui, sempre, llest per al seu llençament. D'aquesta manera, doncs, s'aconsegueix satisfer el principi de l'entrega continua. Malauradament, molts desenvolupadors estableixen un servidor IC i creuen, equivocadament, que estan "fent IC" quan, en realitat, s'estan perdent tots els beneficis. Els errors més comuns inclouen: utilitzar IC en un entorn compartit però el qual es puja irregularment, la qual cosa fa que la integració no sigui realment contínua i s'utilitzi una versió amb poques proves; permetre que la versió estigui en vermell durant molt de temps; o utilitzar IC a branques d'utilitats, la qual cosa es tradueix en un aïllament continu. El "**TEATRE IC**"resultant pot ser satisfactori per a algunes persones, però no passaria cap prova de certificació d'IC creïble.

*Malauradament, molts desenvolupadors estableixen un servidor IC i creuen, equivocadament, que estan "fent IC" quan, en realitat, s'estan perdent tots els beneficis. El "TEATRE IC" resultant pot ser satisfactori per a algunes persones, però no passaria cap prova de certificació d'IC creïble.*

— Teatre IC

Quan es considerava que la millor pràctica era el llençament trimestral o mensual, era necessari mantenir un entorn complet per a dur a terme cicles de proves abans d'enviar-ho a producció. Aquests **ENTORNS DE PROVES D'INTEGRACIÓ A TOTA L'EMPRESA** (sovint coneguts com a SIT o staging) són un coll d'ampolla per a l'entrega continua actual. Els entorns són fràgils i cars de mantenir i sovint tenen components que necessiten ser configurats manualment per un equip de gestió separat. Fer proves en aquest entorn proporciona resultats lents i poc fiables, i requereix el doble d'esforç del que requereix fer proves en components aïllats. Nosaltres recomanem que les organitzacions creïn, cada vegada més, un camí independent per a la producció de components clau. Algunes tècniques importants són les proves de contracte, el desplegament de l'acoblament des del llençament, centrar-se en el temps de recuperació d'una fallada (MTTR) i fer proves durant la producció.

Quan SOAP dominava la industria del programari per a empreses, s'acceptava, i fins i tot es fomentava, la pràctica de crear codi client des de WSDL. Malauradament, el codi resultant era, sovint, complex, inestable, difícil de modificar i no funcionava a plataformes d'implementació. Amb l'arribada de REST, vam creure que era millor evolucionar clients d'API que utilitzen el patró de lectura tolerant per a extreure i processar només els camps necessaris. Recentment, hem vist un retorn pertorbant cap a antics costums on els desenvolupadors generen codi des d'especificacions API escrites en Swagger o RAML. Nosaltres, anomenem aquesta pràctica com a **GENERACIÓ DE CODI BASAT EN LES ESPECIFICACIONS**. Tot i que aquestes eines són molt útils per a propulsar el disseny d'APIs i per a extreure documentació, recomanem evitar caure en la temptació de crear codi client directament des d'aquestes especificacions. Hi ha moltes possibilitats de que el codi sigui difícil de provar i de mantenir.

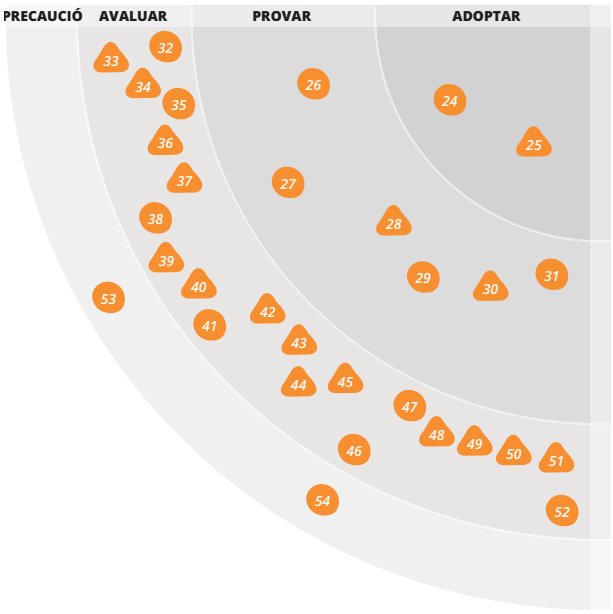
# PLATAFORMES

**ADOPTAR**
24. HSTS
25. Linux Security Modules

**PROVAR**
26. Apache Mesos
27. Auth0
28. AWS Device Farm *NOU*
29. AWS Lambda
30. OpenTracing *NOU*
31. Unity més enllà dels videojocs

**AVALUAR**
32. .NET Core
33. Amazon API Gateway
34. api.ai *NOU*
35. Cassandra carefully
36. Reconeixement d'imatges al núvol *NOU*
37. DataStax Enterprise Graph *NOU*
38. Electron
39. Ethereum
40. Hyperledger *NOU*
41. IndiaStack
42. Kafka Streams *NOU*
43. Keycloak *NOU*
44. Mesosphere DCOS
45. Mosquitto *NOU*
46. Nuance Mix
47. OpenVR
48. PlatformIO *NOU*
49. Tango *NOU*
50. Plataformes de veu *NOU*
51. WebVR *NOU*
52. wit.ai

**PRECAUCIÓ**
53. CMS com a plataforma
54. API gateways massa ambicioses




El Principi de Privilegi Mínim ens anima a restringir els components de programari per a que accedeixin només als recursos que necessiten. Per defecte, però, un procés de Linux pot fer tot el que l'usuari que l'està utilitzant pugui fer: des d'associar-se amb ports arbitraris fins a generar noves shells. L'entorn de treball de **LINUX SECURITY MODULES** (LSM per les seves sigles en anglès), el qual permet que es connectin extensions de seguretat directament al nucli del sistema operatiu (o kernel), s'ha utilitzat per a implementar MAC a Linux. SELinux i AppArmor són les implementacions compatibles amb LSM més conegudes i predominants i venen incloses al kernel. Recomanem que els equips aprenguin a utilitzar algun d'aquests entorns de seguretat (i és per això que l'hem posat a l'anell Adoptar). Ajuda els equips a avaluar preguntes sobre qui té accés a quins recursos a servidors compartits, incloent serveis en contenidors. Aquest enfocament conservador d'accés a la gestió ajudarà els equips a millorar la seguretat dels seus processos SDLC. .

**L'AMAZON API GATEWAY** permet als desenvolupadors d'exposar serveis d'API a clients d'internet. Ofereix totes les característiques normals de l'API Gateway com ara la gestió de trànsit, la
monitorització, l'autenticació i l'autorització. Els nostres equips han tingut experiències positives utilitzant aquest servei per a incloure AWS Lambda com a part d'arquitectures sense servidor. D'altra banda, ens ha costat més utilitzar-la com a passarel·la més general per a punts d'accés HTTP/HTTPS que utilitzen EC2, ja que hem vist una falta d'interoperabilitat amb VPCs i hem tingut problemes per a establir certificats d'autenticació de clients amb la passarel·la. Degut a aquestes experiències, ens agradaria recomanar als equips de provar d'utilitzar AWS API Gateway amb Lambda, però d'anar amb compte quan s'utilitzi en entorns més generals.

La gran quantitat de dispositius mòbils fa gairebé impossible que les empreses puguin provar les seves aplicacions mòbils a cada dispositiu. Benvinguts a **AWS DEVICE FARM**, un servei de proves d'aplicacions que permet fer funcionar i interactuar amb les aplicacions web, d'Android i d'iOS en una gran varietat de dispositius allotjats al núvol simultàniament. Cada vegada que s'utilitza, es generen registres detallats, gràfics de rendiment i captures de pantalla per a proporcionar un visió específica del rendiment de l'aplicació per a cada dispositiu. Aquest servei ofereix molta flexibilitat ja que permet que es pugui canviar l'estat i la configuració de cada dispositiu per a poder reproduir escenaris de proves molt específics. Els nostres equips utilitzen AWS Device Farm per a dur a terme proves completes als dispositius que més han descarregat les seves aplicacions.

*Ara que les aplicacions monolítiques s'estan canviant per ecosistemes de microserveis més complexos, les peticions de seguiment a través de múltiples serveis s'està convertint en la norma.*

— OpenTracing

Ara que les aplicacions monolítiques s'estan canviant per ecosistemes de (micro)serveis més complexos, les peticions de traçament a través de múltiples serveis s'està convertint en la norma. Afortunadament, **OPENTRACING** s'està convertint ràpidament en l'estàndard en el traçament distribuït. Desenvolupat per Uber, Apple, Yelp i d'altres grans empreses, suporta múltiples traçadors com Zipkin, Instana i Jaeger. OpenTracing proporciona, actualment, una implementació neutra en sis idiomes: Go, JavaScript, Java, Python, Objective-C i C++.

Paral·lelament al recent increment dels bots conversacionals i de les plataformes de veu, hem vist una proliferació de plataformes, com **API.AI**, que proporcionen un servei per a extreure la intenció d'un text i gestionar el flux de la conversa. Recentment adquirit per Google, aquesta "comprensió del llenguatge natural com a servei" competeix amb altres participants d'aquest món com wit.ai i el Lex d'Amazon.

El reconeixement d'imatges acostumava a ser un art obscur que necessitava tot un equip de científics. En els últims anys, però, ens hem acostat a trobar una solució per als problemes de classificació i categorització de cares i imatges, comparacions facials, identificació de característiques facials i reconeixement facial. El **RECONEIXEMENT D'IMATGES AL NÚVOL** dóna accés a capacitats d'aprenentatge automàtic gràcies a serveis com Amazon Rekognition, Microsoft Computer Vision API i Google Cloud Vision API, els quals poden complementar aplicacions de RA i qualsevol cosa que tingui a veure amb l'etiquetatge i la classificació d'imatges.

Hem tingut algunes bones experiències amb **DATASTAX ENTERPRISE GRAPH** (DSE Graph) a l'hora de tractar grans bases de dades gràfiques. Construït a sobre de Cassandra, DSE Graph es centra en les grans col·leccions de dades on la nostra eina predilecta, Neo4j, ja comença a mostrar les seves limitacions. Aquesta escala, però, té els seus inconvenients. Per exemple, es perden les transaccions ACID i el temps d'execució sense esquema de Neo4j. No obstant això, l'accés a les taules inferiors de Cassandra, la integració de Spark per a feines analítiques i el poderós llenguatge d'interrogació Tinkerpop/Gremlin, fan que sigui una opció digna de ser considerada.

L'excitació per les cadenes de blocs i la moneda digital sembla que ha arribat al seu màxim com es pot veure pel descens en els anuncis sobre aquest tema i anticipem que alguns dels esforços més especulatius acabaran morint. **ETHEREUM**, una de les cadenes de blocs, està progressant positivament i és digne de veure. Ethereum és una cadena de blocs pública i creada amb un llenguatge de programació que permet que s'hi construeixin "contractes intel·ligents". Són moviments algorítmics de "ether" (la moneda digital de Ethereum) com a resposta a l'activitat que té lloc a la cadena de blocs. R3Cev, el consorci que construeix la tecnologia de cadena de blocs pels bancs, va crear a Ethereum la seva primera prova de concepte. Ethereum s'ha utilitzat per crear una Organització Autònoma Descentralitzada (DAO per les seves sigles en anglès), una de les primeres "corporacions algorítmiques", però un robatori de Ether per valor de 150 milions de dólars demostra que la cadena de blocs segueix sent el "Far West" del món de la tecnologia.

**HYPERLEDGER** és una plataforma construïda sobre tecnologies de cadenes de blocs. Està constituïda per una implementació de cadena de blocs anomenada fabric i per d'altres eines associades. Sense tenir en compte l'excitació sobre les cadenes de blocs, els nostres equips creuen que és fàcil de començar a utilitzar aquestes eines. El fet de que sigui una plataforma de codi obert recolzada per la Linux Foundation no fa més que augmentar encara més el nostre excitament per Hyperledger.

**KAFKA STREAMS** és una llibreria de baix pes per a crear aplicacions de streaming. S'ha dissenyat amb l'objectiu de simplificar prou el processament de serveis de streaming per tal de fer-ho fàcilment accessible com un model establert de programació d'aplicacions per a serveis asíncrons. Pot ser una bona alternativa en els casos on es vol aplicar un model de processament de streaming al seu problema sense haver de recórrer a un clúster (normalment introduïts per entorns de processament de streaming totalment desenvolupats).

Als microserveis, o a qualsevol altre arquitectura distribuïda, una de les necessitats més comunes és la d'assegurar els serveis o APIs a través d'opcions d'autenticació i autorització. Aquí és on Keycloak entra en escena. **KEYCLOAK** és una solució de gestió d'identitat i d'accés que facilita el fet d'assegurar aplicacions o microserveis sense cap, o gairebé cap, codi. Directe de fàbrica, permet l'inici de sessió individual, a través les dades de xarxes socials i protocols estàndard com ara OpenId Connect, Oauth2 i SAML.

**MESOSPHERE DCOS** és una plataforma construïda sobre Mesos que abstreu les infraestructures subjacents tant per a les aplicacions en contenidors com per a les aplicacions que no es volen fer anar a Docker. Això podria ser massa per desplegaments més modestos, però hem començat a tenir èxit tant amb la versió comercial com amb la de codi obert. Ens agrada especialment que facilita la portabilitat entre diferent proveïdors de serveis al núvol, així com amb components dedicats, i que per a feines amb contenidors, no limita a utilitzar un sol entorn d'organització de contenidors. Tot i que actualitzar-ho és una mica més complex del que ens agradaria, en general s'està estabilitzant correctament.

Segons la nostra experiència, el protocol MQTT és una gran opció per a solucions de la Internet de les coses (IoT per les seves sigles en anglès) on un munt de dispositius es comuniquen entre ells i/o amb un servidor central. També ens ha agradat l'agent MQTT **MOSQUITTO**. Pot no satisfer totes les peticions, especialment pel que fa a l'escalabilitat, però la seva naturalesa compacta i la seva fàcil configuració el fan ideal per a finalitats de desenvolupament i proves.

**PLATFORMIO** proporciona un ric ecosistema per al desenvolupament de IoT gràcies a versions multiplataforma, gestió de llibreries i una bona integració amb IDEs existents. La finalització i neteja intel·ligent de codi amb monitor incorporat de terminal i de port en sèrie milloren l'experiència del desenvolupador. També organitza i manté milers de llibreries i proporciona un gestor de dependències amb versionatge semàntic per a facilitar el desenvolupament d'IoT. Hem començat a utilitzar PlatformIO en alguns projectes d'IoT i ens agrada molt la seva simplicitat i el gran suport que té per a plataformes i plaques.

Conjuntament amb la realitat virtual (RV), que té un elevat cost per culpa dels components requerits i l'esforç que suposa crear móns virtuals, la realitat alternativa (RA) i la realitat mixta (RM), també van entrar al mercat global l'any passat. En aquest sentit, Pokemon Go va evidenciar que un telèfon intel·ligent normal és suficient per a crear mons de RA/RM prou convincents. **TANGO** és una nova tecnologia de sensors per a components mòbils que millora les possibilitats de RA/RM als telèfons mòbils. Permet que les aplicacions tinguin una imatge 3D del que envolta l'usuari per tal de permetre que els objectes virtuals es puguin col·locar de manera més convincent en la pantalla del telèfon. Ja estan disponibles els primers telèfons amb aquesta tecnologia.

*Ja hem començat a integrar interfícies d'usuari conversacionals als nostres productes i ja veiem l'impacte d'aquesta nova interacció en com dissenyem les interfícies.*

— Plataformes de veu

Les **PLATAFORMES DE VEU** com ara Amazon Alexa i Google Home estan creant molta excitació. Tanta, que n'hi ha que fins i tot pregonen l'omnipresència de la interfície de veu conversacional. Ja hem començat a integrar interfícies d'usuari conversacionals als nostres productes i ja veiem l'impacte d'aquesta nova interacció en com dissenyem les interfícies. Alexa, especialment, es va crear des del principi sense cap tipus de pantalla per la qual cosa la seva interfície d'usuari conversacional és de primer nivell. Encara és massa aviat, però, per creure'ns tota aquesta emoció i esperem que més empreses entrin en aquest mercat.

**WEBVR** és una API de JavaScript experimental que permet accedir a dispositius de RV des del navegador. Ha aconseguit suport de la comunitat i està disponible en versions de prova diàries i en algunes versions finals. Si es busca crear experiències de RV al navegador, aquest un molt bon lloc pel que començar. Aquesta tecnologia, conjuntament amb eines complementàries com Three.js, A-Frame, ReactVR, Argon.js i Awe.js, aporta la RA al navegador. La quantitat d'eines que existeixen en aquest món i els estàndards de la comissió d'Internet podrien ajudar a enfortir l'adopció de RA i RV.

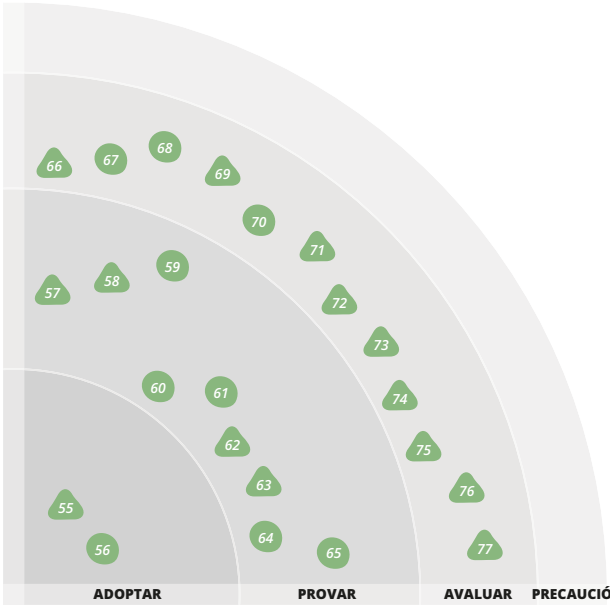
# EINES

**ADOPTAR**
55. fastlane
56. Grafana

**PROVAR**
57. Airflow *NOU*
58. Cake i Fake *NOU*
59. Galen
60. HashiCorp Vault
61. Pa11y
62. Scikit-learn
63. Entorn de treball sense servidor *NOU*
64. Talisman
65. Terraform

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

**PRECAUCIÓ**




Els desenvolupadors d'aplicacions web ho tenen fàcil pel que fa a simplificar i automatitzar diverses feines. Poden escollir entre una varietat de solucions que els ajuden a automatitzar processos de lliurament. Quan es desenvolupa per a mòbils, però, es troben amb dos sistemes operatius diferents que tenen maneres diferents de crear, provar, crear captures de pantalla, firmar i distribuir aplicacions. Per ajudar una mica, els nostres equips han començat a utilitzar **FASTLANE** per a automatitzar el procés de lliurament d'aplicacions d'Android i iOS. Han aconseguit, utilitzant configuracions simples i múltiples canals, Continuous Delivery per al desenvolupament mòbil.

**AIRFLOW** és una eina per a crear, programar i monitoritzar canals de dades de manera programada. Tractant els grafs acíclics dirigits (DAGs per les seves sigles en anglès) com a codi, fomenta els canals de dades que es poden mantenir, versionar i provar. Hem utilitzat aquesta configuració en els nostres projectes i hem pogut crear canals dinàmics que han tingut com a resultat fluxos de dades explícits i lean.

*Els nostres equips han començat a utilitzar fastlane per a automatitzar el procés de lliurament d'aplicacions d'Android i iOS.*

— fastlane

MSBuild ha sigut el principal sistema de creació a l'ecosistema .NET des que es va presentar l'any 2005. No obstant això, té moltes de les febleses que ja hem mencionat sobre Maven. La comunitat .NET ha començat a desenvolupar alternatives a MSBuild que són més fàcils de mantenir, més flexibles i que evolucionen de manera més fluida a mida que un projecte creix. Dos d'aquestes alternatives són **CAKE I FAKE**. Cake utilitza un DSL construït en C#, mentre

que Fake utilitza F#. Ambdues opcions han crescut considerablement durant l'últim any i han provat ser alternatives viables a MSBuild per a gestionar tasques de creació comuna a projectes .NET.

*La comunitat .NET ha començat a desenvolupar alternatives a MSBuild que són més fàcils de mantenir, més flexibles i que evolucionen de manera més fluida a mida que un projecte creix.*

— Cake i Fake

**SCIKIT-LEARN** no és una eina nova (s'acosta al seu desè aniversari), però el que sí que ho és el grau d'adopció d'eines i tècniques d'aprenentatge automàtic més enllà del món acadèmic i de les companyies tecnològiques més grans. Scikit-learn proporciona una robusta col·lecció de models i funcionalitats i juga un paper important en fer que els conceptes i capacitats d'aprenentatge automàtic puguin arribar a un públic més ampli (i, sovint, no expert).

El popular **SERVERLESS FRAMEWORK** proporciona eines per a la creació de l'esquelet i el lliurament d'aplicacions sense servidor utilitzant, principalment, AWS Lambda i d'altres opcions d'AWS. L'entorn de treball sense servidor té suport de plantilles per a JavaScript, Python, Java i C# i compta amb una comunitat molt activa que proporciona extensions per a l'entorn de treball. A més, aquest entorn també accepta el projecte incubador d'Apache OpenWhisk com a alternativa a AWS Lambda.

**AMAZON REKOGNITION** és una de les eines de reconeixement d'imatges al núvol que hem mencionat en aquest Radar. El que ens agrada és que Amazon ha seguit un enfocament lleugerament nou pel qual les cares són anònimes per a l'AWS (mitjançant GUIDS), la qual cosa redueix algunes de les preocupacions sobre privacitat lligades al reconeixement facial.

La combinació d'AWS Lambda amb Amazon API Gateway ha tingut un gran impacte sobre com despleguem serveis i APIs. Fins i tot en aquesta configuració sense servidor, però, la quantitat de configuració necessària per a que tot quedi ben lligat no és pas poca. **CLAUDIA** és una eina que automatitza el desplegament de funcions AWS Lambda escrites en JavaScript i de les configuracions d'API Gateway associades. Proporciona configuracions per defecte raonables i els nostres equips han vist que els permet de començar a treballar ràpidament amb microserveis basats en Lambda.

Com pot, un negoci, donar autonomia als seus equips assegurant-se que els productes creats siguin segurs i segueixin les directrius establertes? Com ens podem assegurar que els servidors, un cop implementats, segueixen sent segurs i adequats durant tota la seva vida útil? Aquests són els problemes que InSpec intenta solucionar. **INSPEC** és una eina de proves d'infraestructures inspirada per Serverspec, però amb algunes modificacions que la fan més útil per a professionals de la seguretat que han d'assegurar-se que tot funciona correctament a milers de servidors. Es poden combinar proves individuals amb proves completes i fer-ho tot remotament amb una línia d'ordres. Inspec és útil per als desenvolupadors però serveix, a la vegada, per a provar contínuament infraestructures desplegades, anant cap a l'Assegurament de la Qualitat durant la producció.

**MOLECULE** està dissenyat per a ajudar en el desenvolupament en les proves dels papers d'Ansible. Creant l'esquelet per a fer proves dels papers d'Ansible en una màquina virtual o contenidor de la nostra elecció, no és necessari establir el nostre entorn de proves de manera manual. Molecule utilitza Vagrant, Docker i OpenStack per a gestionar màquines virtuals/contenidors i suporta Serverspec, Testinfra o Goss facin les proves. Els passos per defecte al model de seqüencia de serveis inclou: gestió de màquines virtuals, neteja ansible, proves d'idempotència i proves de convergència. Tot i ser un projecte bastant jove, creiem que té un gran potencial.

Com diria qualsevol dels seus fans, Emacs és més que un editor de text. És una plataforma per a aplicacions de caràcters assignats. Durant els últims anys hi ha hagut una explosió de nous desenvolupaments en aquesta plataforma, però creiem que **SPACEMACS** es mereix la nostra atenció. Spacemacs és una introducció a la plataforma Emacs amb una nova interfície de teclat, capes de personalització simplificades i una distribució feta amb cura de paquets d'Emacs. Un dels objectius del projecte és el de ser el millor dels dos mons gràcies a la combinació de la interfície d'usuari vim amb les capacitats de reprogramació interna d'Emacs. Considerem que les eines de productivitat per als desenvolupadors són una part vital del desenvolupament de programari efectiu i si fa temps que no es considera Emacs com a opció, recomanem donar un cop d'ull a com Spacemacs ha repensat aquesta clàssica plataforma de desenvolupament.

**SPACY** és una llibreria de processament de llenguatge natural (NLP per les seves sigles en anglès) escrita en Python. És una llibreria d'alt rendiment, pensada per ser utilitzada pels desenvolupadors en producció, la qual aplica models de NLP per al processament de text que sovint barreja emoticones i signes de puntuació inconsistents. A diferència d'altres entorns de NLP, spaCy és una plataforma endollable i no una plataforma. S'ha pensat per a aplicacions en producció enlloc de per a models de formació per a recerca. Funciona bé amb TensorFlow i la resta de l'ecosistema d'IA de Python. Hem utilitzat Python en el context de l'empresa per a crear un motor de cerca que agafa el llenguatge humà i ajuda els usuaris a prendre decisions comercials.

Netflix ha fet que la seva plataforma de microserveis d'entrega contínua (CD per les seves sigles en anglès) **SPINNAKER** passi a ser de codi obert. En comparació amb altres plataformes de CI/CD, Spinnaker implementa la gestió del clúster i el lliurament d'imatges al núvol com a característiques de primer nivell. Suporta, de fàbrica, el lliurament i la gestió de clústers per a múltiples proveïdors de serveis al núvol com Google Cloud Platform, AWS i Pivotal Cloud Foundry. Es pot integrar Spinnaker a Jenkins per a una tasca de Jenkins. Ens agrada l'enfocament obstinat de Jenkins per al desplegament de microserveis al núvol, amb l'excepció de que els seus canals es creen a través d'una interfície d'usuari (UI per les seves sigles en anglès) i no es poden configurar com a codi.

Vist l'ús generalitzat de les eines de gestió d'infraestructures, no és cap sorpresa que hagi augmentat l'ús de la infraestructura com a codi a projectes recents. Amb aquesta tendència, doncs, es crea la necessitat de provar aquest codi. Amb **TESTINFRA** és possible de provar l'estat actual dels servidors ja s'hagin configurat manualment o amb eines com Ansible, Puppet o Docker. Testinfra vol ser l'equivalent a Python de Serverspec i s'ha escrit com un plugin per al motor de proves Pytest.

*Vist l'ús generalitzat de les eines de gestió d'infraestructures, no és cap sorpresa que hagi augmentat l'ús de la infraestructura com a codi a projectes recents.*

— Testinfra

**YARN** és un nou gestor de paquets que reemplaça l'actual flux de treball per al client npm però mantenint la compatibilitat amb el registre npm. Amb el client npm és possible que s'acabi amb una estructura de la carpeta node_modules basada en l'ordre en què s'ha instal·lat les dependències. Aquesta naturalesa no determinista pot crear problemes de "funciona al meu ordinador". Trencant els passos d'instal·lació en tres (resolució, recuperació i vinculació), Yarn evita aquests problemes utilitzant algoritmes deterministes i lockfiles, la qual cosa permet repetir instal·lacions. A més, hem vist velocitats significativament més altes en els nostres entorns d'integració contínua gràcies al fet que Yarn encaua tots els paquets que descarrega.

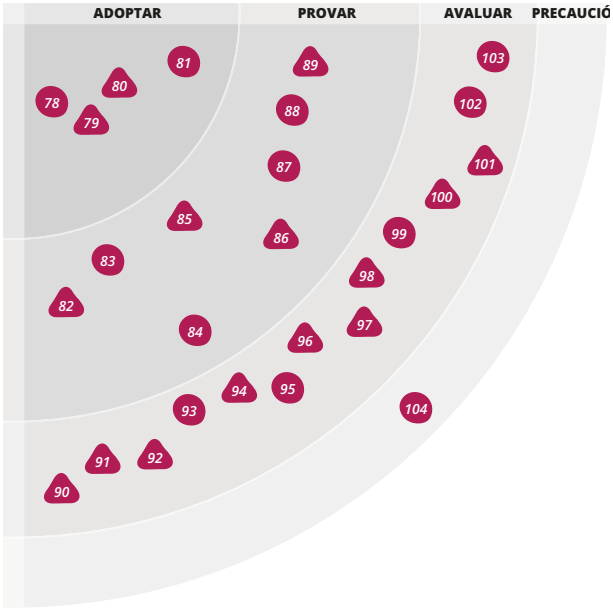
# LLENGUATGES I ENTORNS DE TREBALL

**ADOPT**
78. Ember.js
79. Python 3
80. ReactiveX
81. Redux

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

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

**HOLD**
104. AngularJS




**PYTHON 3** ha introduït moltes característiques útils que no són compatibles amb Python 2.x. També ha eliminat diverses característiques que es mantenien per a la compatibilitat amb versions anterior aconseguint, així, ser més fàcil d'aprendre i més consistent amb la resta del llenguatge. La nostra experiència utilitzant Python 3 en els camps de l'aprenentatge automàtic i el desenvolupament d'aplicacions web demostra que tant el llenguatge com la majoria de les seves llibreries han madurat prou com per a adoptar-lo. Hem pogut solucionar problemes menors a llibreries existents o evitar llibreries incompatibles de Python 2.x que ja s'havien abandonat. Si es desenvolupa utilitzant Python, recomanem passar-se a Python 3.

Els sistemes distribuïts sovint utilitzen la comunicació multifil i basada en esdeveniments i E/S no bloquejant per a millorar l'eficiència global del sistema. Aquestes tècniques de programació creen reptes com els subprocessos de baix nivell, la sincronització, la seguretat dels processos, les estructures de dades concurrents i la E/S no bloquejant. La llibreria de codi obert **REACTIVEX** elimina aquests problemes, proporciona els components requerits per l'aplicació i estén el patró observable en processos d'esdeveniments asíncrons. ReactiveX també compta amb una comunitat de desenvolupadors molt activa i suporta una llista cada vegada més gran de llenguatges, sent RxSwift la seva última incorporació. També permet vincular-se amb plataformes mòbils i de sobretaula.

*La llibreria de codi obert REACTIVEX elimina aquests problemes (subprocessos de baix nivell, sincronització, seguretat dels processos, estructures de dades concurrents i E/S no bloquejant) i proporciona els components requerits per l'aplicació i estén el patró observable en processos d'esdeveniments asíncrons.*

— ReactiveX

**AVRO** és un entorn de treball creat per a la publicació de dades per entregues. Emmagatzemant l'esquema conjuntament amb el text, fomenta l'evolució de l'esquema. Els productors poden editar els noms dels camps, incorporar nous camps o eliminar-ne d'existents i Avro garanteix que els clients continuïn a consumir els missatges. Tenir un esquema permet escriure cada dada sense sobrecàrrega, la qual cosa es tradueix en una codificació més compacta de les dades i en un processament més ràpid de les dades. Tot i que l'intercanvi de missatges sense estructura entre el productor i els consumidor és flexible, hem vist equips tenir problemes amb missatges no processats incompatibles a la cua durant la implementació. Hem utilitzat Avro en diversos projectes i ho recomanaríem com a alternativa a enviar missatges sense estructura.

*Hangfire és fàcil d'utilitzar i flexible i, a més, adopta un estil funcional. És particular ment interessant la seva habilitat de guardar l'estat d'una tasca per a poder continuar quan una aplicació es reinicia després que es penji o es tanqui.*

— Hangfire

Un problema comú en el desenvolupament d'aplicacions és com programar tasques que corrin més enllà del procés principal de manera periòdica o quan es donen certes condicions. El problema s'agreuja quan hi ha esdeveniments inesperats com ara el tancament de l'aplicació. Tal com han descobert els nostres equips, l'entorn de treball **HANGFIRE** pot fer això, i molt més, en l'entorn .NET. Hangfire és fàcil d'utilitzar i flexible i, a més, adopta un estil funcional. És particularment interessant la seva habilitat de guardar l'estat d'una tasca per a poder continuar quan una aplicació es reinicia després que es pengi o es tanqui.

**NIGHTWATCH** és un entorn de treball que permet crear proves d'acceptació automatitzades al navegador en JavaScript i que corren en Node.js. Nightwatch permet definir les proves utilitzant una API fluida que es pot executar contra un servidor Selenium/Webdriver. En el cas d'aplicacions d'una sola pàgina o de pàgines amb molt de JavaScript, ens permet crear i fer proves automatitzades utilitzant el mateix llenguatge i entorn que la majoria del codi.

En el món sempre canviant dels entorns d'interfícies en JavaScript, un dels favorits sembla ser **VUE.JS**. Vue.js és una alternativa de baix pes a Angular.js. S'ha dissenyat per a ser una llibreria molt flexible i menys tossuda que ofereix una sèrie d'eines per a crear interfícies web seguint els conceptes de modularitat, components i flux de dades reactiu. Té una corba d'aprenentatge baixa, la qual cosa el fa interessant per a desenvolupadors júniors o principiants. S'ha de tenir en compte, però, Vue.js no és un entorn de treball complet ja que només es centra en la capa de visualització i, per tant, és fàcil d'integrar amb d'altres llibreries o projectes existents.

Al Radar anterior vam posar AngularJs a l'anell de Precaució (i hi segueix estant en aquesta edició). Pel que fa a **ANGULAR 2**, veiem missatges contradictoris. Durant l'últim any, alguns dels nostres equips l'han utilitzat i ho consideren una opció molt vàlida. No obstant això, Angular 2 és una actualització de AngularJs, no una evolució, i canviar de AngularJs a Angular 2 no és més diferent que passar de AngularJs a un altre entorn. Tenint en compte que hi ha altres opcions millors, segons la nostra experiència, com ara React.js, Ember.js o Vue.js, ens segueix costant recomanar fermament Angular 2. Volem remarcar, però, que no és una mala opció, especialment si s'utilitza TypeScript.

**CAFFE** és una llibreria de codi obert per a l'aprenentatge profund creada pel Berkeley Vision and Learning Center. Es centra, principalment, en xarxes convolucionals per a aplicacions de visió artificial. Caffe és una opció sòlida per a tasques de visió artificial i es poden descarregar molts models fets per usuaris de Caffe des del Caffe Model Zoo per a poder utilitzar-lo des del primer moment. Igual que Keras, Caffe és una API basada en Python. A Keras, però, els models i els components són objectes creats directament en codi Python, mentre que els models de Caffe es descriuen per arxius de configuració Protobuf. Ambdues opcions tenen els seus punts a favor i en contra i és possible convertir-les a l'altra opció.

**DEEPLEARNING.SCALA** és un joc d'eines d'aprenentatge profund i de codi obert escrit en Scala i creat pels nostres col·legues de ThoughtWorks. Creiem que és interessant perquè utilitza la programació funcional diferenciable per a crear i composar xarxes neutres: un desenvolupador simplement escriu codi en Scala utilitzant tipatge estàtic. Deeplearning. scala suporta, actualment, tipatges bàsics com float, double, matrius N-dimensionals accelerades per GPU i tipatges de dades algebraics. Esperem amb ganes les

actualitzacions d'aquest joc d'eines que, sembla, tindran suport per a funcions superiors i entrenament distribuït a Spark.

**INSTANA** és una altra nova opció en el món de la gestió del rendiment d'aplicacions. El fet que s'hagi creat des de 0 per a arquitectures al núvol natives el diferencia de la majoria dels seus competidors. Algunes característiques inclouen la descoberta dinàmica, la cerca distribuïda i la salut del servei, a més de permetre "viatjar en el temps" fins al punt exacte on hi va haver un incident. Encara s'ha de veure si aquest producte pot guanyar protagonisme d'entre tots els projectes de codi obert que fan el mateix, com ara Consul, Prometheus i les implementacions de OpenTracing. No obstant això, val la pena donar-hi un cop d'ull si el que es busca és una solució que funcioni de fàbrica.

**KERAS** és una interfície d'alt nivell en Python per a crear xarxes neutres. Creat per un enginyer de Google, Keras és de codi obert i córrer a sobre de TensorFlow o Theano. Proporcina una interfície increïblement simple per a crear poderosos algoritmes d'aprenentatge profund per a l'entrenament a CPUs o GPUs. Keras s'ha dissenyat amb la modularitat, la simplicitat i l'extensibilitat en ment. A diferència de llibreries com Caffe, Keras té suport per a arquitectures de xarxes més generals, com ara xarxes recurrents, la qual cosa el fa més útil per a l'anàlisi de text, NLP i l'aprenentatge automàtic general. Si la visió artificial, o qualsevol altre branca especialitzada de l'aprenentatge profund, és la principal preocupació, Caffe podria ser una millor opció. Ara bé, si el que es busca és aprendre un entorn de treball més simple però poderós, Keras hauria de ser la primera opció.

**KNET.JL** és l'entorn d'aprenentatge profund de la Universitat de Koç implementat en Julia per Denis Yuret i els seus col·laboradors. A diferència de compiladors com Theano o TensorFlow, que limiten l'usuari a mini-llenguatges, Knet permet definir i entrenar els models d'aprenentatge automàtic utilitzant tot el poder i l'expressivitat de Julia. Knet utilitza gràfics computacionals dinàmics generats durant l'execució per a la diferenciació automàtica de gairebé qualsevol codi Julia. Ens agrada molt el suport de les operacions de GPU a través del KnetArray i, en els casos en què no es té accés a una GPU, l'equip de knet també manté una Amazon Machine Image (AMI) preconfigurada per a poder avaluar-ho al núvol.

La majoria dels nostres desenvolupadors volen avaluar el llenguatge de programació **KOTLIN** i alguns ja l'han utilitzat satisfactòriament. És un llenguatge JVM de codi obert de JetBrains. Als nostres desenvolupadors Swift els agrada perquè és sintàcticament proper a Swift i igual de concís. Als nostres desenvolupadors Java els agrada la seva interoperabilitat amb el llenguatge i les eines Java i han vist que era més senzill que Scala. Kotlin té suport per a conceptes de programació funcional, però amb menys opcions que Scala. Els desenvolupadors dels nostres equips a qui els agrada el tipatge estàtic amb el compilador trobant defectes puntuals nuls, havien d'escriure menys proves models.

*Els equips que creen sistemes composats per microserveis han de pensar en tècniques de coordinació com el descobriment del servei, l'equilibri de la càrrega, el trencament del circuit i les proves de salut.*

— Spring Cloud

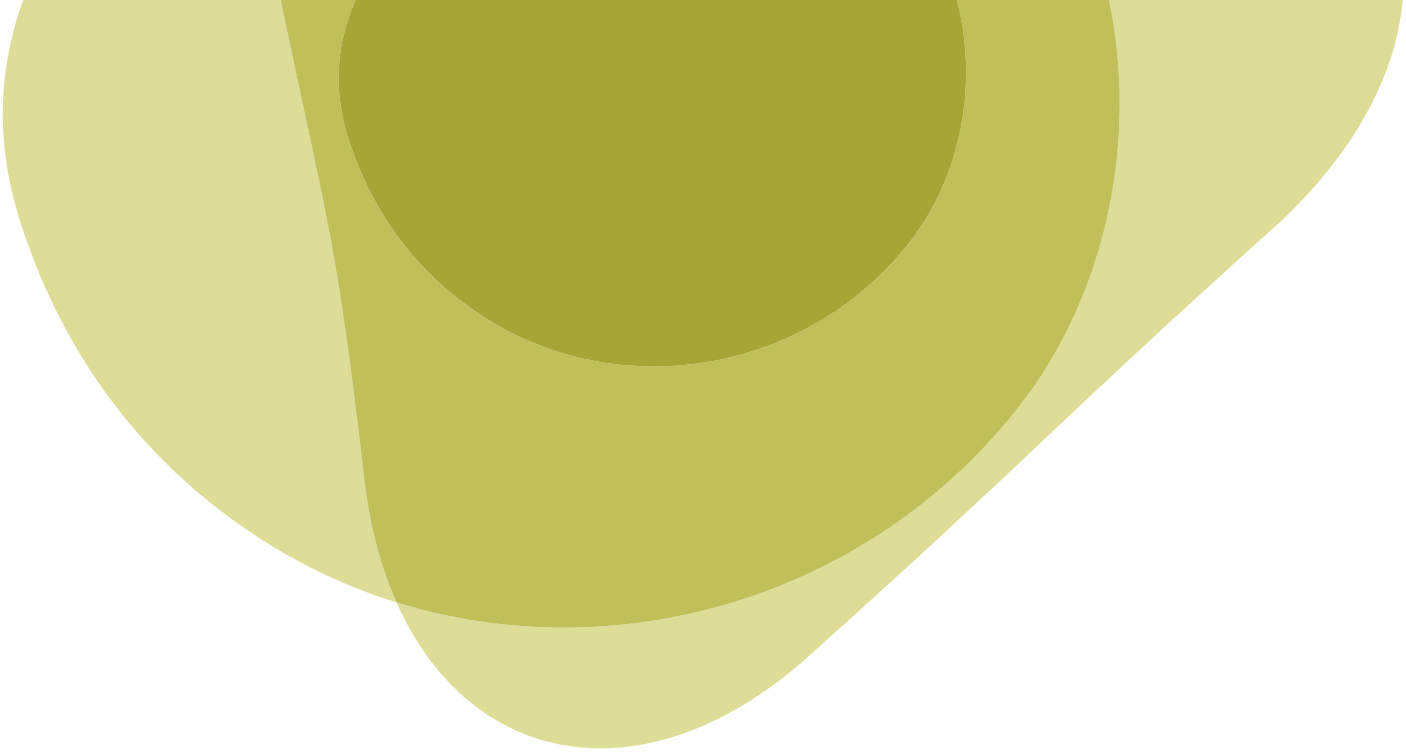
**POSTCSS** és un entorn de JavaScript basat en Node.Js per a operar en una representació sintàctica abstracta de documents CSS amb un gran ecosistema de connectors. Sovint considerat erròniament com un preprocessador (com ara un SaaS o inferior), creiem que el poder real de PostCSS ve de la quantitat de coses que es poden fer amb el gran nombre de connectors, els quals inclou linting (el connector stylelint), compilació creuada (el connector sugarss), canvis de nom per a evitar col·lisions de selectors (el connector de mòduls), generació de codi CSS (el connector autoprefixer), la minificació i molts d'altres. Tot i els diferents nivells de maduresa dels connectors, PostCSS segueix sent un entorn simple i poderós de tractament de CSS com un llenguatge complet per al desenvolupament.

Els equips que creen sistemes composats per microserveis han de pensar en tècniques de coordinació com el descobriment del servei, l'equilibri de la càrrega, el trencament del circuit i les proves de salut. Moltes d'aquestes tècniques requereixen que els equips estableixin les eines necessàries, la qual cosa no sempre és fàcil. El projecte **SPRING CLOUD** proporciona eines per als desenvolupadors per tal que puguin utilitzar aquestes tècniques de coordinació en el familiar entorn Spring. Aquestes eines suporten Consul, Zookeeper i el Netflix OSS full stack, les quals ens agraden totes. Dit senzillament, és fàcil fer el correcte amb aquestes eines. Tot i que pel que fa a Spring ens segueix preocupant el de sempre, bàsicament que amaga massa la complexitat, s'hauria de considerar Spring Cloud si es treballa en l'ecosistema i es necessita solucionar aquests problemes.

Sigues el primer a saber quan surt el Radar Tecnològic i segueix actualitzat amb seminaris en línia i contingut exclusius.

***SUBSCRIURE'S ARA***

***thght.works/Sub-CA***



**ThoughtWorks®**

ThoughtWorks és una consultoria i comunitat tecnològica formada per persones apassionades i mogudes per objectius. Ajudem els nostres clients a posar la tecnologia al centre del seu negoci i, junts, creem els programes que més els importen. Ens dediquem al canvi social positiu: la nostra missió és crear una humanitat millor a través de programes i ens associem amb moltes organitzacions que comparteixen els mateixos objectius.

Fundada fa més de 20 anys, ThoughtWorks ha crescut fins a convertir-se en una empresa formada per més de 4000 persones i amb una divisió de productes encarregada de crear eines pioneres pels equips de programació. ThoughtWorks té 40 oficines repartides en 14 països: Alemanya, Austràlia, Brasil, Equador, Espanya, Estats Units, Índia, Itàlia, Regne Unit, Singapur, Sudàfrica, Turquia, Xile i Xina.

*thoughtworks.com*