ThoughtWorks®

TECHNOLOGY RADAR *VOL.17*

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

QUÈ HI HA DE NOU?

Aquests són els temes principals d'aquesta edició:

PROGRAMARI DE CODI OBERT "MADE IN CHINA"

La marea està pujant. A causa dels canvis tant d'altitud com de polítiques, grans companyies xineses com Alibaba i Baidu estan traient al mercat entorns de treball, eines i plataformes de codi obert. A mesura que s'expandeixen econòmicament, els seus ecosistemes de programari creixen ràpidament.

S'espera que el nombre de projectes de codi obert de qualitat a Github i altres pàgines de codi obert creixi a causa del gran nombre de projectes del gegant mercat xinès. Però per què volen alliberar tants projectes les empreses xineses? Igual que passa a grans mercats de programari com Silicon Valley, existeix molta competència entre els desenvolupadors i apujar els sous no sempre funciona indefinidament.

És per això que el fet de treballar a un projecte de codi obert capdavanter amb altres grans desenvolupadors és un gran al·licient universal. Creiem que les grans innovacions de codi obert seguiran el mateix camí dels arxius README que ja s'escriuen primer en xinès i després en anglès.

KUBERNETES, L'ORQUESTRADOR DE CONTENIDORS D'ELECCIÓ

Hi ha un gran nombre d'entrades a aquest Radar que parlen de Kubernetes i la seva presència cada vegada més gran a molts projectes. Sembla que l'ecosistema de desenvolupament de programari s'està decantant per Kubernetes i les seves eines relacionades per solucionar els problemes relacionats amb el desplegament, l'escalació i l'operació de contenidors.

Algunes entrades del radar com ara GKE, Kops, i Sonobuoy parlen de serveis de plataformes gestionades i d'eines que milloren l'experiència d'adoptar i utilitzar Kubernetes. La seva capacitat d'executar diversos contenidors com una sola unitat de planificació fa possible la xarxa de serveis i els sidecars per seguretat de punt final.

Kubernetes s'ha convertit en el sistema operatiu per contenidors per defecte: molts proveïdors de serveis al núvol han aprofitat la seva arquitectura oberta i modular per adoptar i utilitzar Kubernetes, mentre que les eines treuen profit de les seves APIs obertes per accedir a abstraccions com càrregues de treball, clústers, configuracions i emmagatzematge.

Estem veient com hi ha cada vegada més productes que utilitzen Kubernetes com un ecosistema, la qual cosa el converteix en el següent nivell d'abstracció després dels microserveis i els contenidors. Això, doncs, no deixa de ser una prova més de que els desenvolupadores poden treure partit d'estils d'arquitectura moderns independentment de les complexitats inherents als sistemes distribuïts.

EL NÚVOL COM LA NOVA NORMA

L'altre tema de conversa generalitzat present mentre muntàvem aquesta entrega del Radar tenia un caire molt "ennuvolat". A mesura que els proveïdors de serveis al núvol milloren i equiparen les seves característiques, el nou model públic al núvol s'està convertint en el servei d'elecció per a moltes organitzacions.

A l'hora d'embarcar-se en nous projectes hi ha moltes empreses que en lloc de preguntar-se "per què al núvol?", es pregunten "per què *no* al núvol?". És cert que alguns tipus de programari encara requereixen sistemes in situ, però a mida que els preus baixen i hi ha més i millors capacitats, el desenvolupament al núvol és cada vegada més viable.

Tot i que es podria dir que tots els proveïdors de serveis al núvol ofereixen les mateixes funcions bàsiques, tots tenen funcions úniques per solucions específiques que els diferencien dels altres. Així doncs, estem veient empreses que utilitzen diversos proveïdors gràcies al Polycloud i que seleccionen les capacitats específiques que millor s'ajusten a les seves necessitats.

CONFIANÇA EN CADENES DE BLOCS DISTRIBUÏDES MÉS UNIFORMEMENT

Tot i el caos al voltant de les criptodivises als mercats, molts dels nostres clients estan trobant maneres de treure partit a solucions de cadenes de blocs per a registres comptables i contractes intel·ligents. Moltes de les entrades d'aquest Radar mostren maduresa en l'ús de tecnologies relacionades amb les cadenes de blocs, les quals aporten maneres cada vegada més interessants d'implementar contractes intel·ligents amb una varietat de tècniques i llenguatges de programació.

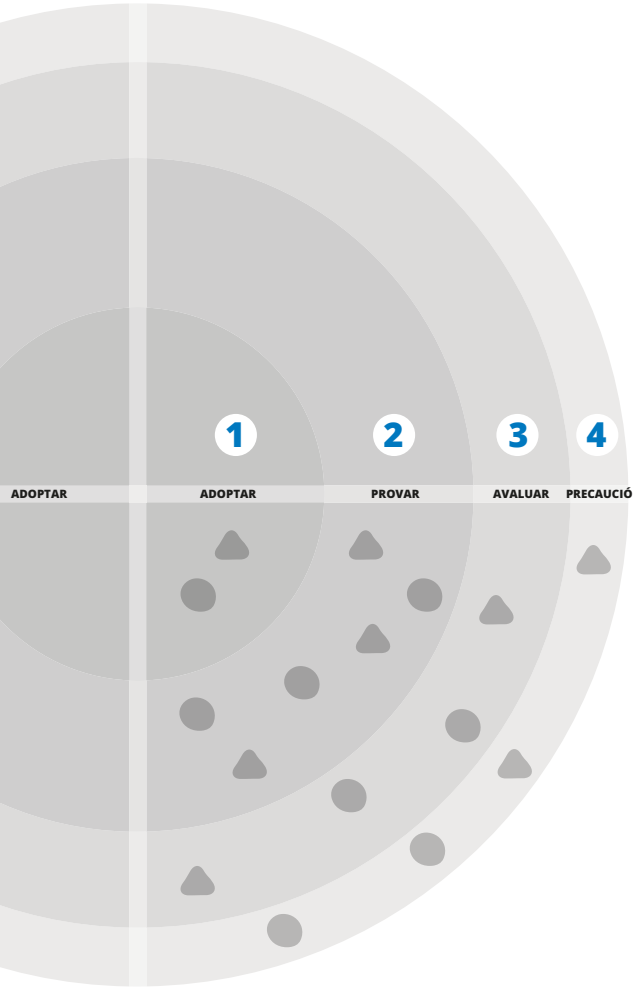
Les cadenes de blocs solucionen el vell problema de la confiança distribuïda i dels registres comptables compartits i inesborrables. Actualment, les empreses estan incrementant la confiança dels seus usuaris en les mecàniques subjacents a les implementacions de cadenes de blocs. Diferents indústries tenen diferents problemes de confiança, però creiem que les solucions de cadenes de blocs trobaran la manera de solucionar-los.

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 CANVIS

● 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. Enregistrament de decisions d'arquitectura lleugera

PROVAR
2. Aplicar la gestió de productes a les plataformes internes *NOU*
3. Funcions d'aptitud arquitectural *NOU*
4. Patró de bombolles autònomes *NOU*
5. Enginyeria del Caos *NOU*
6. Desacoblament de la gestió de secrets del codi font
7. DesignOps *NOU*
8. llegat en una caixa
9. Microinterfícies
10. Canals per infrastructura com a codi *NOU*
11. Arquitectures Serverless
12. Desenvolupament de contenidors basant-se en proves *NOU*

AVALUAR
13. Operacions informàtiques algorítmiques *NOU*
14. Ethereum per aplicacions descentralitzades *NOU*
15. Streaming d'esdeveniments com a font de veritat *NOU*
16. Equips de producció de plataformes
17. Multinúvol *NOU*
18. Xarxa de serveis *NOU*
19. Sidecars per seguretat de punt final *NOU*
20. Les tres Rs de la seguretat *NOU*

PRECAUCIÓ
21. Instància única d'integració contínua per a tots els equips
22. Teatre IC
23. Entorns de proves d'integració a tota l'empresa
24. Recrean antipatrons ESB amb Kafka *NOU*
25. Spec-based codegen

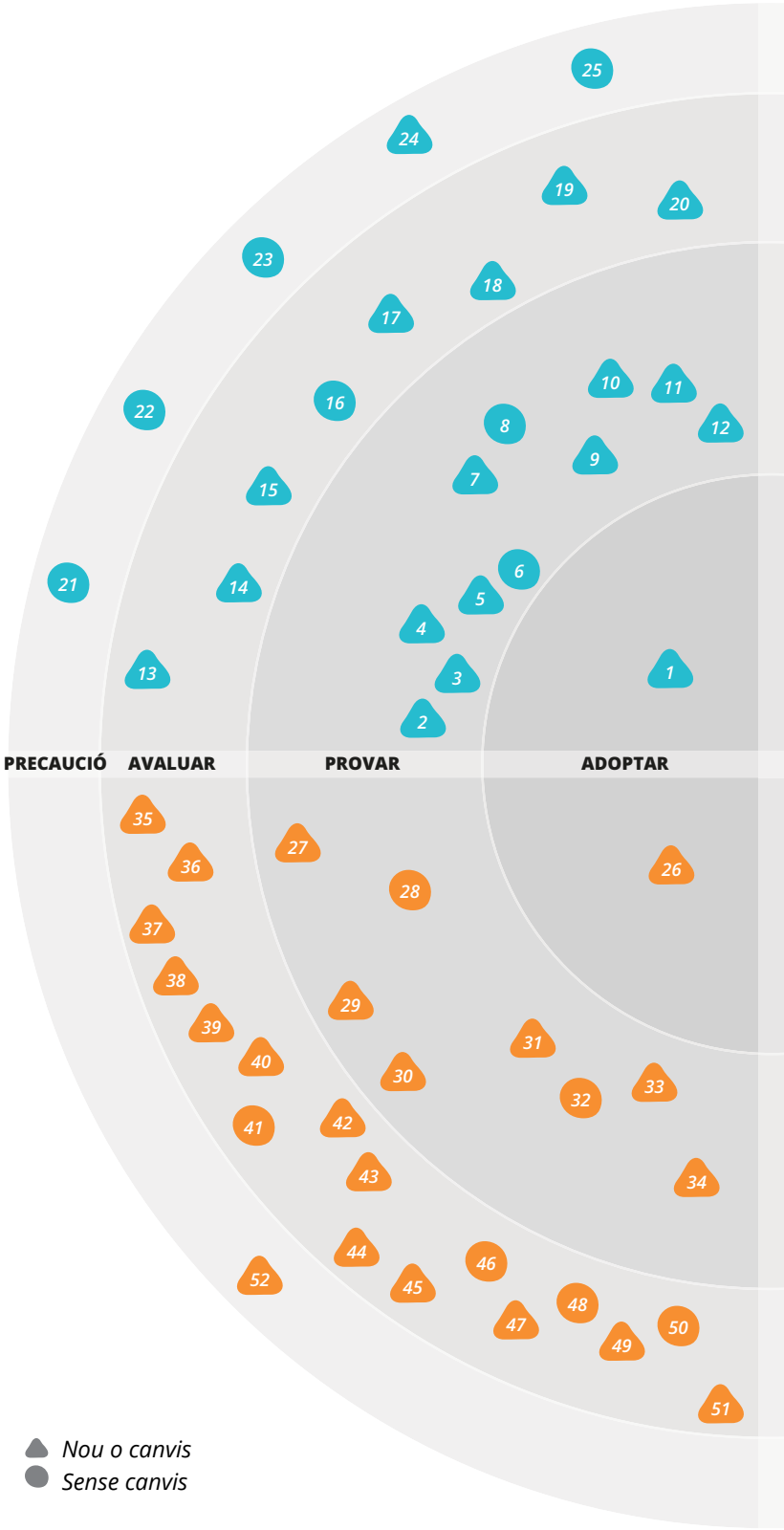
PLATAFORMES

ADOPTAR
26. Kubernetes

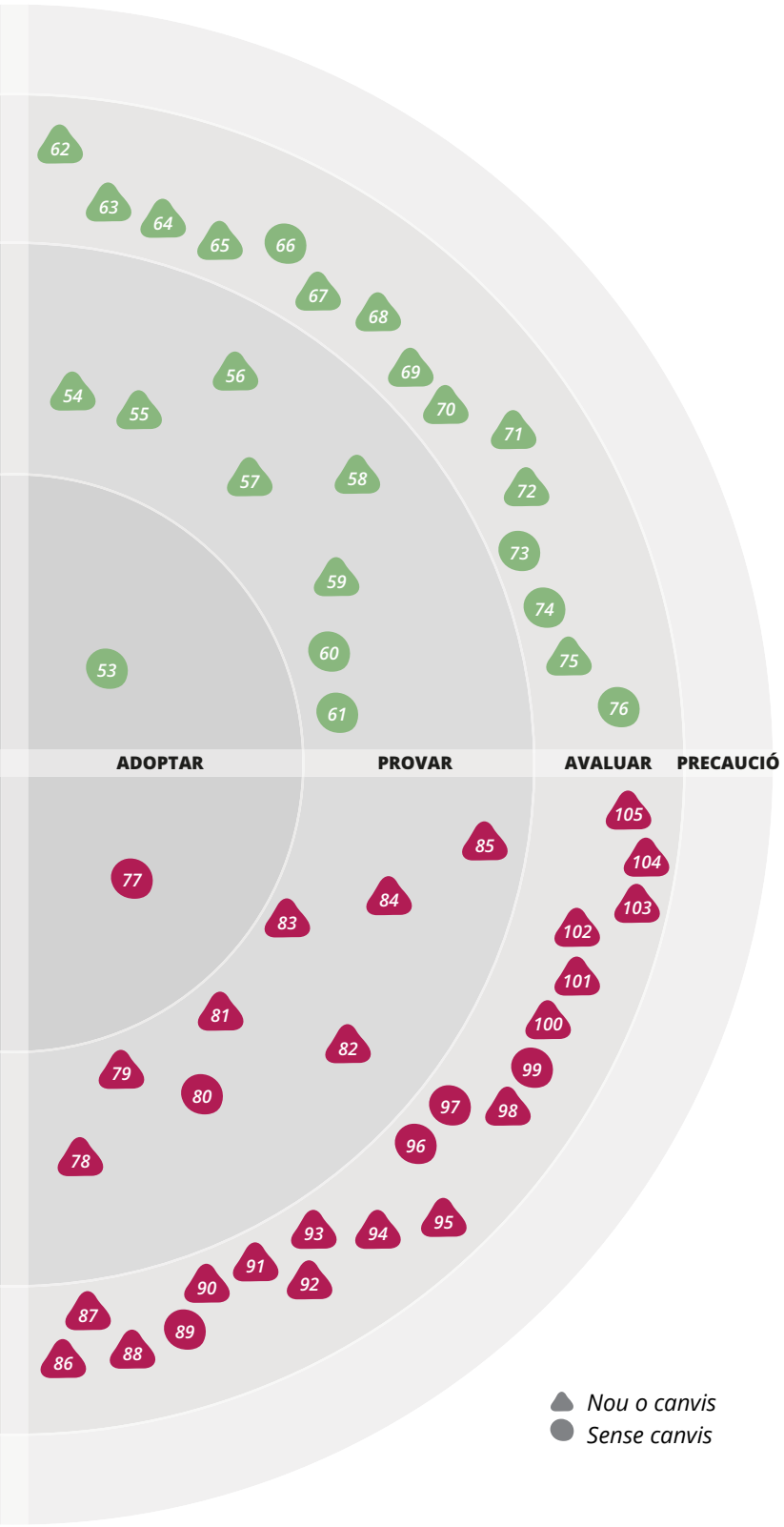
PROVAR
27. .NET Core
28. AWS Device Farm
29. Flood IO *NOU*
30. Google Cloud Platform *NOU*
31. Keycloak
32. OpenTracing
33. Unity més enllà dels videojocs
34. WeChat *NOU*

AVALUAR
35. Azure Service Fabric *NOU*
36. Cloud Spanner *NOU*
37. Corda *NOU*
38. Cosmos DB *NOU*
39. DialogFlow
40. GKE *NOU*
41. Hyperledger
42. Kafka Streams
43. Servidor de sintaxi del Llenguatge *NOU*
44. LoRaWAN *NOU*
45. MapD *NOU*
46. Mosquitto
47. Netlify *NOU*
48. PlatformIO
49. TensorFlow Serving *NOU*
50. Plataformes de veu
51. Windows Containers *NOU*

PRECAUCIÓ
52. API gateway massa ambiciose



EL RADAR



EINES

ADOPTAR
53. fastlane

PROVAR
54. Buildkite *NOU*
55. CircleCI *NOU*
56. gopass *NOU*
57. Headless Chrome per proves d'interfície *NOU*
58. jsoniter *NOU*
59. Prometheus
60. Scikit-learn
61. Entorn de treball sense servidor

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

PRECAUCIÓ

LLENGUATGES I ENTORNS DE PRECAUCIÓ TREBALL

ADOPTAR
77. Python 3

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

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

PRECAUCIÓ

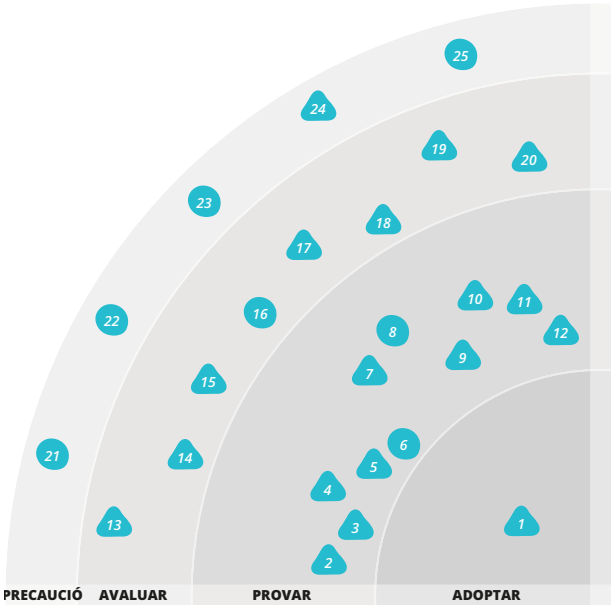
TÈCNIQUES

ADOPTAR
1. Enregistrament de decisions d'arquitectura lleugera

PROVAR
2. Aplicar la gestió de productes a les plataformes internes *NOU*
3. Funcions d'aptitud arquitectural *NOU*
4. Patró de bombolles autònomes *NOU*
5. Enginyeria del Caos *NOU*
6. Desacoblament de la gestió de secrets del codi font
7. DesignOps *NOU*
8. llegat en una caixa
9. Microinterfícies
10. Canals per infrastructura com a codi *NOU*
11. Arquitectures Serverless
12. Desenvolupament de contenidors basant-se en proves *NOU*

AVALUAR
13. Operacions informàtiques algorítmiques *NOU*
14. Ethereum per aplicacions descentralitzades *NOU*
15. Streaming d'esdeveniments com a font de veritat *NOU*
16. Equips de producció de plataformes
17. Multinúvol *NOU*
18. Xarxa de serveis *NOU*
19. Sidecars per seguretat de punt final *NOU*
20. Les tres Rs de la seguretat *NOU*

PRECAUCIÓ
21. Instància única d'integració contínua per a tots els equips
22. Teatre IC
23. Entorns de proves d'integració a tota l'empresa
24. Recrean antipatrons ESB amb Kafka *NOU*
25. Spec-based codegen



Molta de la documentació es pot substituir per codi llegible i proves. En un món d'arquitectura evolucionaria, però, és important enregistrar certes decisions de disseny per a benefici tant de futurs membres de l'equip com de revisions externes. L'**ENREGISTRAMENT DE DECISIONS D'ARQUITECTURA LLEUGERA** és una tècnica per capturar decisions de l'arquitectura importants conjuntament amb el seu context i les seves conseqüències. Nosaltres recomanem enregistrar aquests detalls al sistema control de versions, en lloc de a una wiki o pàgina web, ja que proporcionen un registre que estarà sincronitzat amb el codi. Creiem que per a la majoria de projectes no hi ha cap motiu pel qual no utilitzar aquesta tècnica.

Els últims 12 mesos hem vist un gran increment en l'interès en les plataformes digitals. Les companyies que volen crear noves solucions digitals de manera ràpida i eficient estan creant plataformes internes que proporcionen als equips accés directe a les APIs, eines, coneixements i suport de l'empresa necessaris per crear i operar les seves pròpies solucions. Nosaltres creiem que aquestes plataformes són més efectives quan es tracten com a productes. **APLICAR LA GESTIÓ DE PRODUCTES A LES PLATAFORMES INTERNES** implica establir una relació d'empatia amb els consumidors interns (els desenvolupadors) i col·laborar amb ells en el disseny. Els gestors de plataformes com a productes estableixen els fulls de ruta i s'asseguren que la plataforma aporti valor a l'empresa i millori l'experiència dels desenvolupadors. Hi ha qui, fins i tot, crea una marca per a la seva plataforma interna i l'utilitza per promocionar-la davant dels seus col·legues. Els gestors de plataformes com a productes s'asseguren de la qualitat de la plataforma, reuneixen mètriques d'ús i la milloren contínuament. Tractar la plataforma com un producte ajuda a crear un ecosistema i evita haver de tornar a crear una altra arquitectura orientada als serveis poc utilitzada y estancada.

Inspirades per la computació evolutiva, les funcions d'aptitud s'utilitzen per expressar les possibilitats d'una solució de disseny d'aconseguir els objectius establerts.

(Funcions d'aptitud arquitecturals)

Inspirades per la computació evolutiva, les funcions d'aptitud s'utilitzen per expressar les possibilitats d'una solució de disseny d'aconseguir els objectius establerts. A l'hora de definir un algoritme evolutiu, el dissenyador busca aconseguir un algoritme "millor" i són les funcions d'aptitud les que defineixen el significat de "millor" en aquest context. Una **FUNCIONS D'APTITUD ARQUITECTURAL**, tal com es defineix a Building Evolutionary Architectures, proporciona una avaluació de la integritat de l'objectiu d'algunes característiques arquitecturals que poden incloure criteris de verificació existents com ara proves unitàries, mètriques, monitoratge, etc. Nosaltres creiem que els arquitectes poden comunicar, validar i preservar característiques arquitecturals de manera automàtica i contínua, el que és la clau per a construir arquitectures evolutives.

Es poden utilitzar eines d'integració contínua (IC) i d'entrega contínua (EC) per provar les configuracions de servidors, la creació d'imatges de servidors, l'aprovisionament de l'entorn i la integració d'entorns.

(Canals per infraestructura com a codi)

Moltes organitzacions amb les que treballem estan intentant utilitzar enfocaments d'enginyeria moderns per a la creació de noves capacitats i característiques a la vegada que coexisteixen amb tota una sèrie de sistemes heretats. Una antiga estratègia que, basant-nos en la nostra experiència, està sent cada vegada més útil en casos com aquests és el **PATRÓ DE BOMBOLLES AUTÒNOMES** de Eric Evans. Aquest enfocament implica crear un nou context per al desenvolupament d'aplicacions protegint-lo de tot l'embolic del món heretat. Es tracta d'anar un pas més enllà de l'ús de capes anticorrupció. Dóna al nou context de bombolles control absolut sobre les seves dades de suport i, aleshores, es manté actualitzat de manera asíncrona amb els sistemes heretats. Es necessita una mica de feina per protegir les fronteres de la bombolla i fer que els dos mons siguin consistents, però l'autonomia i la reducció de friccions durant el desenvolupament que s'aconsegueix és un gran primer pas cap a una futura arquitectura modernitzada.

A edicions anteriors del Radar hem parlat de l'ús de l'eina de Netflix Chaos Monkey per veure la capacitat d'un sistema actiu de fer front a talls en la producció desactivant instàncies aleatòries i mesurant els resultats. L'**ENGINYERIA DEL CAOS** és el nom que se li està donant a l'aplicació més extensa d'aquesta tècnica. Duent a terme experiments en sistemes en producció, podem estar segurs que aquells sistemes funcionen correctament en condicions turbulentes. Un bon lloc on començar a entendre aquesta tècnica és la web Principles of Chaos Engineering.

DESIGNOPS, inspirat pel moviment DevOps, és un canvi cultural i una sèrie de pràctiques que permeten redissenyar productes sense posar en perill la qualitat, la coherència del servei o l'autonomia de l'equip. DesignOps promou la creació i l'evolució d'una infraestructura de disseny que minimitzi l'esforç necessari per crear nous conceptes i variacions d'interfícies d'usuari i l'establiment d'una comunicació ràpida i fiable amb l'usuari final. Amb eines com Storybook que promouen la col·laboració, es redueix al mínim la necessitat d'anàlisis previs i de transferències d'especificacions. Gràcies a DesignOps, el disseny està passant de ser una feina específica, a ser part de la feina de tothom.

Hem vist com introduir arquitectures de microserveis té grans beneficis ja que permet als equips d'escalar l'entrega de serveis desplegats i mantinguts independentment. També hem vist, però, com molta gent ha creat interfícies monolítiques a sobre dels seus serveis en forma d'aplicacions web úniques i pesades. El nostre enfocament preferit (i que ja hem provat) és el de dividir el codi del navegador en **MICROINTERFÍCIES**. Així, l'aplicació web es divideix segons les seves funcionalitats i cada funcionalitat està controlada pel seu propi equip. Això assegura que es desenvolupa, posa a prova i s'implementen de manera independent de la resta. Existeixen diverses tècniques per a recombinar les funcionalitats en una experiència d'usuari cohesiva, ja sigui com a components o com a pàgines.

L'ús de canals d'entrega contínua per orquestrar el procés de lliurament de programari s'està convertint en la norma. No obstant això, el fet de provar automàticament els canvis al codi d'infraestructura no està tan estès. Es poden utilitzar eines d'integració contínua (CI) i d'entrega contínua (CD) per provar les configuracions de servidors (com ara cookbooks de Chef, mòduls Puppet o playbooks de Ansible), la creació d'imatges de servidors (com ara Packer),

l'aprovisionament de l'entorn (com ara Terraform o CloudFormation) i la integració d'entorns. L'ús de **CANALS PER INFRASTRUCTURA COM A CODI** permet trobar errors abans d'aplicar els canvis a l'entorn operacional, incloent els entorns usats pel desenvolupament i les proves. També ofereixen una manera d'assegurar que les eines de la infraestructura s'executen consistentment des d'agents CI/CD, en lloc de que s'executin des d'estacions individuals. Encara queden, però, diversos reptes com ara el *feedback* més lent associat als contenidors i les màquines virtuals. Tot i així, creiem que és una tècnica molt interessant.

L'ús d'**ARQUITECTURES SERVERLESS** s'ha tornat ràpidament una opció vàlida per a organitzacions desplegant aplicacions al núvol, amb una gran quantitat d'opcions disponibles pel desplegament. Hi ha, fins i tot, organitzacions tradicionalment conservadores que estan utilitzant parcialment algunes tecnologies sense servidor. La major part de la discussió es centra en les Funcions com a Servei (per exemple, AWS Lambda, Cloud Functions de Google o Funcions d'Azure) mentre que els patrons apropiats d'ús encara estan apareixent. Desplegar funcions sense servidor elimina, indubtablement, l'esforç que suposa tradicionalment la configuració i la gestió del servidor i del sistema operatiu. Les funcions sense servidor, però, no sempre són la millor opció. De moment, s'ha d'estar preparat per a fer un pas enrere i desplegar contenidors i, fins i tot, instàncies del servidor per a requeriments específics. Actualment, però, altres components de l'arquitectura sense servidors, com ara el *Backend* com a servei, s'han pràcticament tornat la opció per defecte.

Molts equips de desenvolupadors han adoptat, pels seus beneficis, pràctiques de desenvolupament basat en proves per escriure codi d'aplicacions. Altres han optat per utilitzar contenidors per empaquetar i desplegar el seu programari i, com ja és normal, utilitzar *scripts* automàtics per crear els contenidors. El que hem vist molt pocs equips fer és combinar les dues tècniques i desenvolupar el contenidor basant-se en proves. Amb entorns de treball com Serverspec i Goss, es pot aconseguir la funcionalitat desitjada tant per contenidors aïllats com orquestrats, amb un bucle de realimentació curt. Això vol dir que es pot seguir utilitzant els mateixos principis que ja es coneixen per escriure codi **DESENVOLUPAMENT DE CONTENIDORS BASANT-SE EN PROVES**. La nostra experiència inicial ha sigut molt positiva.

La quantitat de dades recollides per operacions informàtiques no ha deixat de créixer durant anys. La moda pels microserveis, per exemple, implica que cada vegada hi ha més aplicacions que creen les seves pròpies dades operacionals i eines com Splunk, Prometheus o la pila ELK, faciliten l'enregistrament i el processament de dades per aconseguir informació operacional. Quan es combina amb eines d'aprenentatge automàtic cada vegada més democratitzades, és inevitable que els operadors comencin a incorporar models estadístics i algoritmes de classificació a les seves eines. Tot i que aquests algoritmes ja fa anys que estan disponibles i s'han fet diversos intents per automatitzar la gestió de serveis, tot just comencem a entendre com poden col·laborar màquines i humans per identificar més aviat mancances o la raó d'alguns errors. Tot i el risc que les **OPERACIONS INFORMÀTIQUES ALGORÍTMIQUES** no acabin sent tant importants com sembla, la millora constant dels algoritmes d'aprenentatge automàtic canviarà, sense dubte, el paper dels humans en la gestió dels centres de dades del futur.

Les cadenes de blocs han causat gran excitació i s'han considerat la panacea per tot el relacionat amb el fintech: des de la banca fins a la transparència en les cadenes de subministrament, passant per les monedes digitals.

(Ethereum per aplicacions descentralitzades)

Les cadenes de blocs han causat gran excitació i s'han considerat la panacea per tot el relacionat amb el *fintech*: des de la banca fins a la transparència en les cadenes de subministrament, passant per les monedes digitals. A Radars anteriors hem parlat d'Ethereum per les seves característiques i funcions, entre les quals es troben els contractes intel·ligents. Actualment veiem més desenvolupament utilitzant **ETHEREUM PER APLICACIONS DESCENTRALITZADES** a d'altres àrees. Tot i que segueix sent una tecnologia molt jove, creiem que s'utilitzarà per desenvolupar aplicacions descentralitzades més enllà de les criptodivises i la banca.

A mesura que les plataformes de *streaming* d'esdeveniments, com ara Apache Kafka, són cada cop més populars, molta gent les veu com cues de missatges avançades que s'utilitzen, únicament, per transmetre esdeveniments. Fins i tot quan s'utilitza d'aquesta manera, el *streaming* d'esdeveniments té els seus beneficis sobre les cues de missatges tradicionals. A nosaltres, però, ens interessa més saber com la gent utilitza el **STREAMING D'ESDEVENIMENTS COM A FONT DE VERITAT** amb plataformes (particularment Kafka) com a lloc d'emmagatzematge primari per dades com a esdeveniments immutables. Un servei amb un disseny de Event Sourcing, per exemple, pot utilitzar Kafka i el seu emmagatzematge d'esdeveniments i fer que, així, altres serveis puguin utilitzar aquells esdeveniments. Aquesta tècnica té el potencial de reduir la duplicació d'esforços entre la persistència local i la integració.

Els principals proveïdors de serveis al núvol (Amazon, Microsoft i Google) estan immersos en una ferotge carrera per la paritat en les capacitats bàsiques mentre que els seus productes són només marginalment diferents. Això està causant que algunes organitzacions adoptin una estratègia **MULTINÚVOL**: en lloc d'utilitzar només un proveïdor, utilitzen diferents proveïdors per a diferents tipus de treballs segons el que vagi millor per cada feina. Així, es podria utilitzar AWS per serveis estàndards, però Google per l'aprenentatge automàtic, Azure per aplicacions .NET que utilitzen SQLServer o utilitzar la solució del consorci de cadenes de blocs Ethereum. Aquesta estratègia és diferent de les estratègies antinúvol de buscar la portabilitat entre proveïdors ja que són cares i forcen el pensament de mínim comú denominador. El pensament multinúvol, en canvi, es centra en utilitzar el millor de cada servei al núvol.

Una xarxa de serveis ofereix una plataforma consistent i segura per descobrir, traçar, monitorar i ocupar-se dels errors sense la necessitat d'un recurs compartit com una API gateway o ESB.

(Xarxa de serveis)

A mesura que grans organitzacions tenen equips més autònoms amb els seus propis microserveis, com poden assegurar la consistència i la compatibilitat entre serveis sense dependre d'una estructura de servidor centralitzat? Per què funcionin conjuntament de manera eficient, fins i tot els microserveis autònoms necessiten seguir alguns estàndards organitzatius. Una **XARXA DE SERVEIS** ofereix una plataforma consistent i segura per descobrir, traçar, monitorar i ocupar-se dels errors sense la necessitat d'un recurs compartit com una API *gateway* o ESB. Una implementació típica implica processos lleugers de *proxy* invers desplegats amb cada servei, per exemple, en un contenidor separat. Aquests servidors es comuniquen amb registres de serveis, proveïdors d'identitat, agregadors de registres, etc. La interoperabilitat i la visibilitat dels serveis es dóna gràcies a una implementació compartida d'aquest *proxy* i no a una instància compartida. Hem sigut defensors de l'enfocament descentralitzat dels microserveis des de ja fa un temps i ens agrada veure aparèixer aquest patró. Projectes de codi obert com linkerd i Istio seguiran madurant i faran que les xarxes de serveis siguin encara més fàcils d'implementar.

A causa del gran nombre de serveis que exposen les seves característiques i capacitats a través de APIs i de la seva superfície d'atac més gran, les arquitectures de microserveis han de tenir una arquitectura de seguretat de confiança zero: no confiïs, verifica. No obstant això, sovint s'obvien els controls de seguretat en les comunicacions entre serveis per culpa de la major complexitat del codi del servei i de la falta de llibreries i de suport al llenguatge en entorns poliglots. Per reduir aquesta complexitat, alguns equips deleguen la seguretat a sidecars que es troben fora del procés, és a dir, un procés o contenidor que es desplega i programa amb tots els serveis que comparteixen el mateix context d'execució, el mateix servidor i la mateixa identitat. Els sidecars implementen capacitats de seguretat com ara l'encriptació transparent de la comunicació i la seguretat de la capa de transport (TLS per les sigles en anglès), així com l'autenticació i l'autorització del servei o de l'usuari final. Recomanem donar un cop d'ull a Istio, linkerd o Envoy abans d'implementar **SIDECARS PER SEGURETAT DE PUNT FINAL**.

Els enfocaments tradicionals de la seguretat empresarial sovint busquen tancar-ho tot i alentir els canvis. Sabem, però, que com més trigui un atacant a posar en perill un sistema, més gran és el mal potencial. Les tres Rs de la seguretat empresarial, rotar, reparar i repavimentar, aprofiten l'automatització de la infraestructura i l'entrega contínua per eliminar les oportunitats d'atac. Rotar les credencials, aplicar

pedaços tan aviat com estan disponibles i tornar a construir sistemes des d'estats anteriors segurs, tot això en qüestió de minuts o hores, fa que sigui molt difícil que els atacants tinguin èxit. La tècnica de **LES TRES RS DE LA SEGURETAT** és possible gràcies als avenços en les arquitectures al núvol modernes. Quan es creen i proven aplicacions a través de canals totalment automatitzats i es despleguen com a contenidors, un pedaç de seguretat no és més que una altra petita publicació que es pot enviar fàcilment amb només un clic. Per seguir les millors pràctiques de sistemes distribuïts, però, els desenvolupadors han de dissenyar les seves aplicacions de tal manera que puguin resistir caigudes dels servidors. Això és similar a l'impacte d'implementar Chaos Monkey a un entorn.

Kafka s'està tornant molt popular com a solució de missatgeria i, a la vegada, Kafka Streams és al capdavant de l'interès per les arquitectures de *streaming*. Estem veient organitzacions, però, que a la vegada que comencen a posar Kafka al centre de les seves plataformes de dades i d'aplicacions, **RECREAN ANTIPATRONS ESB AMB KAFKA** centralitzant els components de l'ecosistema Kafka, com ara connectors i processadors de transmissió, en lloc de permetre que els equips de productes o serveis s'encarreguin d'aquests components. Això ens recorda antipatrons ESB realment problemàtics, en els quals es ficava cada vegada més lògica, orquestració i transformació a ESB gestionades centralment creant, així, una gran dependència a l'equip centralitzat. Volem parlar d'aquest tema per evitar més implementacions d'aquest patró defectuós.

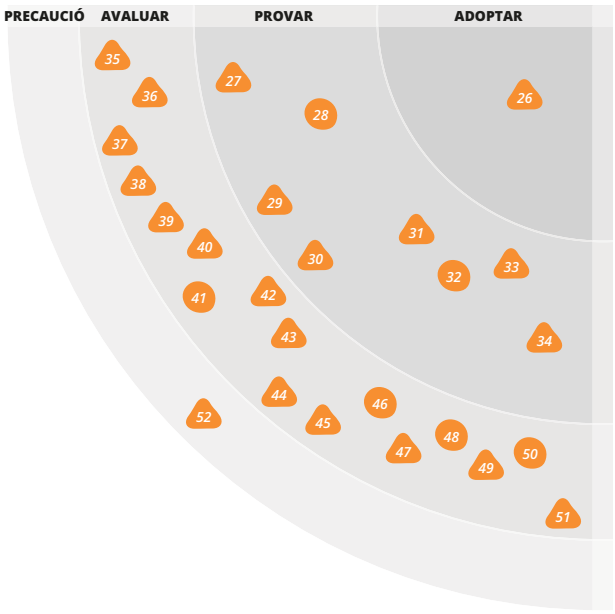
PLATAFORMES

ADOPTAR
26. Kubernetes

PROVAR
27. .NET Core
28. AWS Device Farm
29. Flood IO *NOU*
30. Google Cloud Platform *NOU*
31. Keycloak
32. OpenTracing
33. Unity més enllà dels videojocs
34. WeChat *NOU*

AVALUAR
35. Azure Service Fabric *NOU*
36. Cloud Spanner *NOU*
37. Corda *NOU*
38. Cosmos DB *NOU*
39. DialogFlow
40. GKE *NOU*
41. Hyperledger
42. Kafka Streams
43. Servidor de sintaxi del Llenguatge *NOU*
44. LoRaWAN *NOU*
45. MapD *NOU*
46. Mosquitto
47. Netlify *NOU*
48. PlatformIO
49. TensorFlow Serving *NOU*
50. Plataformes de veu
51. Windows Containers *NOU*

PRECAUCIÓ
52. API gateway massa ambicioses



Des de l'última vegada que vam parlar de **KUBERNETES** al Radar, s'ha convertit en la solució estàndard per a la majoria dels nostres clients a l'hora de desplegar contenidors a un clúster de màquines. Les alternatives existents no comptaven amb la mateixa popularitat i hem vist com, en alguns casos, alguns clients estan canviant el seu "motor" per Kubernetes. Kubernetes s'ha tornat la plataforma de gestió de contenidors per defecte per a grans plataformes públiques de sistemes al núvol entre les quals es troba el Microsoft Azure Container Service i Google Cloud (veure GKE). A més, hi ha molts altres productes que estan enriquint l'ecosistema de Kubernetes. Les plataformes que intenten amagar Kubernetes sota una capa d'abstracció, però, encara han de demostrar la seva vàlua.

Estem veient un increment en l'adopció de .NET Core, un entorn de treball de codi obert i multi-plataforma per a programari. .NET Core possibilita el desenvolupament i desplegament d'aplicacions .NET a Windows, macOS i Linux. Amb el llançament de NET Standard 2.0, que incrementa el nombre d'APIs estàndards entre les plataformes .NET, el camí per migrar cap a .NET Core queda més clar. Els problemes relacionats amb el suport de llibreries a .NET Core estan desapareixen poc a poc i ja hi ha eines multiplataforma de primera classe disponibles, la qual cosa permet el desenvolupament eficient a plataformes que no són Windows. Es proporcionen imatges segures de Docker per facilitar la integració de serveis .NET Core a entorns de contenidors. L'actitud positiva de la comunitat i el feedback dels nostres projectes indiquen que .NET Core està llest pel seu ús generalitzat.

Les proves de càrrega s'han tornat més fàcils gràcies a la maduresa d'eines com Gatling i Locust i, a la vegada, infraestructures elàstiques de serveis al núvol possibiliten simular un gran nombre d'instàncies de clients. Ens agrada veure com Flood i altres plataformes de serveis al núvol van més enllà utilitzant aquestes

tecnologies. Flood IO és un servei de proves de càrrega SaaS que ajuda a distribuir i executar proves a centenars de servidors al núvol. El nostre equip ha descobert que és molt simple migrar les proves de rendiment a Flood reutilitzant els *scripts* ja existents de Gatling.

A mesura que **GOOGLE CLOUD PLATFORM** (GCP) s'expandeix en termes geogràfics i madura en termes de serveis, els clients de tot el món el poden tenir en compte per a les seves estratègies al núvol. Hi ha certes àrees on GCP ha aconseguit igualar-se amb el seu principal competidor, Amazon Web Services, pel que fa a característiques i hi ha d'altres àrees on, fins i tot, ha aconseguit diferenciar-se gràcies, especialment, a plataformes d'aprenentatge automàtic, eines d'enginyeria de dades i una solució pràctica de Kubernetes com a servei (GKE). A la pràctica, els nostres equips només tenen bones paraules sobre l'experiència de desenvolupar amb les eines i APIs de GCP.

A mesura que Google Cloud Platform (GCP) s'expandeix en termes geogràfics i madura en termes de serveis

(Google Cloud Platform)

A una arquitectura de microserveis, o a qualsevol altre tipus d'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 codi obert 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 de dades de xarxes socials i protocols estàndard com ara OpenID Connect, OAuth 2.0 i SAML. Els nostres equips utilitzen aquesta eina i tenen intenció de seguir-ho fent en el futur. Requereix, però, una mica de feina per configurar-la i, com que s'ha de configurar tant quan s'inicialitza com quan s'utilitza, és necessari escriure *scripts* per assegurar que els desplegaments es poden repetir.

A Radars anteriors, hem mencionat que Unity s'ha convertit en la plataforma d'elecció per al desenvolupament d'aplicacions de RV i RA perquè proporciona les abstraccions i les eines d'una plataforma més madura, però sent més accessible que la seva principal alternativa: Unreal Engine. Amb la recent introducció de ARKit per iOS i ARCore per Android, les dues principals plataformes mòbils ara compten amb SDKs potents i natius per crear aplicacions de realitat augmentada. Creiem, però, que molts equips, especialment aquells que tenen poca experiència creant jocs, es beneficiaran d'utilitzar abstraccions com Unity, raó per la qual parlem de **UNITY MÉS ENLLÀ DELS VIDEOJOCS**. Això permet que els desenvolupadors que no estan familiaritzats amb la tecnologia es centrin en un SDK i ofereix una solució per a la gran quantitat de dispositius, especialment Android, que no estan suportats pels SDKs natius.

WECHAT sovint vist com un equivalent de WhatsApp, s'està convertint en la plataforma de facto per empreses a la Xina. És possible que molta gent no ho sàpiga, però WeChat és, a més, una de les plataformes de pagament online més populars. Amb el seu sistema de gestió de continguts (CMS) i la seva gestió d'identitats, els petits comerços estan operant exclusivament amb WeChat. Gràcies a la funcionalitat de comptes de servei, grans organitzacions poden connectar el seu sistema intern amb els seus empleats. Com que més del 70% dels xinesos utilitzen WeChat, és una plataforma a tenir en compte per les empreses que volen expandir-se al mercat xinès.

AZURE SERVICE FABRIC és una plataforma de sistemes distribuïts creada per a microserveis i contenidors. Es pot comparar amb eines d'orquestració de contenidors com Kubernetes, però també funciona amb serveis antics normals. Es pot utilitzar d'una infinitat de maneres, des de serveis simples en el llenguatge que es vulgui fins a contenidors Docker o serveis fets amb un SDK. Des que es va llançar fa un parell d'anys, ha anat incorporant més característiques, incloent el suport per contenidors a Linux. Kubernetes ha sigut l'estrella entre les eines d'orquestració de contenidors, però Service Fabric és la opció estàndard per aplicacions .NET. A ThoughtWorks, l'estem utilitzant en alguns projectes i ens agrada el que veiem de moment.

CLOUD SPANNER és un servei de base de dades relacionals totalment gestionades que ofereix una alta disponibilitat i gran consistència sense comprometre la latència. Ja fa temps que Google treballa en una base de dades distribuïda globalment anomenada Spanner i fa poc que l'ha presentat al món sota el nom de Cloud Spanner. Permet escalar instàncies d'un node

a milers de nodes per tot el món sense preocupar-se per la consistència de les dades. Utilitzant TrueTime, un rellotge altament disponible i distribuït, Cloud Spanner proporciona una alta consistència per lectures i instantànies. Es pot utilitzar SQL per llegir dades de Cloud Spanner, però per operacions d'escriptura cal utilitzar la seva API RPC. Tot i que no tots els serveis requereixen bases de dades distribuïbles a escala global, la disponibilitat general de Cloud Spanner és un gran canvi en la manera com pensem en bases de dades i el seu disseny està influenciant productes de codi obert com CockroachDB.

R3, un important actor en el món de les cadenes de blocs, va decidir, després de molt investigar, que les cadenes de blocs no responien al seu objectiu i van crear **CORDA**. Corda és una plataforma de tecnologia de registre comptable distribuït (DLT en anglès) que es centra en el camp de les finances. R3 té una proposta de valor molt clara i sap que el seu problema requereix una tecnologia pragmàtica. Això coincideix amb la nostra experiència: les solucions actuals de cadenes de blocs poden no ser la opció més raonable per alguns negocis a causa del cost de la mineria i la ineficiència operacional. Tot i que, de moment, la nostra experiència desenvolupant amb Corda no ha sigut la millor, les APIs encara són inestables després del llançament de la versió 1.0, esperem veure com l'espai DLT segueix madurant.

COSMOS DB és el servei de base de dades multimodel i distribuït globalment de Microsoft, que s'ha fet disponible a gran escala aquest any. Mentre la majoria de bases de dades NoSQL ofereixen consistències ajustables, Cosmos DB ho porta un pas més enllà i ofereix 5 models diferents de consistència. Val la pena remarcar que també suporta models múltiples (valor clau, document, família de la columna i gràfic), tots els quals s'associen amb el seu model intern de dades, anomenat *atom-record-sequence* (ARS). Un aspecte interessant de Cosmos DB és que ofereix acords de nivell de servei (SLAs) a la seva latència, al seu rendiment, a la seva consistència i a la seva disponibilitat. El seu gran rang d'aplicacions possibles ha marcat un estàndard molt alt pels altres proveïdors de serveis al núvol.

Paral·lelament al recent augment en l'interès pels bots conversacionals i les plataformes de veu, hem vist una proliferació d'eines i plataformes que proporcionen serveis per extreure la intencionalitat del text i la gestió de fluxos conversacionals als quals és possible connectar-se. **DIALOGFLOW** (abans conegut com API.ai), adquirit per Google, és un servei de "comprensió-de-llenguatge-natural com a servei" que competeix amb wit.ai i Amazon Lex, entre d'altres actors d'aquest camp.

Tot i que l'ecosistema de desenvolupament de programari està gravitant cap a Kubernetes com a plataforma d'orquestració de contenidors, fer anar clústers de Kubernetes segueix sent una tasca operacionalment complexa. **GKE** (*Google Container Engine*) és una solució Kubernetes gestionada per desplegar aplicacions contenidoritzades que alleugen el cost operacional de fer anar i mantenir clústers Kubernetes. Els nostres equips han tingut una bona experiència utilitzant GKE fent que la plataforma sigui la que aplica els pedaços de seguretat, monitoritzi i repari els nodes i gestioni xarxes multiclúster i multiregió. Segons la nostra experiència, l'enfocament de Google de "les APIs primer" a l'hora d'exposar les capacitats d'una plataforma, així com d'utilitzar estàndards de la industria com OAuth per l'autorització de serveis, millora l'experiència del desenvolupador. És important tenir en compte que, tot i els esforços dels desenvolupadors d'aïllar els consumidors dels canvis interns, GKE no para de créixer i això ja ens ha afectat temporalment. Creiem que la Infraestructura com a codi seguirà madurant amb Terraform a GKE i eines similars.

Els servidors de llenguatge introdueixen la capacitat de fer refactoring qualsevol editor de textos pugui accedir a una API per a què treballi amb l'AST.

(Servidor de sintaxi del llenguatge)

KAFKA STREAMS és una llibreria de baix pes per crear aplicacions de *streaming*. S'ha dissenyat amb l'objectiu de simplificar prou el processament de serveis de *streaming* com per fer-ho fàcilment accessible com a model establert de programació d'aplicacions per a serveis asíncrons. Pot ser una bona alternativa en els casos en què es vol aplicar un model de processament de *streaming* a un problema sense haver de recórrer a la complexitat d'executar un clúster (normalment introduït per entorns de processament de *streaming* totalment desenvolupats). Nous desenvolupaments inclouen una cosa similar a l'entrega "exactly-once" per emmagatzemar l'estat d'una transmissió en un clúster

Kafka introduint idempotència a consumidors Kafka. Ens agrada la manera pragmàtica com han decidit mitigar els missatges duplicats a la seva infraestructura. Gran part de la potència d'entorns integrats de desenvolupament (IDEs) ve de la seva habilitat per convertir un programa en un arbre de sintaxi abstracta (AST per les sigles en anglès) i, tot seguit, utilitzar aquest AST per analitzar i manipular el programa. Això suporta característiques com autocompletar, trobada de crides i refatorització. Els servidors de llenguatge introdueixen aquesta capacitat a un procés que permet que qualsevol editor de textos pugui accedir a una API per què treballi amb l'AST. Microsoft ha liderat la creació del **SERVIDOR DE SINTAXI DEL LLENGUATGE** (LSP per les sigles en anglès), creat a partir dels seus projectes de servidor OmniSharp i TypeScript. Qualsevol editor que utilitzi aquest protocol pot treballar amb qualsevol llenguatge que tingui un servidor compatible amb LSP. Això vol dir, doncs, que es pot seguir utilitzant el mateix editor de sempre sense privar-se dels rics modes d'edició de molts llenguatges, la qual cosa és una gran noticia pels amants d'Emacs.

LORAWAN és una xarxa de baixa potència i gran abast dissenyada per tenir un baix consum d'energia i un gran abast utilitzant taxes de bits baixes. Possibilita la comunicació entre dispositius i portes d'enllaç per, tot seguit, reenviar les dades a, per exemple, aplicacions o servidors. Es seu ús més típic és amb una sèrie de sensors o amb dispositius del internet de les coses (IoT per les sigles en anglès) pels quals és molt important comptar amb una gran autonomia i un gran abast. LoRaWAN soluciona dos dels problemes clau a l'hora d'utilitzar Wi-Fi per aquest tipus d'aplicacions: abast i consum d'energia. Hi ha diverses implementacions d'entre les quals destaca The Things Network, una implementació gratuïta i de codi obert.

A mida que es passa d'usos experimentals a la producció, es necessita una manera fiable d'allotjar i desplegar els models que es poden accedir remotament i escalar segons el nombre de consumidors.

(TensorFlow Serving)

MAPD és una base de dades columnar en memòria amb suport per SQL creada per utilitzar la GPU. Hem discutit sobre si la càrrega de la base de dades és d'E/S o computacional, però hi ha instàncies on el paral·lelisme de la GPU, combinat amb la gran amplada de banda de la memòria VRAM, pot ser molt útil. MapD gestiona a la memòria VRAM, de manera transparent, les dades que més s'utilitzen (com ara columnes involucrades en la funció "group by", filtres, càlculs i condicions) i emmagatzema la resta a la memòria principal. Gràcies a aquesta configuració, MapD aconsegueix una gran capacitat de consulta sense necessitar índexs. Tot i que hi ha altres opcions de bases de dades de GPU al mercat, MapD és al capdavant d'aquest sector perquè va obrir el codi de la seva base de dades essencial i perquè fa part de la iniciativa GPU Open Analytics Initiative. Si es té una alta càrrega computacional, es pot explotar el paral·lelisme que ens proporciona la GPU, nosaltres recomanem provar MapD.

Ens agrada **NETLIFY**, permet crear contingut estàtic d'una pàgina web, penjar-ho a GitHub i tenir la web activa i disponible de manera fàcil i ràpida. Compta amb una línia d'ordres per controlar el procés disponible; suporta les xarxes de lliurament de contingut (CDNs per les sigles en anglès); pot treballar conjuntament amb eines com Grunt; i, el més important, Netlify suporta HTTPS.

Els models d'aprenentatge automàtic estan començant a arribar a les aplicacions del dia a dia. Quan disposen de suficients dades, aquests algoritmes poden solucionar problemes pels quals, no fa gaire, s'hagués necessitat models estadístics complexos o la tècnica de la prova i error. A mida que es passa d'usos experimentals a la producció, es necessita una manera fiable d'allotjar i desplegar els models que es poden accedir remotament i escalar segons el nombre de consumidors. **TENSORFLOW SERVING** soluciona part del problema exposant una interfície gRPC remota a un model exportat, la qual cosa permet desplegar un model entrenat de diferents maneres. TensorFlow Serving també accepta una onada de models per incorporar actualitzacions d'aprenentatge continu i els seus desenvolupadors mantenen un arxiu Dockerfile per facilitar el procés de desplegament. Suposadament, s'ha utilitzat gPRC per ser coherent amb el model d'execució TensorFlow, però generalment no ens refiem massa de protocols que demanen generar codi i enllaços natius.

Microsoft s'està posant al dia en el món dels contenidors amb **WINDOWS CONTAINERS**. En el moment d'escriure aquestes línies, Microsoft proporciona dues imatges del sistema operatiu

Windows en contenidors Docker: Windows Server 2016 Server Core i Windows Server 2016 Nano Server. Tot i que Windows Containers encara pot millorar, per exemple reduint la mida de les imatges i enriquint el suport i la documentació de l'ecosistema, els nostres equips l'han començat a utilitzar en escenaris on d'altres contenidors han funcionat correctament anteriorment, com ara agents de compilació.

Ens segueix preocupant la lògica empresarial i la orquestració de processos que s'està implementant al programari intermediari, especialment quan requereix capacitats i eines d'alt nivell per crear punts únics d'escalació i control. Els proveïdors del mercat altament competitiu de les API gateway segueixen aquesta moda incorporant característiques gràcies a les quals esperen diferenciar els seus productes. El resultat són **API GATEWAY MASSA AMBICIOSES** la funcionalitat de les quals, a sobre del que és, en essència, un *proxy* invers, fomenta dissenys que segueixen sent difícils de provar i desplegar. Les API gateways sí que són útils a l'hora de tractar alguns problemes en concret, com ara l'autenticació i el control de trànsit, però les dades de qualsevol domini haurien de ser a les aplicacions o serveis.

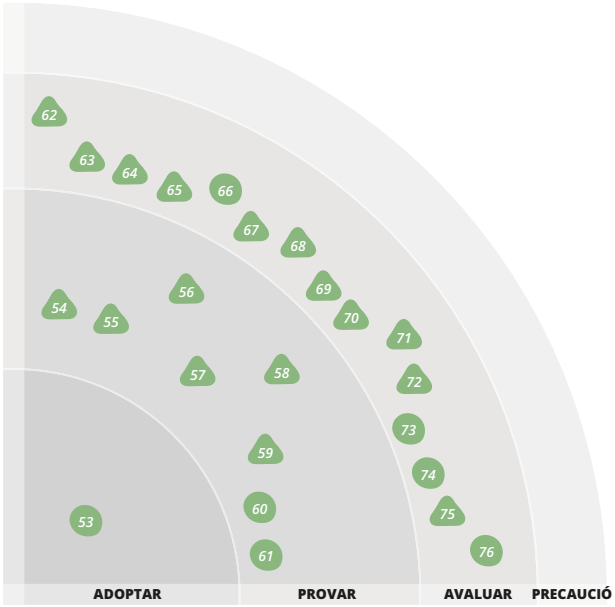
EINES

ADOPTAR
53. fastlane

PROVAR
54. Buildkite *NOU*
55. CircleCI *NOU*
56. gopass *NOU*
57. Headless Chrome per proves d'interfície *NOU*
58. jsoniter *NOU*
59. Prometheus
60. Scikit-learn
61. Entorn de treball sense servidor

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

PRECAUCIÓ



Als nostres equips els agrada molt l'eina d'integració i entrega contínua **BUILDKITE** per la seva simplicitat i facilitat de configuració. Buildkite permet utilitzar les màquines existents per executar compilacions, ja sigui localment o al núvol, i instal·lar una aplicació agent de baix pes per connectar l'agent de compilació al servei allotjat. Tenir aquest nivell de control sobre la configuració de l'agent de compilació és, en molts casos, un avantatge sobre l'ús d'agents allotjats.

CIRCLECI és un motor d'integració contínua venut com a SaaS in situ. Per a molts dels nostres equips de desenvolupament, CircleCI ha sigut l'eina d'elecció pel que fa a SaaS d'IC, ja que necessitaven una *pipeline* de desplegament i compilació fàcil de configurar i de baixa fricció. La versió 2.0 de CircleCI suporta fluxos de compilació amb fluxos de connectivitat d'entrada i de sortida i portes manuals, i desenvolupament per a mòbils. Permet que els desenvolupadors executin els canals localment i s'integra sense problemes amb Slack i d'altres sistemes de notificacions i alertes. Recomanem que es revisin les pràctiques de seguretat de CircleCI igual que es faria amb qualsevol altre producte SaaS que allotgi els actius d'una empresa.

És un descendent de pass i incorpora característiques com: suport per la gestió de receptors i emmagatzematge de múltiples contrasenyes en un sol arbre; funcionalitat de cerca interactiva; suport per contrasenyes temporals d'un sol ús (TOTP per les sigles en anglès); i emmagatzematge de dades binàries.

(gopass)

GOPASS és una solució de gestió de contrasenyes per equips, construïda sobre GPG i Git. És un descendent de pass i incorpora característiques com: suport per la gestió de receptors i emmagatzematge de múltiples contrasenyes en un sol arbre; funcionalitat de cerca interactiva; suport per contrasenyes temporals d'un sol ús (TOTP per les sigles en anglès); i emmagatzematge de dades binàries. Migrar les contrasenyes des de pass és una tasca bastant senzilla perquè gopass és altament compatible amb el format utilitzat per pass. Això, a més, implica que es pot aconseguir la integració en els fluxos d'aprovisionament gràcies a una única crida a un secret guardat.

Els usuaris de Chrome han tingut, des de mitjans de 2017, la possibilitat d'utilitzar-lo en mode "headless". Aquesta opció és ideal per fer proves de navegador sense la necessitat de mostrar res en una pantalla. Això era, antigament, el territori de PhantomJS però Headless Chrome està substituint ràpidament aquest enfocament de WebKit basat en JavaScript. Les proves a Headless Chrome haurien d'anar molt més ràpid i comportar-se més com un navegador real. Els nostres equips, però, han vist que consumeix més memòria que PhantomJS. És molt possible que, gràcies a tots els seus beneficis, **HEADLESS CHROME** es converteixi en l'estàndard **PER PROVES D'INTERFÍCIE**.

Si s'està busca un codificador/descodificador de JSON amb un alt rendiment a Go i Java, val la pena donar un cop d'ull a la llibreria de codi obert **JSONITER**. És compatible amb el paquet de codificació JSON estàndard a Go.

Hem vist millores constants i un augment en l'adopció de Prometheus, l'eina de base de dades de sèries temporals (TSDB) i monitoratge originalment desenvolupada per Soundcloud.

(Prometheus)

Hem vist millores constants i un augment en l'adopció de **PROMETHEUS**, l'eina de base de dades de sèries temporals (TSDB) i monitoratge originalment desenvolupada per Soundcloud. Prometheus suporta, primordialment, el model HTTP basat en la tecnologia de tramesa induïda, però també té suport per alertes, la qual cosa el fa una part important de qualsevol joc d'eines. En el moment d'escriure aquestes línies, Prometheus 2.0 és en estat de prellançament i segueix evolucionant. Els desenvolupadores de Prometheus han centrat els seus esforços en bases de dades de sèries temporals essencials i en la varietat de mètriques disponibles. Grafana s'ha convertit en l'eina de visualització d'instruments d'elecció pels usuaris de Prometheus i l'eina porta incorporat el suport per Grafana. Els nostres equips també creuen que el monitoratge de Prometheus complementa molt bé les capacitats d'indexació i de cerca d'una pila Elastic.

Apex és una eina per compilar, desplegar i gestionar fàcilment funcions AWS Lambda i que permet escriure funcions en llenguatges que encara no estan suportats nativament a AWS com ara Golang, Rust o d'altres. Això s'aconsegueix gràcies a una petita llibreria *shim* de Node.js que crea un procés fill i processa esdeveniments a través de l'entrada i la sortida estàndard (*stdin* i *stdout*). Apex té moltes característiques que milloren l'experiència del desenvolupador i a nosaltres ens agrada, particularment, l'habilitat de provar funcions de manera local i de fer proves dels canvis abans d'aplicar-los als recursos de AWS.

ASSERTJ-SWAGGER, una llibreria de AssertJ, permet validar la conformitat de la implementació d'una API amb les especificacions del seu contracte. Els nostres equips utilitzen assertj-swagger per descobrir problemes quan el punt final de la implementació d'una API canvia sense actualitzar la seva especificació Swagger o no aconsegueix publicar la documentació actualitzada.

Solucionar errors en proves completes a sistemes de CI pot ser una experiència horrible, especialment en mode headless. Cypress és una eina molt útil que ajuda els desenvolupadors a crear fàcilment proves completes i emmagatzema tots els passos en un vídeo en format MP4. En lloc de reproduir el problema en mode headless, els desenvolupadors poden veure el vídeo per solucionar-lo. Cypress no és només un entorn de proves, sinó que és una plataforma molt poderosa. Actualment, hem integrat la seva línia d'ordres (CLI) amb CI headless als nostres projectes.

FLOW és un verificador de tipatge estàtic per JavaScript que permet incorporar la verificació de tipatge al codi de manera incremental. A diferència de Typescript, que és un llenguatge diferent, Flow es pot incorporar de manera gradual a qualsevol base de codi JavaScript que suporti la 5ª, 6ª i 7ª edició de ECMAScript. Nosaltres recomanem incorporar Flow als canals d'integració contínua, començant pel codi que sigui més necessari. Flow millora la claredat del codi i la fiabilitat de la refacció, i troba errors de tipatge en els primers moments del desenvolupament.

Durant l'últim parell d'anys, hem vist un augment constant en la popularitat dels notebooks analítics. Es tracta d'aplicacions inspirades per Mathematica que combinen text, visualització i codi en un document computacional viu. A Radars anteriors hem parlat de GorillaREPL, una variant Clojure d'aquestes aplicacions. No obstant això, l'increment en l'interès per l'aprenentatge automàtic, conjuntament amb l'aparició de Python com a llenguatge preferit pels que treballen en aquest àmbit, ha fet que l'atenció es mogués cap als notebooks Python, d'entre els quals sembla que el que té més acceptació entre els nostres equips és **JUPYTER**.

Kong és una API Gateway de codi obert construïda i patrocinada per Mashape, qui, a més, té una proposta que integra Kong amb les seves pròpies eines d'anàlisi d'APIs i de portals de desenvolupadors. Es poden desplegar com a API Gateway extern o com a *proxy* API intern amb una gran varietat de configuracions. A través dels seus mòduls Nginx, OpenResty proporciona una base forta i potent amb extensions en forma de connectors Lua. Kong pot utilitzar tant PostgreSQL per desplegaments a una sola regió, com Cassandra per configuracions multiregionals. Als nostres desenvolupadors els ha agradat l'alt rendiment de Kong, el seu enfocament de "les API primer" (que permet l'automatització de la seva configuració) i la seva facilitat per desplegar-lo com a contenidor. L'**KONG API GATEWAY**, a diferència de les API Gateway massa ambicioses, no té tantes característiques, però, igualment, compta amb les més bàsiques com el control del trànsit, la seguretat, l'inici de sessió, el monitoratge i l'autenticació. Tenim ganes d'avaluar Kong en una configuració en sidecar en un futur pròxim.

KOPS és una línia d'ordres per crear i gestionar clústers Kubernetes de producció d'alta disponibilitat. Inicialment es va crear per AWS, però ja compta amb suport experimental per més proveïdors. Permet tenir-ho tot configurat i en marxa molt ràpid i, tot i que encara s'han de desenvolupar del tot algunes característiques (com ara actualitzacions contínues), ens ha impressionat molt la seva comunitat.

Un problema recurrent per les aplicacions web amb molt contingut de JavaScript és com fer que els motors de cerca puguin trobar les porcions dinàmiques d'aquestes pàgines.
(Rendertron)

LIGHTHOUSE és una eina escrita per Google per avaluar la fidelitat d'aplicacions web als estàndards de les Aplicacions Web Progressives (PWA). La versió 2.0 de Lighthouse d'aquest any incorpora a la seva selecció d'eines bàsiques les funcions de mètriques de rendiment i control d'accessibilitat. Aquesta funcionalitat afegida s'ha incorporat a les eines estàndard per desenvolupadors de Chrome sota la pestanya "audit". Així, Lighthouse 2.0 és un altre beneficiari del mode headless de Chrome. Això proporciona una alternativa a Pa11y i altres eines similars per dur a terme controls d'accessibilitat a canals d'integració contínua, ja que l'eina es pot executar des d'una línia d'ordres o independentment com a aplicació Node.js.

Un problema recurrent per les aplicacions web amb molt contingut de JavaScript és com fer que els motors de cerca puguin trobar les porcions dinàmiques d'aquestes pàgines. Històricament, els desenvolupadors han utilitzat tota una sèrie de trucs, entre els quals hi ha la renderització al servidor amb React, serveis externs o prerenderitzar el contingut. El mode headless de Google Chrome incorpora, ara, un nou truc a la llista: **RENDERTRON**, una solució de renderització de headless Chrome. Rendertron ajusta una instància de headless Chrome a un contenidor Docker i ho deixa tot llest per desplegar-ho com a servidor HTTP independent. Així, es pot redirigir els bots que no renderitzen JavaScript a aquest servidor per què el renderitzin. Tot i que els desenvolupadors sempre poden desplegar els seu propi *proxy* de headless Chrome i tot el relacionat, Rendertron simplifica el procés de configuració i desplegament i proporciona exemples de codi de programari intermediari per detectar i dirigir bots.

SONOBUOY és una eina de diagnòstic per dur a terme proves completes de conformitat a qualsevol clúster Kubernetes de manera no-destructiva. L'equip de Heptio, fundat per dos dels creadors del projecte Kubernetes, ha creat aquesta eina per assegurar que la gran varietat de distribucions i configuracions de Kubernetes s'ajusten a les millors pràctiques i segueixen els estàndards de codi obert per l'interoperabilitat de clústers. Estem experimentant executant-lo com a part del nostre canal de compilació d'infraestructura com a codi i monitorant contínuament les nostres instal·lacions de Kubernetes per validar el comportament i la salut de tot el clúster.

Si s'està implementant serveis de Java utilitzant l'entorn de treball Spring, és possible que es vulgui tenir en compte **SPRING CLOUD CONTRACT** per les proves de contracte basades en consumidors. L'ecosistema actual d'aquesta eina suporta la verificació de crides de clients i la implementació del servidor sobre el contracte. En comparació amb Pact, una sèrie d'eines de proves de contractes basats en consumidors de codi obert, li manca fer d'intermediari dels contractes i el suport per més llenguatges de programació. Tot i això, s'integra bé amb l'ecosistema Spring per a la redirecció de missatges amb Spring Integration.

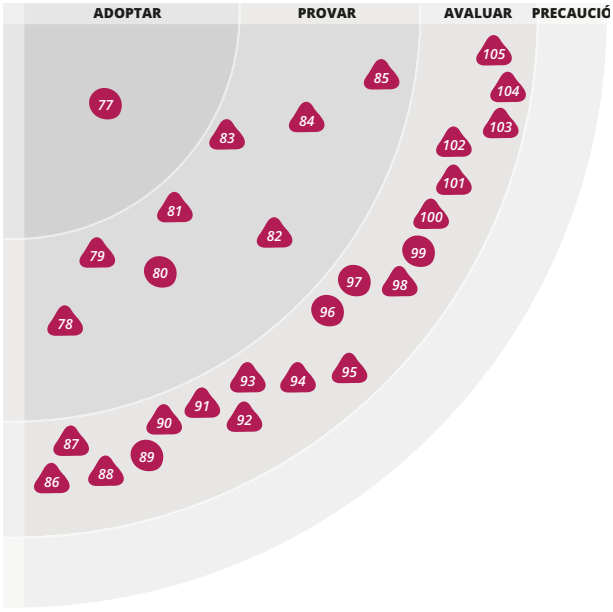
LLENGUATGES I ENTORNS DE TREBALL

ADOPTAR
77. Python 3

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

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

PRECAUCIÓ



A radars anteriors ens ha costat recomanar amb decisió **ANGULAR** perquè era, essencialment, un nou entorn de treball més aviat poc emocionant i que l'únic que tenia era que compartia el nom amb AngularJS, un entorn que ens agradava molt antigament. Amb el temps, però, Angular, que ja es és a la versió 5, no ha deixat de millorar a la vegada que ha mantingut la compatibilitat amb versions anteriors. Diversos dels nostres equips estan utilitzant aplicacions de Angular i, de moment, els agrada el que veuen. En aquest Radar, doncs, movem Angular al quadrant de Provar per simbolitzar que alguns dels nostres equips creuen que és una opció sòlida. La majoria dels nostres equips, però, segueixen preferint React, Vue o Ember abans que Angular.

ASSERTJ és una llibreria de Java que proporciona una interfície fluida per assercions, el que facilita expressar intencionalitat al codi de proves. AssertJ proporciona missatges d'error llegibles, assercions febles i suport millorat per col·leccions i excepcions. Estem veien com alguns equips es passen a AssertJ deixant de banda JUnit en combinació amb Hamcrest.

CSS Grid Layout un sistema de disseny bidimensional basat en una quadrícula que, gràcies a un joc de comportaments de mides predictibles, proporciona mecanismes per dividir l'espai disponible pel disseny en files i columnes.

(CSS Grid Layout)

CSS és la opció predilecte per dissenyar pàgines web tot i que no tenia suport explícit pel disseny. Flexbox va ajudar amb dissenys més simples i unidimensionals, però els desenvolupadors acostumaven a buscar

llibreries i eines per dissenys més complexos. **CSS GRID LAYOUT** és un sistema de disseny bidimensional basat en una quadrícula que, gràcies a un joc de comportaments de mides predictibles, proporciona mecanismes per dividir l'espai disponible pel disseny en files i columnes. La quadrícula no requereix cap tipus de llibreria i funciona bé amb Flexbox i altres elements de CSS. No obstant això, com que només suporta parcialment IE11, ignora als usuaris que encara utilitzen un navegador de Microsoft a Windows 7.

La majoria de grans bases de codi de CSS requereixen esquemes de noms complexos per evitar conflictes al namespace global. **CSS MODULES** soluciona aquest problema creant un namespace local per tots els noms de classe en un sol arxiu CSS. Aquest arxiu s'importa a un mòdul de JavaScript, on les classes CSS es referencien com cadenes. Aleshores, es canvien els noms per cadenes úniques generades al pipeline de muntatge (Webpack, Browserify, etc.), la qual cosa representa un canvi significatiu de responsabilitats. Abans era una persona la que havia de gestionar tots els noms per evitar conflictes; ara, aquesta responsabilitat és cosa de les eines de muntatge. Un petit inconvenient que hem vist en els models CSS és que les proves no es troben, generalment, a nivell local i, per tant, no poden referenciar classes pel nom que s'ha definit a l'arxiu CSS. Nosaltres recomanem utilitzar identificadors o atributs de dades.

Jest proporciona una experiència "configuració zero" i, de fàbrica, compte amb funcions com simulació d'objectes i cobertura de codi

(Jest)

Els nostres equips estan encantats amb els resultats d'utilitzar **JEST** per a proves d'interfície. Proporciona una experiència "configuració zero" i, de fàbrica, compte amb funcions com simulació d'objectes i cobertura de codi. Aquest entorn de proves es pot aplicar no només a aplicacions React, sinó també a altres entorns de JavaScript. Una de les característiques més preuades de Jest són les proves per captures. Aquestes proves serien una molt bona incorporació a la capa superior de la piràmide de proves, però cal recordar que les proves unitàries segueixen sent la millor opció.

L'anunci del suport per Android del llenguatge de ràpid creixement **KOTLIN** li ha donat una empenta important, i és per això que estem seguint el progrés de Kotlin/Native, la habilitat de compilar natiu. Alguns dels beneficis d'aquest llenguatge són "null safety", classes de dades, creació de DSLs i la llibreria de desenvolupament d'Android Anko. Tot i els inconvenients, com són una compilació inicial lenta i dependència a IntelliJ per suportar IDEs de primera classe, recomanem provar aquest llenguatge modern fresc i concís.

SPRING CLOUD segueix evolucionant i incorporant noves funcionalitats interessants. El suport per connectar-se a Kafka Streams del projecte spring-cloud-streams, per exemple, fa que sigui relativament fàcil crear aplicacions basades en missatges amb connectors per Kafka i RabbitMQ. Als nostres equips que l'estan utilitzant els agrada la simplicitat que aporta a l'hora d'utilitzar infraestructures que sovint poden ser complexes com ZooKeeper i el seu suport per problemes comuns que cal tenir en compte quan es creen sistemes distribuïts, com ara traçar amb spring-cloud-sleuth. Cal anar amb compte, però l'estem utilitzant satisfactòriament a diversos projectes.

Històricament, a la documentació sobre Android de Google hi faltaven exemples d'arquitectura i estructura. Això canvia, però, amb el llançament de **ANDROID ARCHITECTURE COMPONENTS**, una sèrie de llibreries que ajuden els desenvolupadors a crear aplicacions amb millor arquitectura. Solucionen problemes recurrents del desenvolupament Android: gestió de cicles de vida, paginació, bases de dades SQLite i persistència de dades sobre canvis de configuració. No cal utilitzar totes les llibreries: es poden seleccionar les que més es necessiten per integrar-les a projectes existents.

Hem vist una gran quantitat d'activitat relacionada amb la realitat augmentada gràcies a **ARKIT I ARCORE**, les llibreries natives de RA de Apple i Google, respectivament. Aquestes llibreries estan acostant les tecnologies de RA a la majoria de consumidors, però, no obstant això, les empreses tenen un gran repte per davant ja que els hi han de trobar usos que aportin solucions que realment millorin l'experiència de l'usuari.

En un moment on hi ha cada vegada menys usuaris que descarreguin noves aplicacions, aplicar una estratègia multiaplicació és molt controvertit. En lloc d'introduir una nova aplicació i patir pel nombre de descàrregues, els equips múltiples han de proporcionar noves funcionalitats a través d'una aplicació que ja utilitzi molta gent, la qual cosa implica un repte arquitectural més gran. **ATLAS** i **BEEHIVE** són solucions modulars per aplicacions Android i iOS, respectivament, que fan possible que equips múltiples que treballen en mòduls físicament aïllats puguin ajuntar i carregar dinàmicament aquests mòduls des d'una aplicació principal. Curiosament, les dues solucions són projectes de codi obert de Alibaba i la raó d'això és que Alibaba també va tenir problemes de reducció del nombre de descàrregues i els mateixos reptes arquitecturals.

La nostra regla d'or per a seleccionar un motor de regles és: no cal cap motor de regles. Hem vist a massa gent lligar-se a un motor de regles difícil de provar per raons incorrectes quan escriure codi personalitzat hagués sigut una opció molt millor. Un cop dit això, però, hem tingut bones experiències amb **CLARA RULES** per aquells casos en què un motor de regles sí que té sentit. Ens agrada que utilitzi codi Clojure simple per expressar i avaluar les regles, la qual cosa vol dir que es poden incloure en el procés de refacció, de proves i de control d'origen. En lloc de perseguir la idea que els empresaris haurien de manipular directament les regles, fomenta la col·laboració entre els experts del negoci i els desenvolupadors.

CSS A JS és una tècnica que permet escriure fulls d'estil CSS en el llenguatge de programació JavaScript. Això fomenta un patró comú d'escriptura amb el component JavaScript al qual aplica, compartint les qüestions de presentació i lògica. Els nous participants d'aquest espai, entre els quals hi ha JSS, emotion i styled-components, depenen de les eines per traduir el codi CSS escrit en JS per separar els fulls d'estil i fer-los adequats pels navegadors. Aquest és el segon intent d'escriure CSS en JS i, a diferència del primer, la tècnica no depèn dels estils *inline*. Això vol dir, doncs, que té el benefici de suportar totes les característiques de CSS, compartir CSS gràcies a l'ecosistema npm i utilitzar els components a diferents plataformes. Els nostres equips han comprovat que styled-components funciona bé amb entorns de treball basats en components, com ara React, i va bé per fer proves unitàries de CSS amb jest-styled-components.

Estem davant d'un nou espai que canvia molt ràpidament. L'enfocament requereix alguns esforços per depurar manualment els noms de classes generats al navegador i pot no ser adequat per projectes on l'arquitectura de la interfície no suporti la reutilització de components i requereixi un estil global.

DIGDAG és una eina per generar, executar, planificar i monitorar canals complexos de dades al núvol. Permet definir els canals en YAML utilitzant la gran quantitat d'operadors disponibles o creant-ne de nous amb la seva API. Digdag té la majoria de les característiques estàndard de les solucions de canals de dades com la gestió de dependències, fluxos de treball modulars per fomentar la reutilització, seguretat en la gestió de secrets i suport multillenguatge. La característica que més ens crida, però, és el suport per l'estratègia multinúvol, que permet moure i unir dades entre AWS RedShift, S3 i BigQuery de Google. Nosaltres creiem que, a mesura que més proveïdors de serveis al núvol ofereixin millors solucions de processament de dades, Digdag (i altres eines similars) seran molt útils per seleccionar la millor opció per cada tasca.

DRUID és un conjunt de connexions JDBC amb moltes característiques de monitoratge. Té un analitzador SQL incorporat que monitoritza la semàntica de les instruccions de SQL que s'executen a la base de dades. Les injeccions o instruccions de SQL sospitoses es bloquegen i registren directament des de la capa JDBC i, a més, es poden combinar les consultes segons la seva semàntica. Aquest projecte de codi obert és responsabilitat de Alibaba i reflecteix el que ha après operant els seus propis sistemes de bases de dades.

Android Architecture Components, una sèrie de llibreries que ajuden els desenvolupadors a crear aplicacions amb millor arquitectura.

(Android Architecture Components)

ECHARTS és una llibreria de lleugera de creació de gràfics que suporta diferents tipus de gràfics i d'interaccions. Com que ECharts es basa totalment en l'API Canvas, té un rendiment increïble, fins i tot amb punts de dades de més de 100k, i també s'ha optimitzat per l'ús mòbil. Conjuntament amb el seu projecte germà, ECharts-X, té suport per dibuixar en 3D. Echarts és un projecte de codi obert de Baidu.

L'habilitat de compilar el llenguatge de programació Go a natiu ha generat interès entre els desenvolupadors per utilitzar-lo a sistemes integrats. **GOBOT** és un entorn de treball pensat per la robòtica, la informàtica física i el internet de les coses, escrit en el llenguatge de programació Go i que suporta una sèrie de plataformes. Nosaltres hem utilitzat l'entorn de treball per projectes de robòtica experimental en els quals no necessitàvem una resposta en temps real, i hem creat controladors de programari de codi obert amb Gobot. Les APIs HTTP de Gobot possibiliten una integració simple del maquinari amb dispositius mòbils per crear aplicacions més riques.

Els nostres equips de desenvolupadors per mòbil estan entusiasmats amb **LEAKCANARY**, una eina dissenyada per detectar aquelles pèrdues de memòria tan molestes a Android i Java. És fàcil de connectar i proporciona notificacions que permeten trobar la causa de la pèrdua. Incorporar aquesta eina pot estalviar la gran quantitat d'hores que implica buscar els errors de memòria a múltiples dispositius.

PYTORCH és la transcripció completa de l'entorn d'aprenentatge automàtic Torch de Lua a Python. Tot i que és bastant nou i immadur en comparació amb Tensorflow, els programadors consideren que és molt més fàcil d'utilitzar. Gràcies a la seva orientació a objectes i la seva implementació nativa de Python, els models es poden expressar de manera més concisa i clara, i es poden depurar mentre s'estan executant. Tot i que recentment han aparegut molts d'aquests entorns de treball, Pytorch disposa del recolzament de Facebook i d'una gran varietat de socis, com ara NVIDIA, la qual cosa hauria d'assegurar el suport per arquitectures CUDA. Els nostres equips creuen que PyTorch és útil per experimentar i desenvolupar models, però encara recorren a TensorFlow per aplicacions a gran escala pel seu rendiment.

SINGLE-SPA és un "entorn de treball d'entorns de treball" de JavaScript que ens permet crear micrinterfícies utilitzant diferents entorns de treball que coexisteixen a una sola aplicació. En termes generals, no recomanem utilitzar més d'un entorn de treball per una sola aplicació però hi ha certs casos en què és inevitable haver-ho de fer. single-spa, per exemple, pot ser bastant útil quan es treballa amb una aplicació heretada a la qual se li vol posar una nova característica, ja sigui amb una versió actual de l'entorn de treball de l'aplicació o utilitzant-ne un de totalment diferent. Tenint en compte la curta vida de molts entorns de treball de JavaScript, creiem que seria necessari que existís una solució que permetés futurs canvis i experimentació en l'entorn de treball sense que això afectés tota l'aplicació. En aquest aspecte, sembla que single-spa pot ser un bon començament.

Hem tingut bones experiències amb Clara rules per aquells casos en què un motor de regles sí que té sentit. Ens agrada que utilitzi codi Clojure simple per expressar i avaluar les regles, la qual cosa vol dir que es poden incloure en el procés de refacció, de proves i de control d'origen.

(Clara rules)

Programar per contractes intel·ligents requereix un llenguatge més expressiu que els sistemes de seqüencies per transaccions. **SOLIDITY** és el llenguatge més popular entre els dissenyats per a contractes intel·ligents. Es tracta d'un llenguatge orientat a contractes i de tipatge estàtic que té una sintaxi similar a la de JavaScript. Proporciona abstraccions per escriure lògica empresarial autoexecutable a contractes intel·ligents. La seva cadena d'eines no para de créixer i, actualment, Solidity és l'elecció principal a la plataforma Ethereum. Tenint en compte la naturalesa immutable dels contractes intel·ligents desplegats, no cal dir que és de vital importància fer proves i auditar rigorosament les dependències.

TENSORFLOW MOBILE fa possible que els desenvolupadors puguin incorporar una gran quantitat de tècniques d'intel·ligència artificia i comprensió a les seves aplicacions de iOS o Android, la qual cosa és especialment útil si es té en compte la diversitat de dades de sensors disponible als telèfons mòbils. Permet carregar models de TensorFlow preentrenats a una aplicació mòbil i aplicar-los a certs punts com ara fotogrames de vídeos en directe, text o veu. Curiosament, els telèfons mòbils són una fantàstica plataforma per implementar aquests models computacionals. Els models de TensorFlow s'exporten i es carreguen com a arxius protobuf, els quals poden suposar algun problema pels implementadors. El format binari de protobuf pot dificultar l'examen de models i requereix que s'enllaci la versió correcte de la llibreria protobuf a l'aplicació mòbil. L'execució local, en canvi, ofereix una alternativa molt atractiva a TensorFlow Serving sense la càrrega de comunicacions que suposa executar-ho remotament.

TRUFFLE és un entorn de desenvolupament que porta una experiència de desenvolupament web moderna a la plataforma Ethereum. S'encarrega de la compilació de contractes intel·ligents, de l'enllaçament i desplegament de llibreries, i, a més, dels artefactes a diferents xarxes de cadenes de blocs. Una de les raons per les quals ens encanta Truffle és que fomenta que la gent escrigui proves pels seus contractes intel·ligents. Com que els contractes intel·ligents s'associen molt sovint amb diners, cal prendre's el procés de proves molt seriosament. Gràcies al seu entorn de proves incorporat i a la seva integració amb TestRPC, Truffle possibilita que s'escrigui el contracte seguint el desenvolupament basat en proves (TDD). Nosaltres creiem que hi haurà més tecnologies similars a Truffle que promoguin la integració contínua a l'àrea de les cadenes de blocs.

WEEX és un entorn de treball per crear aplicacions mòbils multiplataforma utilitzant la sintaxi del component Vue.js. Pels que prefereixen la simplicitat de Vue.js, Weex no només és una opció viable per aplicacions mòbils natives, sinó que també funciona molt bé amb aplicacions més complicades. Hem vist algunes aplicacions més aviat complicades desenvolupades sobre aquest entorn de treball que funcionen molt bé, com ara TMall i Taobao, dos de les aplicacions mòbils més populars a la Xina. Weex ha estat desenvolupat per Alibaba i ja fa part del projecte Apache incubator.

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