ThoughtWorks®

TECHNOLOGY RADAR *VOL.17*

Nossas ideias sobre tecnologias e tendências que estão moldando o futuro

thoughtworks.com/pt/radar

#TWTechRadar

CONTRIBUIÇÕES

O Technology Radar é preparado pelo Conselho Consultivo de Tecnologia da ThoughtWorks, composto por:

Rebecca Parsons (Diretora de Tecnologia) | Martin Fowler (Cientista-chefe)

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

TRADUÇÃO: Alexey Villas Bôas, Maitê Balhester, Marcos Brizeno, Marco Valtas e Paula Ribas

O QUE HÁ DE NOVO?

Estes são os temas em destaque nessa edição:

CHINA E SOFTWARE LIVRE

A maré está virando. Em razão de mudanças tanto de atitude quanto políticas, grandes empresas chinesas como Alibaba e Baidu estão lançando frameworks, ferramentas e plataformas de código aberto com rapidez. O crescimento de seus ecossistemas de software está acelerando rapidamente à medida que se expandem economicamente.

Dado o número de projetos de software nos efervescentes e massivos mercados chineses, espera-se que o número e a qualidade dos projetos de código aberto disponíveis no GitHub e em outros sites de código aberto aumente. Mas por qual motivo as empresas chinesas abririam o código de tantos ativos? Como é o caso dos mercados de software mais populares, como no Vale do Silício, a concorrência para pessoas desenvolvedoras é acirrada, e oferecer uma maior remuneração só funciona até certo ponto.

A perspectiva de trabalhar em projetos de código aberto de ponta com outras pessoas desenvolvedoras talentosas é um incentivo universal. Esperamos que grandes inovações de código aberto continuem a tendência dos arquivos README escritos em chinês primeiro e em inglês depois.

KUBERNETES, A ESCOLHA PARA ORQUESTRAÇÃO DE CONTÊINERES

Um grande número de entradas no Radar girou em torno do Kubernetes e de sua presença cada vez mais dominante em muitos projetos. Parece que o ecossistema de desenvolvimento de software está adotando Kubernetes e ferramentas relacionadas como padrão para resolver os problemas comuns relacionados à implantação, escala e operação de contêineres.

Entradas do Radar como GKE, Kops e Sonobuoy introduzem serviços e ferramentas de plataforma gerenciada que melhoram a experiência geral de adotar e executar Kubernetes. De fato, sua capacidade de simplesmente executar vários contêineres como uma unidade de agendamento permite o uso da malha de serviços e de sidecars para segurança de endpoints.

O Kubernetes tornou-se o sistema operacional padrão para contêineres: muitos provedores de nuvem aproveitaram sua arquitetura aberta e modular para adotar e executar o Kubernetes, enquanto as ferramentas utilizam suas APIs abertas para acessar abstrações como cargas de trabalho, clusters, configuração e armazenamento.

Vemos mais produtos utilizando Kubernetes como um ecossistema, tornando-o o próximo nível de abstração depois de microsserviços e contêineres. Essa é uma evidência adicional de que as pessoas desenvolvedoras podem alavancar estilos de arquitetura modernos com sucesso, apesar da complexidade inerente aos sistemas distribuídos.

NUVEM COMO O NOVO NORMAL

O outro tópico onipresente nas conversas enquanto esta edição era criada tinha uma natureza distintamente "nebulosa". À medida que os provedores de nuvem se tornam mais capacitados e alcançam paridade de funcionalidades, o modelo de nuvem pública está se tornando o novo padrão para muitas organizações.

Em vez de perguntar "Por que na nuvem?", muitas empresas agora perguntam "Por que *não* na nuvem?" ao iniciar novos projetos. Certamente, alguns tipos de software ainda exigem sistemas locais, mas à medida que os preços caem e as capacidades se expandem, o desenvolvimento nativo na nuvem torna-se cada vez mais viável.

Embora exista paridade de funcionalidades básicas entre os principais provedores de nuvem, cada um deles também oferece diferenciação em tipos específicos de soluções. Assim, vemos empresas se beneficiando do uso de vários provedores diferentes por meio de Polycloud, escolhendo as capacidades específicas das plataformas que melhor atendam às necessidades de suas clientes.

CONFIANÇA EM BLOCKCHAIN MAIS UNIFORMEMENTE DISTRIBUÍDA

Apesar do caos que permeia as criptomoedas no mercado, muitas de nossas clientes estão encontrando maneiras de alavancar soluções de blockchain para livros de registro distribuídos e contratos inteligentes. Várias entradas do Radar evidenciam a maturidade no uso de tecnologias relacionadas a blockchain, fornecendo maneiras cada vez mais interessantes de implementar contratos inteligentes, com uma variedade de técnicas e linguagens de programação.

Blockchain resolve o antigo problema da confiança distribuída e dos livros de registo permanentes e compartilhados. Hoje, as empresas estão aumentando a confiança dos usuários na mecânica subjacente das implementações de blockchain. Muitas indústrias têm diferentes problemas de confiança distribuída; acreditamos que as soluções blockchain continuarão a encontrar maneiras de resolvê-las.

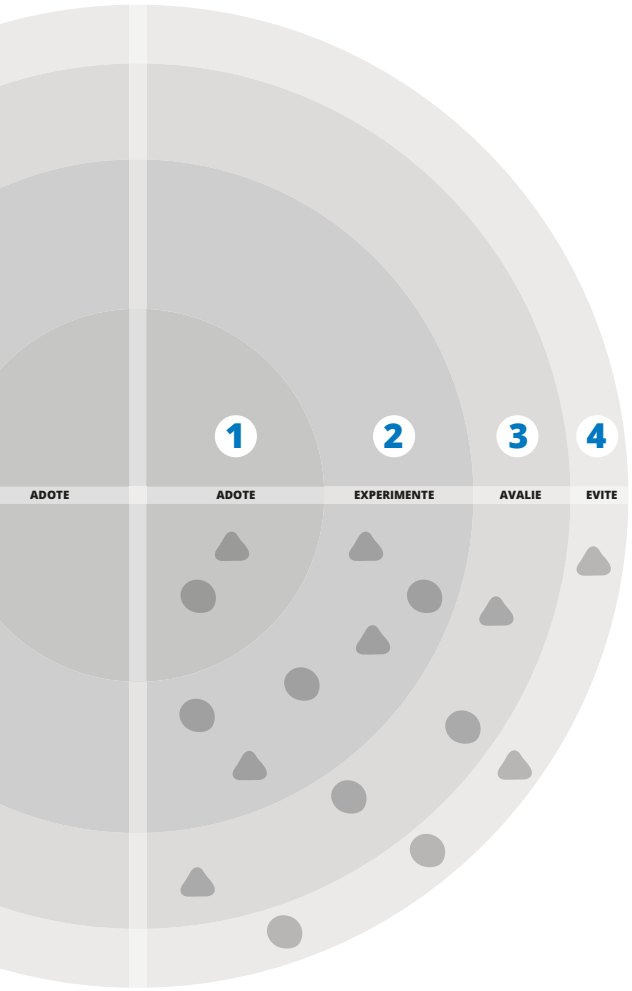
SOBRE O RADAR

'ThoughtWorkers' são pessoas apaixonadas por tecnologia. Nós desenvolvemos, pesquisamos, testamos, contribuímos com código livre, escrevemos e visamos a sua constante melhoria — para todas as pessoas. Nossa missão é liderar e promover a excelência de software e revolucionar a indústria de TI. Nós criamos e compartilhamos o Technology Radar da ThoughtWorks para apoiar essa missão. O Conselho Consultivo de Tecnologia (Technology Advisory Board, ou TAB), um grupo de líderes experientes em tecnologia da ThoughtWorks, é responsável por criar o Radar. O grupo se reúne regularmente para discutir a estratégia global de tecnologia para a empresa e as tendências tecnológicas que impactam significativamente a nossa indústria.

O Radar captura o resultado das discussões do TAB em um formato que procura oferecer valor a uma ampla gama de pessoas interessadas, de CTOs a pessoas que desenvolvem. O conteúdo é concebido para ser um resumo conciso.

Nós encorajamos você a explorar essas tecnologias para obter mais detalhes. O Radar é gráfico por natureza, agrupando os itens em técnicas, ferramentas, plataformas e linguagens & frameworks. Quando itens do radar puderem ser classificados em mais de um quadrante, escolhemos aquele que parece mais adequado. Além disso, agrupamos esses itens em quatro anéis para refletir a nossa posição atual em relação a eles.

Para mais informações sobre o Radar, veja:
thoughtworks.com/pt/radar/faq



RADAR EM UM RELANCE

1 ADOTE

Acreditamos firmemente que a indústria deveria adotar esses itens. Nós os usamos quando são apropriados em nossos projetos.

2 EXPERIMENTE

Vale a pena ir atrás. É importante entender como desenvolver essa capacidade. As empresas devem experimentar esta tecnologia em um projeto que possa lidar com o risco.

3 AVALIE

Vale a pena explorar com o objetivo de compreender como isso afetará sua empresa.

4 EVITE

Prossiga com cautela.

▲ NOVO OU MODIFICADO

● SEM MODIFICAÇÃO

Itens novos ou que sofreram alterações significativas desde o último Radar são representados como triângulos, enquanto os itens que não mudaram são representados como círculos.

! Nosso Radar é um olhar para o futuro. Para abrir espaço para novos itens, apagamos itens em que não houve mudança recentemente, o que não é uma representação de seu valor, mas uma solução para nossa limitação de espaço.

O RADAR

TÉCNICAS

ADOTE
1. Registros de decisões de arquiteturas leves

EXPERIMENTE
2. Aplicar gestão de produtos a plataformas internas *NOVO*
3. Função de aptidão arquitetural *NOVO*
4. Padrão de bolha autônoma *NOVO*
5. Engenharia de Caos *NOVO*
6. Desacoplar gerenciamento de segredos do código-fonte
7. DesignOps *NOVO*
8. Legado encaixotado
9. Micro frontends
10. Pipelines para infraestrutura como código *NOVO*
11. Arquitetura sem servidor
12. TDD em contêineres *NOVO*

AVALIE
13. Operações de TI algorítmicas *NOVO*
14. Ethereum para aplicações descentralizadas *NOVO*
15. Streaming de eventos como fonte única da verdade *NOVO*
16. Times de produto de engenharia de plataforma
17. Polycloud *NOVO*
18. Malha de serviços *NOVO*
19. Sidecars para segurança de endpoints *NOVO*
20. Os três Rs de segurança *NOVO*

EVITE
21. Instância única de integração contínua para todos os times
22. CI de faz de conta
23. Ambientes de teste de integração empresariais
24. Recriando anti-padrões ESB com Kafka *NOVO*
25. Geração de código baseada em especificações

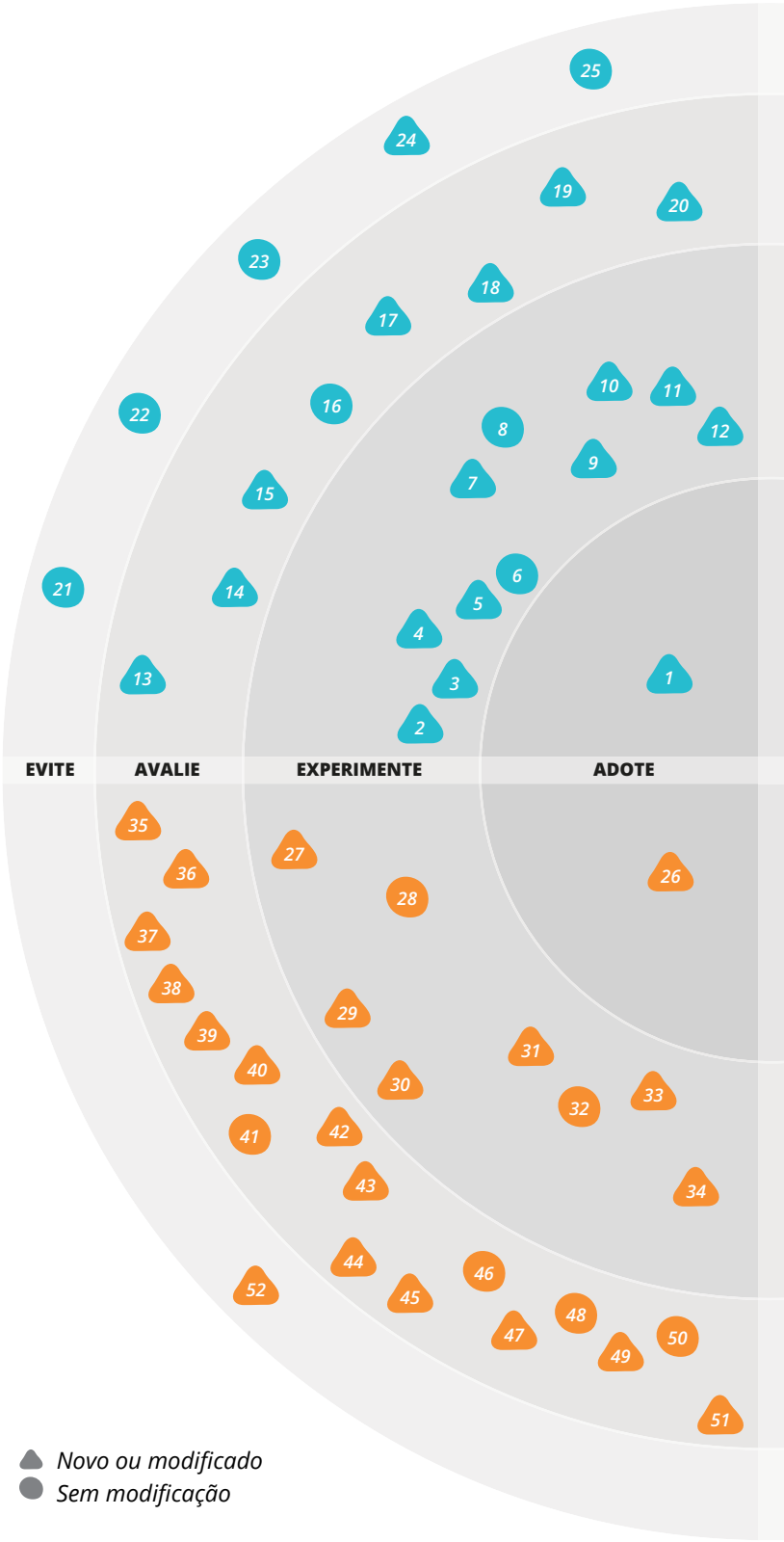
PLATAFORMAS

ADOTE
26. Kubernetes

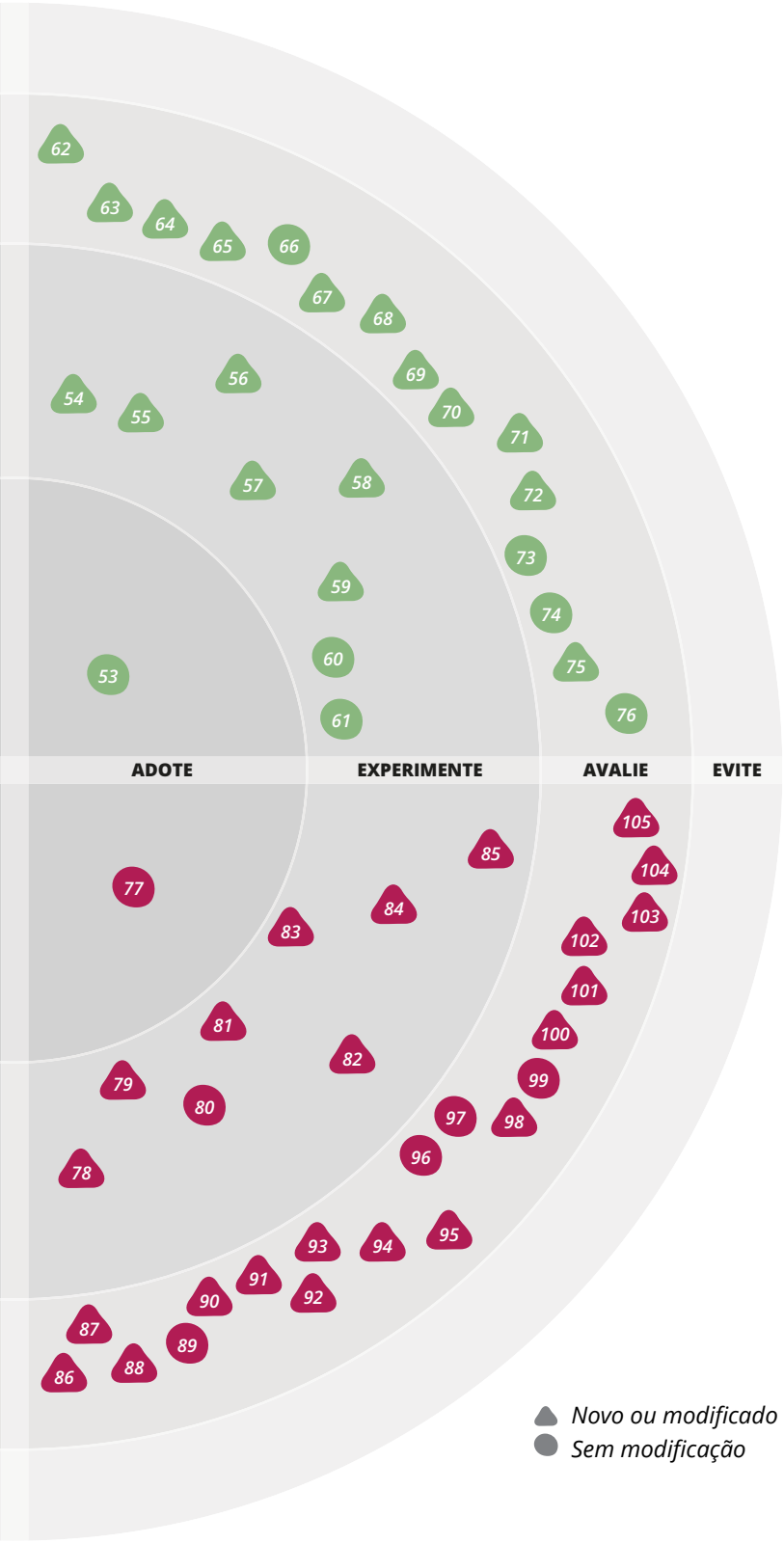
EXPERIMENTE
27. .NET Core
28. AWS Device Farm
29. Flood IO *NOVO*
30. Google Cloud Platform *NOVO*
31. Keycloak
32. OpenTracing
33. Unity além dos jogos
34. WeChat *NOVO*

AVALIE
35. Azure Service Fabric *NOVO*
36. Cloud Spanner *NOVO*
37. Corda *NOVO*
38. Cosmos DB *NOVO*
39. DialogFlow
40. GKE *NOVO*
41. Hyperledger
42. Kafka Streams
43. Language Server Protocol *NOVO*
44. LoRaWAN *NOVO*
45. MapD *NOVO*
46. Mosquitto
47. Netlify *NOVO*
48. PlatformIO
49. TensorFlow Serving *NOVO*
50. Plataformas conversacionais
51. Contêineres do Windows *NOVO*

EVITE
52. API Gateways excessivamente ambiciosos



O RADAR



FERRAMENTAS

ADOTE
53. fastlane

EXPERIMENTE
54. Buildkite *NOVO*
55. CircleCI *NOVO*
56. gopass *NOVO*
57. Headless Chrome para testes de front-end *NOVO*
58. jsoniter *NOVO*
59. Prometheus
60. Scikit-learn
61. Serverless Framework

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

EVITE

LINGUAGENS E FRAMEWORKS

ADOTE
77. Python 3

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

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

EVITE

TÉCNICAS

ADOTE
1. Registros de decisões de arquiteturas leves

EXPERIMENTE
2. Aplicar gestão de produtos a plataformas internas *NOVO*
3. Função de aptidão arquitetural *NOVO*
4. Padrão de bolha autônoma *NOVO*
5. Engenharia de Caos *NOVO*
6. Desacoplar gerenciamento de segredos do código-fonte
7. DesignOps *NOVO*
8. Legado encaixotado
9. Micro frontends
10. Pipelines para infraestrutura como código *NOVO*
11. Arquitetura sem servidor
12. TDD em contêineres *NOVO*

AVALIE
13. Operações de TI algorítmicas *NOVO*
14. Ethereum para aplicações descentralizadas *NOVO*
15. Streaming de eventos como fonte única da verdade *NOVO*
16. Times de produto de engenharia de plataforma
17. Polycloud *NOVO*
18. Malha de serviços *NOVO*
19. Sidecars para segurança de endpoints *NOVO*
20. Os três Rs de segurança *NOVO*

EVITE
21. Instância única de integração contínua para todos os times
22. CI de faz de conta
23. Ambientes de teste de integração empresariais
24. Recriando anti-padrões ESB com Kafka *NOVO*
25. Geração de código baseada em especificações

Muita documentação pode ser substituída por códigos e testes altamente legíveis. Em um universo de arquitetura evolutiva, no entanto, é importante registrar determinadas decisões de design para o proveito de futuros membros do time e também para supervisão externa. Os **REGISTROS LEVES DE DECISÕES ARQUITETURAIS** são uma técnica para capturar decisões de arquitetura importantes juntamente com seu contexto e suas consequências. Nós recomendamos o armazenamento desses detalhes em uma fonte de controle, em vez de uma wiki ou um website, visto que eles podem oferecer um registro que permanece sincronizado com o próprio código. Para a maioria dos projetos, não vemos uma razão para não usar essa técnica.

Temos observado um aumento acentuado no interesse pelo tópico de plataformas digitais nos últimos 12 meses. Empresas buscando lançar novas soluções digitais de maneira rápida e eficiente estão construindo plataformas internas, que proporcionam aos times acesso self-service a APIs de negócios, ferramentas, conhecimento e suporte necessários para construir e operar suas próprias soluções. Consideramos essas plataformas mais eficazes quando são tratadas com a mesma atenção que uma oferta de produto externo. **APLICAR GESTÃO DE PRODUTOS A PLATAFORMAS INTERNAS** significa estabelecer empatia com clientes internas (leia-se pessoas desenvolvedoras) e colaborar com elas no design. Gerentes de produtos de plataforma definem roadmaps e asseguram que a plataforma entregue valor ao negócio e melhore a experiência de quem desenvolve. Algumas até criam uma identidade da marca para a plataforma interna e a usam para promover os benefícios entre suas colegas. Gerentes de produtos de plataforma zelam pela qualidade da plataforma, coletam métricas de uso e a melhoram continuamente ao longo do tempo. Tratar a plataforma como um produto ajuda a criar um ecossistema próspero e evita a armadilha de construir mais uma arquitetura orientada a serviço subaproveitada e estagnada.

Emprestada da computação evolutiva, uma função de aptidão é usada para resumir o quão próxima uma determinada solução de design está de alcançar os objetivos predefinidos.

(Função de aptidão arquitetural)

Emprestada da computação evolutiva, uma função de aptidão é usada para resumir o quão próxima uma determinada solução de design está de alcançar os objetivos predefinidos. Ao definir um algoritmo evolutivo, a pessoa designer busca um algoritmo "melhor"; a função de aptidão define o que "melhor" significa neste contexto. Uma **FUNÇÃO DE APTIDÃO ARQUITETURAL**, conforme definido em Construindo Arquiteturas Evolutivas, fornece uma avaliação objetiva da integridade de algumas características arquiteturais, que podem abranger critérios de verificação existentes, como testes de unidade, métricas, monitores, etc. Acreditamos que as pessoas arquitetas podem comunicar, validar e preservar características arquiteturais de forma automática e contínua, que é fundamental para a construção de arquiteturas evolutivas.

Muitas organizações com as quais trabalhamos estão se esforçando para usar abordagens de engenharia modernas para construir novas capacidades e recursos, ao mesmo tempo também tendo que coexistir com uma vasta gama de sistemas legados. Uma estratégia antiga que, com base em nossa experiência, tornou-se cada vez mais útil nesses cenários, é o **PADRÃO DE BOLHA AUTÔNOMA** de Eric Evans. Essa abordagem envolve criar um contexto novo para o desenvolvimento de novas aplicações, que esteja protegido das amarras do universo legado. Essa é uma etapa que vai além de apenas usar uma camada anticorrupção. Ela dá ao novo contexto de bolha controle total sobre seus dados de apoio, que são então atualizados de forma assíncrona com os sistemas legados. É necessário algum esforço para proteger os limites da bolha e manter ambos os universos consistentes, porém a autonomia resultante e a redução na fricção de desenvolvimento são um primeiro passo corajoso em direção a uma arquitetura futura modernizada.

Em edições anteriores do Radar, falamos sobre o uso do Chaos Monkey da Netflix para testar como um sistema em execução consegue lidar com interrupções na produção ao desabilitar instâncias aleatoriamente e medir os resultados. **ENGENHARIA DE CAOS** é um termo emergente para a aplicação mais ampla desta técnica. Ao executar experimentos em sistemas distribuídos em produção, conseguimos ter segurança que os sistemas funcionam conforme o esperado sob condições turbulentas. Um bom lugar para começar a entender essa técnica é o site Princípios de Engenharia de Caos.

Inspirado pelo movimento DevOps, o **DESIGNOPS** é uma mudança cultural e um conjunto de práticas que permitem que as pessoas em uma organização redesenhem produtos de forma contínua, sem comprometer a qualidade, a consistência do serviço ou a autonomia do time. O DesignOps defende a criação e a evolução de uma infraestrutura de design que minimize o esforço necessário para criar novos conceitos e variações de IU e para estabelecer um ciclo de feedback rápido e confiável com usuários finais. Com ferramentas como o Storybook promovendo uma colaboração próxima, a necessidade de análises antecipadas e delegação de especificações é reduzida a um mínimo absoluto. Com o DesignOps, o design está deixando de ser uma prática específica para se tornar uma parte do trabalho de todas as pessoas.

Temos observado benefícios significativos na introdução de arquiteturas de microsserviços, que vêm permitindo aos times escalar a entrega de serviços implantados e mantidos de forma independente. Infelizmente, também temos observado muitos times criarem monólitos de frontend — uma aplicação para navegador única, grande e extensa — em cima de seus serviços de backend. Nossa abordagem preferida (e verificada) é dividir o código baseado em navegador em **MICRO FRONTENDS**. Nessa abordagem, a aplicação web é dividida em funcionalidades, e cada funcionalidade é de propriedade, do frontend ao backend, de um time diferente. Isso assegura que cada funcionalidade seja desenvolvida, testada e implantada independentemente de outras funcionalidades. Existem diversas técnicas para recombinar as funcionalidades — às vezes como páginas, às vezes como componentes — para uma experiência de usuário consistente.

O uso de pipelines de entrega contínua para orquestrar o processo de implantação de software tornou-se um conceito mainstream. No entanto, o teste automático de alterações no código de infraestrutura não é

tão amplamente compreendido. As ferramentas de integração contínua (CI) e de entrega contínua (CD) podem ser usadas para testar a configuração do servidor (por exemplo, Chef cookbooks, módulos Puppet, Ansible playbooks), criação de imagens do servidor (por exemplo, Packer), provisionamento de ambiente (por exemplo, Terraform, CloudFormation) e integração de ambientes. O uso de **PIPELINES PARA INFRAESTRUTURA COMO CÓDIGO** permite que os erros sejam encontrados antes das alterações serem aplicadas aos ambientes operacionais — incluindo ambientes utilizados para desenvolvimento e testes. Eles também oferecem uma maneira de assegurar que as ferramentas de infraestrutura sejam executadas de forma consistente, a partir de agentes CI/CD, em vez de serem executadas a partir de estações de trabalho individuais. Contudo, alguns desafios permanecem, como os longos ciclos de feedback associados aos contêineres ativos e às máquinas virtuais. Ainda assim, achamos que essa é uma técnica valiosa.

Ferramentas de CI e CD podem ser usadas para testar a configuração do servidor, criação de imagens do servidor, provisionamento de ambiente e integração de ambientes.

(Pipelines para infraestrutura como código)

O uso de **ARQUITETURA SEM SERVIDOR** rapidamente se tornou uma abordagem aceita para organizações que estão implantando aplicações na nuvem, com uma abundância de opções disponíveis para implantação. Até mesmo organizações tradicionalmente conservadoras estão fazendo uso parcial de algumas tecnologias sem servidor. A maioria das discussões tem como foco Funções como Serviço (por exemplo, AWS Lambda, Google Cloud Functions, Azure Functions), ao passo em que os padrões apropriados para uso ainda estão surgindo. A implantação de funções sem servidor inegavelmente elimina o esforço não-trivial que tradicionalmente é empregado na configuração e orquestração do servidor e do sistema operacional. As funções sem servidor, no entanto, não se adequam a todos os requisitos. No estágio atual você deve se preparar para voltar à implantação de contêineres ou mesmo instâncias de servidor para requisitos específicos.

Enquanto isso, os outros componentes de uma arquitetura sem servidor, como Backend como Serviço, tornaram-se quase uma opção padrão.

Em razão de seus benefícios muitos times adotaram práticas de desenvolvimento guiadas por testes para escrever códigos de aplicações. Outros optaram por contêineres para empacotar e implantar seus softwares, e usar scripts automatizados para construir os contêineres é uma prática aceita. O que temos visto ser feito por alguns times até agora é combinar as duas tendências e escrever os scripts de contêiner usando testes. Com frameworks como Serverspec e Goss, é possivel expressar a funcionalidade pretendida para tanto para contêineres isolados quanto orquestrados, com ciclos de feedback curtos. Isso significa que é possível utilizar os mesmos princípios que defendemos para código aplicando **TDD EM CONTÊINERES**. Nossa experiência inicial em fazê-lo foi muito positiva.

A quantidade de dados coletados por operações de TI vem aumentando há anos. A tendência de adoção de microsserviços, por exemplo, significa que mais aplicações estão gerando seus próprios dados operacionais, e ferramentas como Splunk, Prometheus ou a stack ELK tornam mais fácil o processo de armazenar e processar dados mais tarde, para obtenção de insights operacionais. Quando combinadas com ferramentas de aprendizagem de máquina cada vez mais democratizadas, é inevitável que pessoas operadoras comecem a incorporar modelos estatísticos e algoritmos de classificação treinados em seus conjuntos de ferramentas. Embora esses algoritmos estejam disponíveis há anos e várias tentativas de automatizar o gerenciamento de serviços tenham sido feitas, estamos apenas começando a entender como máquinas e humanos podem colaborar para identificar interrupções mais cedo ou apontar a fonte das falhas. Ainda que exista um risco de exaltar demasiadamente **OPERAÇÕES DE TI ALGORÍTMICAS**, a melhoria constante nos algoritmos de aprendizagem de máquinas irá inevitavelmente mudar o papel dos seres humanos na operação dos datacenters no futuro.

Blockchains foram amplamente divulgadas como a solução para todas as coisas relacionadas a fintech, do banco à moeda digital à transparência da cadeia de suprimentos. Nós já falamos sobre o Ethereum, devido

ao seu conjunto de recursos, que inclui contratos inteligentes. Agora, temos visto mais desenvolvimento usando **ETHEREUM PARA APLICAÇÕES DESCENTRALIZADAS** em outras áreas. Embora ainda seja uma tecnologia muito recente, tem nos encorajado vê-la sendo usada para construir aplicações descentralizadas para além de cryptomoedas e bancos.

Blockchains foram amplamente divulgadas como a solução para todas as coisas relacionadas a fintech, do banco à moeda digital à transparência da cadeia de suprimentos.

(Ethereum para aplicações descentralizadas)

Conforme as plataformas de streaming de eventos, como Apache Kafka, crescem em popularidade, muitas pessoas as consideram uma forma avançada de enfileirar mensagens, usada somente para transmitir eventos. Mesmo quando usada dessa forma, o streaming de eventos tem seus benefícios em relação ao enfileiramento de mensagens tradicional. Entretanto, nos interessa mais como as pessoas usam **STREAMING DE EVENTOS COMO FONTE ÚNICA DA VERDADE**, com plataformas (especialmente Kafka) para armazenamento primário de dados como eventos imutáveis. Um serviço com um design de Event Sourcing, por exemplo, pode usar Kafka como seu armazenamento de eventos, e esses eventos estarão, então, disponíveis para outros serviços usarem. Essa técnica tem o potencial de reduzir esforços duplicados entre a persistência e integração locais.

Os principais provedores de nuvem (Amazon, Microsoft e Google) estão disputando uma corrida agressiva para manter a paridade em capacidades fundamentais, enquanto seus produtos se diferenciam apenas marginalmente. Isso está fazendo algumas organizações adotarem uma estratégia de **POLYCLOUD** — em vez de "apostar tudo" em um provedor, elas passam diferentes tipos de carga de trabalho para diferentes provedores em uma abordagem de melhor opção. Isso pode envolver, por exemplo, colocar serviços padrão em AWS, mas usar Google para aprendizagem de máquina, Azure para aplicações .NET que usam SQLServer, ou potencialmente usar a solução Ethereum Consortium Blockchain. Isso é diferente de uma estratégia agnóstica de nuvem de ter por objetivo a portabilidade entre provedores, que é dispendiosa e força o pensamento de menor denominador comum. Em vez disso, Polycloud tem como foco o uso do melhor que cada nuvem oferece.

Conforme grandes organizações transicionam para times mais autônomos que são proprietários e operam seus próprios microsserviços, como elas podem assegurar a consistência e compatibilidade necessárias entre esses serviços sem depender de uma infraestrutura de hospedagem centralizada? Para trabalhar em conjunto de forma eficiente, mesmo microsserviços autônomos precisam se alinhar com alguns padrões organizacionais. Uma **MALHA DE SERVIÇOS** oferece descoberta, segurança, rastreamento, monitoramento e processamento de falhas consistentes sem a necessidade de um recurso compartilhado como um API gateway ou ESB. Uma implementação típica envolve um processo de proxy reverso leve implantado juntamente com cada processo de serviço, possivelmente em um contêiner separado. Esses proxies se comunicam com registros de serviço, provedores de identidade, agregadores de log, e assim por diante. A interfuncionalidade e observabilidade são obtidas por meio de uma implementação compartilhada desse proxy, mas não uma instância de tempo de execução compartilhada. Nós defendemos uma abordagem descentralizada para a gestão de microsserviços há algum tempo, e estamos felizes em ver esse padrão consistente surgir. Os projetos de código aberto, como linkerd e Istio continuarão a amadurecer e tornar a malha de serviços ainda mais fácil de ser implementada.

Uma malha de serviços oferece descoberta, segurança, rastreamento, monitoramento e processamento de falhas consistentes sem a necessidade de um recurso compartilhado como um API gateway ou ESB.

(Malha de serviços)

A arquitetura de microsserviços, com uma grande quantidade de serviços demonstrando seus recursos e capacidades por meio de APIs e uma superfície de ataque aumentada, exige uma arquitetura de

segurança de confiança zero — "nunca confie, sempre verifique". No entanto, a imposição de controles de segurança para a comunicação entre serviços é muitas vezes negligenciada, devido à maior complexidade do código de serviço e à falta de bibliotecas e suporte de idiomas em um ambiente poliglota. Para superar essa complexidade, alguns times delegam a segurança a um sidecar fora do processo — um processo ou um contêiner que é implantado e agendado com cada serviço compartilhando o mesmo contexto de execução, hospedagem e identidade. Os sidecars implementam capacidades de segurança, como criptografia transparente da comunicação e terminação TLS (Transport Layer Security), bem como autenticação e autorização do serviço de chamadas ou do usuário final. Recomendamos que você avalie usar Istio, linkerd ou Envoy antes de implementar seus próprios **SIDECARS PARA SEGURANÇA DE ENDPOINTS**.

Abordagens tradicionais de segurança corporativa geralmente concentram-se em travar e diminuir a velocidade de mudança. No entanto, sabemos que quanto mais tempo um invasor tiver para comprometer um sistema, maior será o dano potencial. Os três Rs da segurança corporativa — rotacionar, reparar e repavimentar — se aproveitam da automação da infraestrutura e da entrega contínua para eliminar oportunidades de ataque. Rotacionar credenciais, aplicando patches assim que estiverem disponíveis e reconstruir sistemas a partir de um estado conhecido e seguro — tudo em questão de minutos ou horas — dificulta o sucesso de invasores. A técnica dos **TRÊS Rs DE SEGURANÇA** é viabilizada pelo advento das arquiteturas modernas nativas da nuvem. Quando aplicações são implantadas como contêineres, e construídas e testadas por meio de um pipeline completamente automatizado, um patch de segurança é apenas uma outra pequena versão que pode ser enviada por meio do pipeline com um clique. Naturalmente, ao manter as melhores práticas de sistemas distribuídos, as pessoas desenvolvedoras precisam projetar suas aplicações para serem resistentes a quedas inesperadas de servidor. Isso é semelhante ao impacto de implementação do Chaos Monkey em seu ambiente.

Kafka está se tornando muito popular como uma solução de mensagens, e junto dela, o Kafka Streams está na vanguarda de uma onda de interesse em arquiteturas de streaming. Infelizmente, conforme elas começam a inserir Kafka no centro de suas plataformas de dados e aplicações, observamos que algumas organizações estão **RECRIANDO ANTI-PADRÕES ESB COM KAFKA** ao centralizar os componentes do ecossistema Kafka — como conectores e processadores de stream — ao invés de permitir que esses componentes ganhem vida com os times de produto ou serviço. Isso nos lembra de anti-padrões ESB extremamente problemáticos, nos quais mais e mais lógica, orquestração e transformação foram forçados em um ESB gerenciado de forma centralizadora, criando uma dependência significativa em um time centralizado. Nós chamamos a atenção para isso para dissuadir implementações futuras desse padrão falho.

PLATAFORMAS

ADOTE
26. Kubernetes

EXPERIMENTE
27. .NET Core
28. AWS Device Farm
29. Flood IO *NOVO*
30. Google Cloud Platform *NOVO*
31. Keycloak
32. OpenTracing
33. Unity além dos jogos
34. WeChat *NOVO*

AVALIE
35. Azure Service Fabric *NOVO*
36. Cloud Spanner *NOVO*
37. Corda *NOVO*
38. Cosmos DB *NOVO*
39. DialogFlow
40. GKE *NOVO*
41. Hyperledger
42. Kafka Streams
43. Language Server Protocol *NOVO*
44. LoRaWAN *NOVO*
45. MapD *NOVO*
46. Mosquitto
47. Netlify *NOVO*
48. PlatformIO
49. TensorFlow Serving *NOVO*
50. Plataformas conversacionais
51. Contêineres do Windows *NOVO*

EVITE
52. API Gateways excessivamente ambiciosos

Desde a última vez que mencionamos **KUBERNETES** no Radar, esta se tornou a solução padrão para a maioria de nossas clientes na implementação de contêineres em um cluster de máquinas. As alternativas não conquistaram tanta atenção e, em alguns casos, nossas clientes estão até mesmo mudando seu 'mecanismo' para Kubernetes. Kubernetes tornou-se a plataforma de orquestração de contêineres de escolha para as principais plataformas de nuvem públicas, incluindo Azure Container Service da Microsoft e Google Cloud (veja o blip GKE). E há muitos produtos úteis que enriquecem o ecossistema em rápido crescimento do Kubernetes. Plataformas que tentam esconder Kubernetes sob uma camada de abstração, no entanto, ainda têm que se mostrar à altura.

Temos observado um aumento na adoção de **.NET CORE**, o framework de software multiplataforma de código aberto. O .NET Core permite o desenvolvimento e a implantação de aplicativos .NET em Windows, MacOS e Linux. Com o lançamento do .NET Standard 2.0 ampliando o número de APIs padrão em plataformas .NET, o caminho de migração para o .NET Core tornou-se mais claro. As questões relacionadas ao suporte da biblioteca no .NET Core estão se tornando menos problemáticas, e as ferramentas multiplataforma de primeira classe estão disponíveis agora, permitindo que o desenvolvimento seja produtivo em plataformas que não sejam Windows. As imagens do Blessed Docker são fornecidas para facilitar a integração dos serviços do .NET Core em um ambiente conteinerizado. As orientações positivas na comunidade e o feedback de nossos projetos indicam que o .NET Core está pronto para uso geral.

O teste de carga ficou mais fácil com o amadurecimento de ferramentas como Gatling e Locust. Ao mesmo tempo, as infraestruturas de nuvem flexíveis tornam possível a simulação de um grande número de instâncias de clientes. Estamos felizes em ver Flood e outras plataformas de nuvem irem além ao alavancar essas tecnologias. **FLOOD IO** é um serviço de teste

de carga SaaS que ajuda a distribuir e executar scripts de teste em centenas de servidores na nuvem. Nossa equipe considera simples a migração de teste de desempenho para Flood reutilizando scripts de Gatling existentes.

Como a **GOOGLE CLOUD PLATFORM** (GCP) cresceu em termos de regiões geográficas disponíveis e maturidade dos serviços, clientes em todo o mundo agora podem considerá-lo seriamente para sua estratégia na nuvem. Em algumas áreas, a GCP atingiu paridade de funcionalidades com sua principal concorrente, Amazon Web Services, enquanto em outras áreas se diferenciou — de forma notável com plataformas de aprendizagem de máquina acessíveis, ferramentas de engenharia de dados e Kubernetes operável como uma solução de serviço (GKE). Na prática, nossos times só têm elogios à experiência de desenvolvimento trabalhando com as ferramentas e APIs da GCP.

Google Cloud Platform (GCP) cresceu em termos de regiões geográficas disponíveis e maturidade dos serviços, clientes em todo o mundo agora podem considerá-lo seriamente para sua estratégia na nuvem.

(Google Cloud Platform)

Em uma arquitetura de microsserviços ou qualquer outra arquitetura distribuída, uma das necessidades mais comuns é proteger os serviços ou APIs por meio de funcionalidades de autenticação e autorização. É aí que entra a **KEYCLOAK**.Keycloak é uma solução de gerenciamento de acesso e identidade de código aberto que facilita a segurança de aplicações ou microsserviços com pouco ou nenhum código. Ela oferece suporte para logon único, login social e protocolos padrão, como OpenID Connect, OAuth2 e SAML. Nossos times vêm usando essa ferramenta e planejam continuar usando-a no futuro próximo. Entretanto ela exige um pouco de trabalho para configurar. Em razão da configuração ocorrer tanto na inicialização quanto durante o tempo de execução por meio de APIs, é necessário escrever scripts para assegurar que as implantações sejam repetíveis.

Em edições anteriores, nós mencionamos que Unity se tornou a plataforma de escolha para desenvolvimento de aplicações de RV e RA, pois ela oferece as abstrações e as ferramentas de uma plataforma madura, ao mesmo tempo sendo mais acessível que sua principal alternativa, a Unreal Engine. Com as introduções recentes do ARKit para iOS e ARCore para Android, as duas principais plataformas móveis agora possuem SDKs nativos poderosos para construir aplicações de realidade aumentada. Contudo, sentimos que muitos times, especialmente os que não possuem experiência profunda na construção de jogos, se beneficiarão do uso de uma abstração como a Unity, que é o motivo de estarmos chamando a atenção para a **UNITY ALÉM DE JOGOS**. Isso permite que pessoas desenvolvedoras não familiarizadas com a tecnologia se concentrem em um SDK. Ele também oferece uma solução para o enorme número de dispositivos, principalmente Android, que não são suportados pelos SDKs nativos.

O **WECHAT**, frequentemente visto como equivalente ao WhatsApp, está se tornando a plataforma de negócios dominante na China. Muitas pessoas podem não saber, mas WeChat também é uma das mais populares plataformas de pagamento online. Com CMS e gestão de membros embutidos no aplicativo, pequenas empresas estão realizando seus negócios integralmente no WeChat. Por meio da funcionalidade Conta de Serviço, grandes organizações podem gerar uma interface de seu sistema interno para funcionários. Dado que mais de 70% da população chinesa está usando o WeChat, é uma consideração importante para as empresas que querem expandir para o mercado chinês.

A **AZURE SERVICE FABRIC** é uma plataforma de sistemas distribuídos construída para microsserviços e contêineres. Ela é comparável a orquestradores de contêiner como Kubernetes, mas também trabalha com serviços antigos. Ela pode ser usada em uma variedade desconcertante de formas, de serviços simples em sua linguagem de escolha a contêineres Docker ou serviços construídos usando um SDK. Desde seu lançamento alguns anos atrás, ela tem adicionado constantemente mais recursos, incluindo suporte a contêineres Linux. Kubernetes tem sido o exemplo proeminente de ferramenta de orquestração de contêiner, mas Service Fabric é a escolha padrão para aplicações .NET. Estamos a usando em alguns projetos na ThoughtWorks e gostamos do que vimos até agora.

CLOUD SPANNER é um serviço de banco de dados relacional completamente gerenciado que oferece alta disponibilidade e grande consistência sem comprometer a latência. O Google tem trabalhado em um banco de dados distribuído globalmente chamado Spanner já há algum um tempo. O serviço foi lançado recentemente para o mundo externo como Cloud Spanner. É possível escalar sua instância de banco de dados de um a milhares de nós em todo o mundo sem se preocupar com a consistência dos dados. Ao alavancar o TrueTime, um relógio distribuído e altamente disponível, o Cloud Spanner oferece grande consistência para leituras e snapshots. Você pode usar SQL padrão para ler dados do Cloud Spanner, mas para escrever operações você precisa usar o API RPC deles. Embora nem todos os serviços exijam um banco de dados distribuído em escala global, a disponibilidade geral de Cloud Spanner é uma grande mudança no modo que pensamos os bancos de dados. E seu design está influenciando produtos de código aberto como o CockroachDB.

Após uma exploração minuciosa, a R3, uma empresa importante no espaço de blockchain, percebeu que blockchain não se encaixava bem em seus objetivos, e então criou a **CORDA**. A Corda é uma plataforma de tecnologia de contabilidade distribuída (DLT) focada na área financeira. A R3 tem uma proposta de valor muito clara e sabe que seu problema requer uma abordagem de tecnologia pragmática. Isso corresponde à nossa própria experiência. As soluções de blockchain atuais podem não ser a escolha mais sensata para alguns casos de negócio, devido aos custos de mineração e à ineficiência operacional. Embora a experiência de desenvolvimento que temos com Corda até agora não tenha sido a mais suave — APIs ainda estão instáveis após a versão v1.0 — esperamos ver o espaço DLT amadurecer ainda mais.

COSMOS DB é o serviço de banco de dados multimodelo distribuído globalmente pela Microsoft, que ficou disponível para o público geral no início deste ano. Enquanto a maioria dos bancos de dados NoSQL modernos oferecem consistência ajustável, o Cosmos DB faz dela uma cidadã de primeira classe e oferece cinco modelos de consistência diferentes. Vale ressaltar que também suporta vários modelos — valor-chave, documento, família de colunas e gráfico — todos os quais mapeiam seu modelo de dados internos, chamado de atom-record-sequence (ARS). Um aspecto interessante do Cosmos DB é que ele oferece acordos de nível de serviço (SLAs) em sua latência, taxa de transferência, consistência e disponibilidade. Com sua ampla gama de aplicabilidade, estabeleceu um padrão alto para outros fornecedores de nuvem igualarem.

Paralelamente ao aumento recente de chatbots e plataformas de voz, nós observamos uma proliferação de ferramentas e plataformas que oferecem um serviço para extrair intenção de texto e gestão de fluxos conversacionais aos quais você pode se conectar. O **DIALOGFLOW** (anteriormente API.ai), que foi adquirido pela Google, é um tipo de oferta de "compreensão de linguagem natural como serviço" que compete com wit.ai e Amazon Lex, entre outros players nesse espaço.

Embora o ecossistema de desenvolvimento de software esteja convergindo para Kubernetes como plataforma de orquestração para contêineres, a execução de clusters Kubernetes permanece operacionalmente complexa. **GKE** (Google Container Engine) é uma solução Kubernetes gerenciada para implantação de aplicações em contêineres que alivia a sobrecarga operacional de execução e manutenção de clusters Kubernetes. Nossos times tiveram boas experiências usando o GKE, com a plataforma fazendo o trabalho pesado de aplicação de patches de segurança, monitoramento e reparação automática dos nós, e gerenciando redes multicluster e multiregionais. Na nossa experiência, a primeira abordagem API-first do Google exibindo as capacidades da plataforma, além de usar padrões da indústria como OAuth para autorização de serviço, melhora a experiência de desenvolvimento. É importante considerar que o GKE está em um ritmo de desenvolvimento rápido que, apesar dos melhores esforços da equipe de desenvolvimento para abstrair as pessoas consumidoras das mudanças subjacentes, nos impactou temporariamente no passado. Esperamos uma melhoria contínua na maturidade da Infraestrutura como código com Terraform no GKE e ferramentas similares.

KAFKA STREAMS é uma biblioteca leve para a construção de aplicações de streaming. Ela foi projetada com o objetivo de simplificar o processamento de

streams o suficiente para torná-lo facilmente acessível por meio de um modelo de programação de aplicações comum para serviços assíncronos. Pode ser uma boa alternativa para cenários em que você deseja aplicar um modelo de processamento de stream a seu problema sem incluir a complexidade de executar um cluster (geralmente introduzido por frameworks completos de processamento de stream). Novos desenvolvimentos incluem processamento de stream "exatamente uma vez" em um cluster Kafka. Isso foi possível introduzindo idempotência em produtores Kafka e permitindo escritas atômicas em múltiplas partições usando a nova Transactions API.

Servidores de linguagem usam funcionalidades como autopreenchimento, identificadores de chamada e refatoração em uma API que permite que qualquer editor de texto trabalhe com a árvore sintática abstrata da linguagem

(Language Server Protocol)

Muito do poder de IDEs sofisticados vem de sua capacidade de analisar um programa em uma árvore sintática abstrata (AST) e então usar essa AST para análise e manipulação do programa. Isso dá suporte a recursos como autopreenchimento, identificadores de chamada e refatoração. Servidores de linguagem usam essa capacidade em um processo que permite que qualquer editor de texto acesse uma API para trabalhar com a AST. A Microsoft liderou a criação do **LANGUAGE SERVER PROTOCOL** (LSP), extraído a partir de seus projetos OmniSharp e TypeScript Server. Qualquer editor que usa esse protocolo consegue trabalhar com qualquer linguagem que possui um servidor compatível com LSP. Isso significa que nós conseguimos continuar usando nossos editores favoritos sem abdicar de modos ricos de edição de muitas linguagens — o que muito agrada as pessoas apreciadoras do Emacs.

LORAWAN é uma rede de área ampla de baixa potência, projetada para baixo consumo de energia e comunicação de longa distância usando baixas taxas de transmição. Fornece comunicação entre dispositivos e gateways, podendo encaminhar os dados para, por exemplo, aplicações ou servidores. Um uso típico é para um conjunto distribuído de sensores, ou dispositivos de Internet de Coisas (IoT), para os quais longa duração da bateria e comunicação de longa distância são obrigatórias. O LoRaWAN visa dois dos principais problemas quando se tenta usar Wi-Fi normal para tais aplicações: alcance e consumo de energia. Existem várias implementações, tendo destaque a The Things Network, uma implementação gratuita de código aberto.

MAPD é um banco de dados analítico colunar em memória com suporte SQL, construído para rodar em GPU. Nós discutimos se a carga de trabalho do banco de dados é de fato E/S ou vinculada computacionalmente, mas há instâncias em que o paralelismo da GPU, combinado com a grande largura de banda de VRAM, pode ser bastante útil. MapD gerencia de forma transparente os dados mais frequentemente usados em VRAM (como colunas envolvidas em condições de agrupar-por, filtros, cálculos e associação) e armazena o restante dos dados na memória principal. Com essa configuração de gerenciamento de memória, o MapD atinge um desempenho de consulta significativo sem a necessidade de indexação. Embora haja outros fornecedores de bancos de dados para GPU, MapD está liderando este segmento com a recente versão código aberto de seu banco de dados principal e por meio da GPU Open Analytics Initiative. Se sua carga de trabalho analítico é computacionalmente pesada, consegue explorar o paralelismo da GPU e consegue se encaixar na memória principal, recomendamos avaliar o MapD.

Nós gostamos de ferramentas simples que resolvem um problema muito bem, e a **NETLIFY** se encaixa muito bem nessa descrição. Você consegue criar conteúdo de website estático, adicioná-lo ao GitHub e, então, de forma rápida e fácil, colocar seu site no ar. Há uma CLI disponível para controlar o processo; redes de distribuição de conteúdo (CDNs) são suportadas; ela pode trabalhar em conjunto com ferramentas como Grunt; e, o mais importante, a Netlify suporta HTTPS.

Modelos de aprendizagem de máquina estão começando a se infiltrar nas aplicações comerciais comuns. Quando dados de treinamento suficientes estão disponíveis, esses algoritmos conseguem lidar com problemas que possivelmente requeriam anteriormente modelos estatísticos complexos ou heurísticas. Conforme nos movemos do uso experimental para produção, precisamos de uma maneira confiável de

hospedar e implantar os modelos que podem ser acessados remotamente e escalados com o número de clientes. **TENSORFLOW SERVING** lida com parte desse problema ao expor uma interface remota de gRPC a um modelo exportado; isso permite que um modelo treinado seja implantado em uma variedade de maneiras. TensorFlow Serving também aceita um stream de modelos para incorporar atualizações de treinamento contínuas, e seus autores mantêm um Dockerfile para facilitar o processo de implantação. Presumivelmente, a escolha de gRPC deve ser consistente com o modelo de execução TensorFlow, no entanto, adotamos cautela de modo geral com protocolos que exigem geração de código e associações nativas.

A Microsoft está se recuperando no espaço de contêiner com os **CONTÊINERES DO WINDOWS**. No momento da redação, a Microsoft fornece duas imagens do SO Windows como contêineres Docker, Windows Server 2016 Server Core e Windows Server 2016 Nano Server. Embora exista margem para melhoria dos Contêineres do Windows, por exemplo, diminuindo os grandes tamanhos de imagem e enriquecendo o suporte e a documentação do ecossistema, nossos times começaram a usá-los em cenários onde outros contêineres têm tido sucesso, por exemplo agentes de compilação.

Conforme nos movemos do uso experimental para produção, precisamos de uma maneira confiável de hospedar e implantar os modelos que podem ser acessados remotamente e escalados com o número de clientes.

(TensorFlow Serving)

Continua nos preocupando implementar lógica do negócio e orquestração de processos no middleware, especialmente onde se requer expertise em habilidades e ferramentas ao criar pontos únicos de escala e controle. Fornecedores no competitivo mercado de API Gateways estão alimentando essa tendência ao adicionar funcionalidades na tentativa de diferenciar seus produtos. Isso resulta em **API GATEWAYS EXCESSIVAMENTE AMBICIOSOS**, cuja funcionalidade — com base no que é essencialmente um proxy reverso — encoraja projetos que continuem sendo difíceis de testar e implantar. API gateways são sim úteis para lidar com algumas preocupações específicas — como autenticação e limitação de taxas de utilização —, mas todas as inteligências de domínio devem estar em aplicações ou serviços.

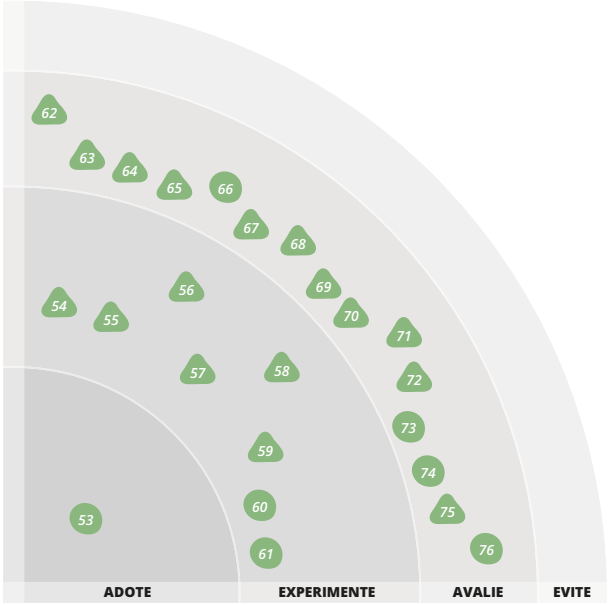
FERRAMENTAS

ADOTE
53. fastlane

EXPERIMENTE
54. Buildkite *NOVO*
55. CircleCI *NOVO*
56. gopass *NOVO*
57. Headless Chrome para testes de front-end *NOVO*
58. jsoniter *NOVO*
59. Prometheus
60. Scikit-learn
61. Serverless Framework

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

EVITE



Nossas equipes gostam muito da ferramenta de CI/CD hospedado **BUILDKITE** por sua simplicidade e rápida configuração. Com Buildkite, você fornece suas próprias máquinas a execução de builds — localmente ou na nuvem — e instala um aplicação de agente leve para conectar o agente de compilação ao serviço hospedado. Em muitos casos, ter este nível de controle sobre a configuração de seus agentes de compilação é um ponto positivo quando comparado com o uso de agentes hospedados.

CIRCLECI é um mecanismo de integração contínua oferecido como SaaS e hospedado localmente. O CircleCI tem sido a ferramenta SaaS CI de escolha para muitos de nossos times de desenvolvimento que precisam de um pepeline de compilação e implantação de baixa fricção e fácil de configurar. A CircleCI versão 2.0 suporta fluxos de trabalho de compilação, com fluxos de entrada e saída e portas manuais, bem como desenvolvimento para dispositivos móveis. Permite que as pessoas desenvolvedoras executem os pipelines localmente e se integra facilmente com Slack e outros sistemas de notificação e alerta. Recomendamos que você examine mais de perto as práticas de segurança da CircleCI, assim como você faria com qualquer outro produto SaaS que hospede os recursos da sua empresa.

Descendente do pass, gopass adiciona funcionalidades como suporte para gestão de receptor e armazenamento de múltiplas senhas em uma única árvore; uma funcionalidade de busca interativa; suporte a senha única baseada em tempo (TOTP); e armazenamento de dados binários.
(gopass)

GOPASS é uma solução de gerenciamento de senhas para times, construída com GPG e Git. Ela é descendente do pass e adiciona funcionalidades como suporte para gestão de receptor e armazenamento

de múltiplas senhas em uma única árvore; uma funcionalidade de busca interativa; suporte a senha única baseada em tempo (TOTP); e armazenamento de dados binários. A migração de seus dados do pass é bastante direta, pois gopass é amplamente compatível com o formato usado por pass. Isso também significa que uma integração em provisionamento de fluxos de trabalho pode ser alcançada com uma única chamada a um segredo armazenado.

Desde meados de 2017, usuários do Chrome tiveram a opção de executar o navegador no modo headless. Esse recurso é ideal para executar testes de navegador front-end sem a sobrecarga de exibir as ações em uma tela. Anteriormente, esse território foi amplamente dominado pelo PhantomJS, mas o Headless Chrome está substituindo rapidamente a abordagem de WebKit guiado por JavaScript. Testes no Headless Chrome podem ser executados com muito mais rapidez e se comportam mais como em um navegador real, mas nossos times observaram que ele, de fato, usa mais memória que o PhantomJS. Com todas essas vantagens, é provável que o **HEADLESS CHROME PARA TESTES DE FRONT-END** se torne efetivamente o padrão.

ISe você está procurando um codificador/decodificador JSON com alto desempenho em Go e Java, confira a biblioteca de código aberto **JSONITER**. A biblioteca é compatível com o pacote de codificação JSON padrão em Go.

Temos observado melhoria contínua e aumento na adoção de Prometheus, a ferramenta de monitoramento e banco de dados de séries temporais originalmente desenvolvida pelo Soundcloud.

(Prometheus)

Temos observado melhoria contínua e aumento na adoção de **PROMETHEUS**, a ferramenta de monitoramento e banco de dados de séries temporais originalmente desenvolvida pelo Soundcloud. O Prometheus suporta essencialmente um modelo HTTP baseado em pull, mas também suporta alertas, tornando-se uma parte ativa do seu conjunto de ferramentas operacionais. No momento desta redação, o Prometheus 2.0 está em fase de pré-lançamento e continua a evoluir. O time de desenvolvimento do Prometheus concentrou seus esforços em bancos de dados de séries temporais fundamentais e na variedade de métricas disponíveis. Grafana tornou-se a principal escolha como ferramenta de visualização de painel para usuários do Prometheus, e o suporte para Grafana vem junto com a ferramenta. Nossos times também consideram que o monitoramento do Prometheus complementa bem as capacidades de indexação e pesquisa de uma Elastic Stack.

APEX é a ferramenta para construir, implementar e gerenciar funções de AWS Lambda com facilidade. Com Apex, você pode escrever funções em linguagens que não são ainda suportadas nativamente em AWS, incluindo Golang, Rust e outras. Isso é possibilitado por um Node.js shim, que cria um processo inicial e processa eventos por meio de stdin e stdout. Apex tem muitas funcionalidades interessantes que melhoram a experiência de desenvolvimento, e nós particularmente gostamos da capacidade de testar funções localmente e realizar um exercício de simulação das mudanças antes destas serem aplicadas aos recursos AWS.

A biblioteca AssertJ, **ASSERTJ-SWAGGER** permite validar a conformidade de uma implementação da API com as especificações do contrato. Nossos times usam assertj-swagger para detectar problemas quando a implementação do endpoint da API muda sem atualizar sua especificação Swagger ou falha em publicar a documentação atualizada.

Consertar falhas end-to-end em CI pode ser uma experiência dolorosa, especialmente no modo headless. **CYPRESS** é uma ferramenta útil que ajuda pessoas desenvolvedoras a construir testes end-to-end com facilidade e registra todas as etapas de teste no formato de vídeo em um arquivo MP4. Em vez de reproduzir o problema em modo headless, as pessoas que estão desenvolvendo podem assistir ao vídeo do teste para consertá-lo. Cypress é uma plataforma poderosa, e não somente um framework de teste. Atualmente, nós integramos sua CLI com CI headless em nossos projetos.

FLOW é um verificador de tipos estáticos para JavaScript, que permite adicionar verificação de tipos na base de código de maneira incremental. Ao contrário do TypeScript, que é uma linguagem diferente, Flow

pode ser adicionado incrementalmente a uma base de código JavaScript existente, suportando as versões 5, 6 e 7 do ECMAScript. Recomendamos adicionar o Flow na sua pipeline de integração contínua, começando pelo código que mais lhe preocupa. Flow adiciona clareza ao código, aumenta a confiabilidade de refatoração e encontra problemas relacionados a tipos durante a compilação.

Ao longo dos últimos anos, nós notamos um aumento constante na popularidade de notebooks analíticos, que são aplicações inspiradas no Mathematica, combinando texto, visualização e código em um documento computacional vivo. Em uma edição anterior, nós mencionamos o GorillaREPL, que é uma variante em Clojure destas aplicações. Porém, o crescimento do interesse em aprendizado de máquina — junto com o surgimento do Python como a linguagem de escolha para profissionais da área — voltou às atenções particularmente para notebooks Python, dos quais **JUPYTER** parece estar ganhando mais tração entre os times da ThoughtWorks.

Kong é um API gateway de código aberto desenvolvido e patrocinado pela Mashape, que também possui uma oferta empresarial que integra Kong com suas ferramentas proprietárias de API analytics e portal de desenvolvedores. Eles podem ser implantados com uma variedade de configurações, como um API gateway de ponta ou um proxy de API interno. O OpenResty, por meio de seus módulos Nginx, fornece uma base sólida e com boa performance, com plugins Lua para extensões. O Kong pode usar PostgreSQL para implantações de região única ou Cassandra para configuração multiregião. Nossas pessoas desenvolvedoras gostaram do alto desempenho do Kong, sua abordagem API-primeiro (que permite a automação de sua configuração) e sua facilidade de implantação como contêiner. **KONG API GATEWAY**, ao contrário de API gateways excessivamente abiciosos, tem um conjunto menor de funcionalidades, mas implementa o conjunto essencial de recursos de API gateway, como controle de tráfego, segurança, registro, monitoramento e autenticação. Esperamos ansiosamente avaliar o Kong em uma configuração sidecar em um futuro próximo.

KOPS é uma ferramenta de linha de comando para criar e gerenciar a produção de alta disponibilidade de clusters Kubernetes. Inicialmente visando AWS, possui agora suporte experimental para outros provedores. Ela é capaz de colocar tudo para funcionar com rapidez, e apesar de algumas funcionalidades (como lançar atualizações) ainda precisarem ser desenvolvidas por completo, a comunidade tem nos impressionado.

Um problema perene para aplicações na web e pesadas em JavaScript é como tornar a porção dinâmica dessas páginas disponível para mecanismos de busca. Bots que não renderizam JavaScript podem ser roteados para esse servidor para fazer a renderização por eles.

(Rendertron)

LIGHTHOUSE é uma ferramenta escrita pelo Google para avaliar as aplicações web para aderência aos padrões Progressive Web App. O lançamento do Lighthouse 2.0 neste ano adiciona métricas de desempenho e verificações de acessibilidade ao conjunto básico de ferramentas. Essa funcionalidade adicional já foi incorporada nas ferramentas padrão de desenvolvimento do Chrome na aba de auditoria. O Lighthouse 2.0 é mais uma ferramenta que se beneficia do modo headless do Chrome. É uma alternativa para Pa11y e ferramentas similares para executar verificações de acessibilidade em um pipeline de integração contínua, uma vez que a ferramenta pode ser executada a partir da linha de comando ou de forma autônoma como uma aplicação Node.js.

Um problema perene para aplicações na web e pesadas em JavaScript é como tornar a porção dinâmica dessas páginas disponível para mecanismos de busca. Historicamente, pessoas desenvolvedoras recorrem a uma variedade de truques, incluindo renderizar pelo lado do servidor com React, serviços externos ou prerrenderizar conteúdos. Agora, o novo modo headless do Google Chrome adiciona um novo "truque" à caixa de ferramentas — **RENDERTRON**, uma solução de renderização do headless do Chrome. O Rendertron

envolve uma instância headless do Chrome em um contêiner Docker, pronto para ser implementado como um servidor HTTP independente. Bots que não renderizam JavaScript podem ser roteados para esse servidor para fazer a renderização por eles. Embora pessoas desenvolvedoras sempre possam implementar seu próprio proxy headless do Chrome e maquinário de roteamento associado, o Rendertron simplifica o processo de configuração e implantação, e oferece código de middleware exemplo para detectar e rotear bots.

SONOBUOY é uma ferramenta de diagnóstico para executar testes de conformidade de ponta a ponta em qualquer cluster Kubernetes de forma não destrutiva. O time da Heptio, fundada por duas das pessoas que criaram os projetos Kubernetes, construiu essa ferramenta para garantir que a ampla gama de distribuições e configurações do Kubernetes estejam em conformidade com as melhores práticas, ao mesmo tempo seguindo a padronização de código aberto para interoperabilidade de clusters. Estamos testando a execução de Sonobuoy como parte do nosso pipeline de compilação de infraestrutura como código, bem como o monitoramento contínuo de nossas instalações Kubernetes, para validar o comportamento e a integridade de todo o cluster.

Se você estiver implementando serviços Java usando o framework Spring, você pode querer considerar usar **SPRING CLOUD CONTRACT** para testes de contrato guiados pelo consumidor. O ecossistema atual desta ferramenta suporta a verificação das chamadas de cliente e a implementação do servidor contra o contrato. Em comparação com Pact, um conjunto de código aberto de ferramentas de testes de contrato guiados pelo consumidor, falta a negociação dos contratos e suporte para outras linguagens de programação. No entanto, se integra bem com o ecossistema Spring, por exemplo no roteamento de mensagens com Spring Integration.

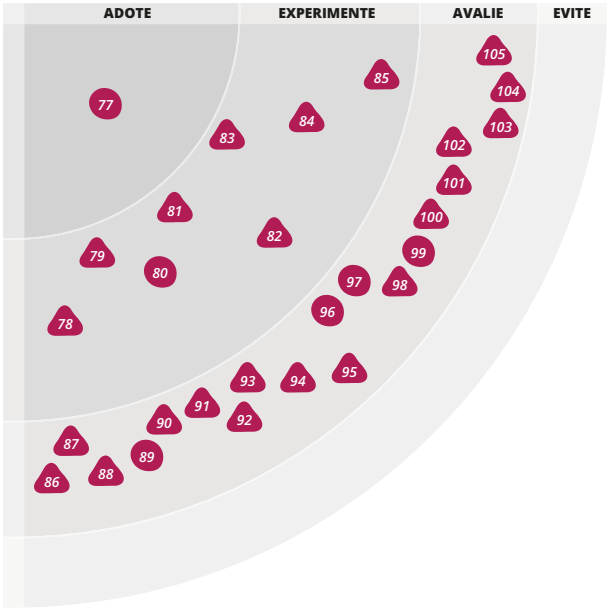
LINGUAGENS E FRAMEWORKS

ADOTE
77. Python 3

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

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

EVITE



Em edições anteriores do Radar, hesitamos em dar uma recomendação sólida a **ANGULAR**, porque era essencialmente um framework novo e, de forma geral, não muito animador, compartilhando apenas seu nome com o AngularJS, um antigo framework que gostamos muito no passado. Enquanto isso, o Angular, agora na versão 5, melhorou de forma consistente, ao mesmo tempo oferecendo compatibilidade com versões anteriores. Vários de nossos times têm aplicações Angular em produção e, segundo relatos têm gostado do que vêem. Por esse motivo, estamos movendo Angular para o anel Avalie neste Radar, refletindo o fato de alguns de nossos times considerarem uma escolha sólida agora. A maioria de nossos times, no entanto, ainda prefere React, Vue ou Ember em relação a Angular.

ASSERTJ é uma biblioteca Java que oferece uma interface fluente para asserções, as quais tornam fácil transmitir intenção dentro do código de teste. AssertJ fornece mensagens de erro legíveis, asserções flexíveis, e suporte aperfeiçoado de grupos e exceções. Nós estamos vendo alguns times padronizarem seu uso em vez de JUnit combinado com Hamcrest.

O CSS Grid Layout é um sistema de layout baseado em grade bidimensional que fornece um mecanismo para dividir o espaço disponível para layout em colunas e linhas usando um conjunto de comportamentos de dimensionamento previsível.

(CSS Grid Layout)

CSS é a escolha preferida para layout de páginas da web, mesmo quando não fornecia suporte explícito para a criação de layouts. Flexbox ajudava com layouts mais simples e unidimensionais, mas as pessoas desenvolvedoras de forma geral buscavam bibliotecas e conjuntos de ferramentas para layouts mais complexos.

O **CSS GRID LAYOUT** é um sistema de layout baseado em grade bidimensional que fornece um mecanismo para dividir o espaço disponível para layout em colunas e linhas usando um conjunto de comportamentos de dimensionamento previsível. O Grid não requer quaisquer bibliotecas e roda bem com Flexbox e outros elementos de visualização de CSS. Entretanto, uma vez que o IE11 é somente parcialmente suportado, ele ignora usuários que ainda dependem de um navegador da Microsoft no Windows 7.

A maioria dos grandes bases de código CSS requer esquemas de nomes complexos para ajudar a evitar conflitos de nomes no namespace global. O **CSS MODULES** aborda esses problemas ao criar um escopo local para todos os nomes de classe em um único arquivo CSS. Esse arquivo é importado para um módulo JavaScript, em que as classes CSS são referenciadas como strings. Então, no pipeline de compilação (Webpack, Browserify, etc.), os nomes de classe são substituídos com sequências únicas geradas. Essa é uma alteração significativa nas responsabilidades. Anteriormente, um humano tinha que gerenciar o namespace global, para evitar conflitos de nomenclatura de classe; agora, essa responsabilidade recai sobre a ferramenta de compilação. Um pequeno ponto negativo que encontramos com o CSS Modules: testes funcionais estão geralmente fora do escopo local e podem, portanto, não referenciar classes pelo nome definido no arquivo CSS. Nós recomendamos usar IDs ou dados atribuídos em seu lugar.

Jest é uma ferramenta de testes de front-end com "configuração zero" e recursos inovadores, como mocking e cobertura de código, focada em React e outros frameworks JavaScript.

(Jest)

Nossos times estão encantados com os resultados do uso de **JEST** para testes front-end. Ele oferece experiência de "configuração zero" e possui recursos inovadores, como mocking e cobertura de código. Você pode aplicar esse framework de testes não apenas a aplicações React, mas também a outros frameworks JavaScript. Um dos recursos mais exaltados do Jest é o teste snapshot de IU. O teste snapshot seria uma boa adição à camada superior da pirâmide de testes, mas lembre-se, o teste de unidade ainda é fundamental.

O anúncio do suporte de primeira classe no Android deu um impulso extra para o rápido progresso da linguagem **KOTLIN**, e estamos acompanhando o progresso de Kotlin/Native atentamente — a capacidade suportada por LLVM para compilar executáveis nativos. Segurança no uso de nulls, classes de dados e facilidade de criar DSLs são alguns dos benefícios que desfrutamos, junto com a biblioteca Anko para desenvolvimento em Android. Apesar dos pontos negativos de compilação inicial demorada e dependência do IntelliJ para suporte de IDE de primeira classe, recomendamos testar essa nova e concisa linguagem moderna.

A **SPRING CLOUD** continua a evoluir e a adicionar novas funcionalidades interessantes. O suporte para ligação com Kafka Streams, por exemplo, no projeto spring-cloud-streams torna relativamente fácil criar aplicações orientadas por mensagem com conectores para Kafka e RabbitMQ. Nossos times a usando apreciam a simplicidade que ela traz ao uso de infraestruturas às vezes complexa, como ZooKeeper, e suporte para problemas comuns que temos que lidar ao criar sistemas distribuídos, rastreando com spring-cloud-sleuth, por exemplo. As ressalvas habituais se aplicam, mas estamos usando com êxito em vários projetos.

Historicamente, os exemplos de documentação Android do Google careciam de arquitetura e estrutura. Isso mudou com o lançamento dos **ANDROID ARCHITECTURE COMPONENTS**, um conjunto de bibliotecas que ajudam pessoas desenvolvedoras a criar aplicativos Android com melhor arquitetura. Eles lidam com pontos problemáticos antigos no desenvolvimento em Android: manipulação de ciclos de vida; paginação; bancos de dados SQLite; e persistência de dados sob alterações de configuração. As bibliotecas não precisam ser usadas em conjunto — você pode escolher aquelas que mais precisa e integrá-las em seu projeto.

Observamos uma gigantesca atividade em realidade aumentada móvel, em grande parte fomentada por **ARKIT E ARCORE**, as bibliotecas nativas de RA usadas

por Apple e Google, respectivamente. Essas bibliotecas estão trazendo as tecnologias RA móveis para o mercado. No entanto, o desafio para as empresas será encontrar casos de uso que vão além de recursos chamativos e forneçam soluções originais que, de fato, melhorem a experiência do usuário.

Android Architecture Components are a set of opinionated libraries that help developers create Android applications with better architecture.

(Android Architecture Components)

Uma estratégia multiaplicativo é realmente controversa, particularmente em um momento em que cada vez menos usuários estão baixando novos aplicativos. Em vez de apresentar um novo aplicativo e ter dificuldades com o número de downloads, os vários times têm que entregar funcionalidade por meio de um único aplicativo já amplamente instalado, o que cria um desafio arquitetural. **ATLAS E BEEHIVE** são soluções de modularização para aplicativos Android e iOS, respectivamente. Atlas e BeeHive permitem que times trabalhem em módulos fisicamente isolados para remontar ou carregar dinamicamente esses módulos a partir de um aplicativo de fachada. Ambos são projetos de código aberto da Alibaba, já que a Alibaba encontrou o mesmo problema de redução de downloads e desafios arquiteturais de aplicativos únicos.

Nosso primeiro princípio básico ao selecionar um mecanismo de regras normalmente é: você não precisa de um mecanismo de regras. Nós vimos muitas pessoas se amarrarem a um mecanismo de regras de caixa preta difícil de testar sem ter um bom motivo, quando um código personalizado teria sido uma solução melhor. Dito isto, tivemos sucesso usando **CLARA RULES** para cenários em que um mecanismo de regras faz sentido. Nós gostamos do fato de ele usar um simples código Clojure para expressar e avaliar as regras, o que significa que elas são acessíveis para controle de versão, teste e refatoração. Em vez de correr atrás da ilusão de que a área de negócios deveria manipular diretamente as regras, ele conduz a colaboração entre especialistas em negócio e pessoas desenvolvedoras.

CSS EM JS é uma técnica de escrever estilos CSS na linguagem de programação JavaScript. Isso encoraja um padrão comum de escrever o estilo com o componente JavaScript ao qual se aplica, colocalizando preocupações de apresentação e lógica. Os novos players — incluindo JSS, Emotion e styled-components — confiam na ferramenta para traduzir o código CSS em JS para separar stylesheets CSS, tornando-as adequadas para o consumo do navegador. Essa é a abordagem de segunda geração para escrever CSS em JavaScript e, ao contrário das abordagens anteriores, não depende dos estilos em linha. Isso significa que ela oferece o benefício de suportar todos as funcionalidades CSS, compartilhamento de CSS usando o ecossistema npm e utilização de componentes em várias plataformas. Nossos times consideraram que o styled-components trabalha bem com frameworks baseados em componentes, como React e teste unitário de CSS com jest-styled-components. Esse espaço é novo e está mudando rapidamente; a abordagem requer algum esforço para a depuração manual dos nomes das classes geradas no navegador e pode não se aplicar a alguns projetos onde a arquitetura front-end não suporta reutilização de componentes e requer estilo global.

DIGDAG é uma ferramenta para construir, executar, agendar e monitorar pipelines de dados complexos na nuvem. Você pode definir esses pipelines em YAML, usando o rico conjunto de operadores padrão ou construir seu próprio conjunto por meio de API. O Digdag possui a maioria das funcionalidades comuns em uma solução de pipeline de dados, como gestão de dependência, fluxo de trabalho modular para promover o reuso, gestão de segredos protegidos e suporte multi-idiomas. A funcionalidade que mais nos animou foi o suporte polycloud, que permite que você mova e junte dados entre AWS RedShift, S3, e Google BigQuery. Como cada vez mais provedores de nuvem oferecem soluções de processamento de dados competitivas, nós achamos que o Digdag (e ferramentas semelhantes) serão úteis para alavancar a melhor opção para a tarefa.

DRUID é um pool de conexões JDBC com diversas funcionalidades de monitoramento. Ele possui um analisador de SQL embutido, que oferece monitoramento semântico das instruções de SQL em execução no banco de dados. Injeções ou instruções

de SQL suspeitas serão bloqueadas e registradas diretamente a partir da camada JDBC. Além disso, as consultas podem ser incorporadas com base em sua semântica. Esse é um projeto de código aberto da Alibaba, e reflete as lições aprendidas da Alibaba pela operação de seus próprios sistemas de banco de dados.

ECHARTS é uma biblioteca leve de gráficos com vasto suporte para diferentes tipos de gráficos e interações. Pelo fato de ser totalmente baseada em Canvas API, ela possui um desempenho incrível mesmo quando lida com mais de 100 mil pontos de dados, e também é otimizada para uso em dispositivos móveis. Em conjunto com seu projeto-irmão, ECharts-X, ela consegue suportar plotagem 3D. ECharts é um projeto de código aberto da Baidu.

A capacidade de compilar a linguagem de programação Go diretamente para hardware despertou o interesse entre as pessoas desenvolvedoras em usar a linguagem para sistemas embarcados. **GOBOT** é um framework para robótica, computação física e Internet das Coisas, escrito na linguagem de programação Go e oferecendo suporte para uma variedade de plataformas. Temos utilizado o framework para projetos experimentais de robótica em que a resposta em tempo real não é um requisito e criamos drivers de software de código aberto com Gobot. A API HTTP de Gobot permite a integração de hardware simples com dispositivos móveis para criar aplicações mais ricas.

Nossos times de dispositivos móveis ficaram entusiasmados com o **LEAKCANARY**, uma ferramenta para detectar vazamentos de memória desagradáveis em Android e Java. É simples de conectar e oferece notificações com identificações claras da causa do vazamento. Adicioná-lo a seu conjunto de ferramentas pode te poupar horas entediantes de resolução de erros de falta de memória em diversos dispositivos.

PYTORCH é uma reformulação completa do framework de aprendizagem de máquina Torch de Lua para Python. Embora seja bastante novo e imaturo em comparação com o Tensorflow, pessoas programadoras consideram o PyTorch muito mais fácil de se trabalhar. Por causa de sua orientação a objeto e sua implementação de Python nativo, os modelos podem ser expressos de forma mais clara e sucinta, e depurados durante a execução. Embora muitos desses frameworks tenham surgido recentemente, PyTorch tem o apoio do Facebook e uma ampla gama de organizações parceiras, incluindo a NVIDIA, o que deve assegurar o suporte contínuo para arquiteturas CUDA. Os times da ThoughtWorks acham o PyTorch útil para experimentar e desenvolver modelos, mas ainda confiam na performance do TensorFlow para treinamento e classificação em escala de produção.

SINGLE-SPA é um metaframework em JavaScript que nos permite construir micro frontends usando diferentes frameworks que podem coexistir em uma única aplicação. Em geral, nós não recomendamos usar mais que um framework para uma aplicação, mas há vezes em que não podemos evitar fazê-lo. Por exemplo, o single-spa pode ser bastante útil quando você está trabalhando com uma aplicação legada e quer experimentar desenvolver uma nova funcionalidade, seja com uma nova versão do framework existente ou com um completamente diferente. Devido à curta vida útil de muitos frameworks JavaScript, nós vemos a necessidade de uma solução que permita alterações futuras no framework e experimentação localizada, sem afetar toda a aplicação. single-spa parece ser um bom começo nessa direção.

Programar para contratos inteligentes requer uma linguagem mais expressiva que script de transação. **SOLIDITY** é a mais popular entre as novas linguagens de programação projetadas para contratos inteligentes. Solidity é uma linguagem orientada por contrato estaticamente tipificada, cuja sintaxe é similar a JavaScript. Ela fornece abstrações para escrever lógica empresarial de auto-aplicação em contratos inteligentes. A cadeia de ferramentas em torno de Solidity está crescendo rapidamente. Atualmente, Solidity é a primeira escolha na plataforma Ethereum.

TENSORFLOW MOBILE torna possível para as pessoas desenvolvedoras incorporar uma ampla gama de técnicas de compreensão e classificação em suas aplicações iOS ou Android. Isto é particularmente útil, dado o alcance dos dados de sensor disponíveis nos telefones celulares. Os modelos TensorFlow pré-treinados podem ser carregados em uma aplicação móvel e aplicados em entradas, como quadros de video ao vivo, texto ou fala. Os telefones celulares apresentam uma plataforma surpreendentemente oportuna para implementar esses modelos computacionais. Os modelos TensorFlow são

exportados e carregados como arquivos protobuf, que podem apresentar alguns problemas para implementadores. O formato binário do protobuf pode dificultar a análise de modelos e exige que você vincule a versão correta da biblioteca protobuf ao seu aplicativo móvel. Mas a execução do modelo local oferece uma alternativa atrativa para TensorFlow Serving sem a sobrecarga de comunicação da execução remota.

Tivemos sucesso usando Clara rules para cenários em que um mecanismo de regras faz sentido. Nós gostamos do fato de ele usar um simples código Clojure para expressar e avaliar as regras, o que significa que elas são acessíveis para controle de versão, teste e refatoração.

(Clara rules)

TRUFFLE é um framework de desenvolvimento que traz uma experiência moderna de desenvolvimento web para a plataforma Ethereum. Ele assume o trabalho de compilação de contratos inteligentes, conexão e implantação de bibliotecas, bem como o de lidar com artefatos em diferentes redes de blockchain. Um dos motivos pelos quais nós gostamos muito de Truffle é que ele incentiva pessoas a escrever testes para seus contratos inteligentes. É preciso levar os testes realmente a sério, já que a programação de contratos inteligentes é geralmente relacionada a dinheiro. Com seu framework de testes embutido e integração com TestRPC, Truffle possibilita escrever o contrato em uma forma TDD. Nós esperamos ver mais tecnologias semelhantes a Truffle para promover a integração contínua na área do blockchain.

WEEX é um framework para construir aplicativos móveis multiplataforma usando a sintaxe do componente Vue. Para quem prefere a simplicidade de Vue.js, Weex é uma opção viável para aplicativos móveis nativos, mas ela também funciona muito bem para aplicativoss mais complicados. Nós observamos muitos casos de sucesso para aplicativos móveis bastante complicados construídos nesse framework, incluindo TMall e Taobao, dois dos aplicativos móveis mais populares na China. Weex foi desenvolvido pela Alibaba, e é atualmente um projeto da incubadora Apache.

Saiba em primeira mão sobre o lançamento do próximo Technology Radar, e atualize-se com webinars e conteúdos exclusivos.

INSCREVA-SE

thght.works/Sub-PT

ThoughtWorks®

A ThoughtWorks é uma consultoria de tecnologia e uma comunidade de pessoas apaixonadas e guiadas por propósitos. Ajudamos nossos clientes a colocar a tecnologia no centro dos negócios e, de forma conjunta, criar o software que mais importa para eles. Dedicada à mudança social positiva, nossa missão é melhorar a humanidade através do software, e fazemos parcerias com muitas organizações que lutam na mesma direção.

Fundada há mais de 20 anos, a ThoughtWorks se tornou uma empresa com mais de 4500 pessoas, incluindo uma área de produtos que desenvolve ferramentas pioneiras para times de software. A ThoughtWorks tem 42 escritórios em 15 países: África do Sul, Alemanha, Austrália, Brasil, Canadá, Chile, China, Equador, Espanha, Estados Unidos, Índia, Itália, Reino Unido, Singapura e Turquia.

thoughtworks.com/pt