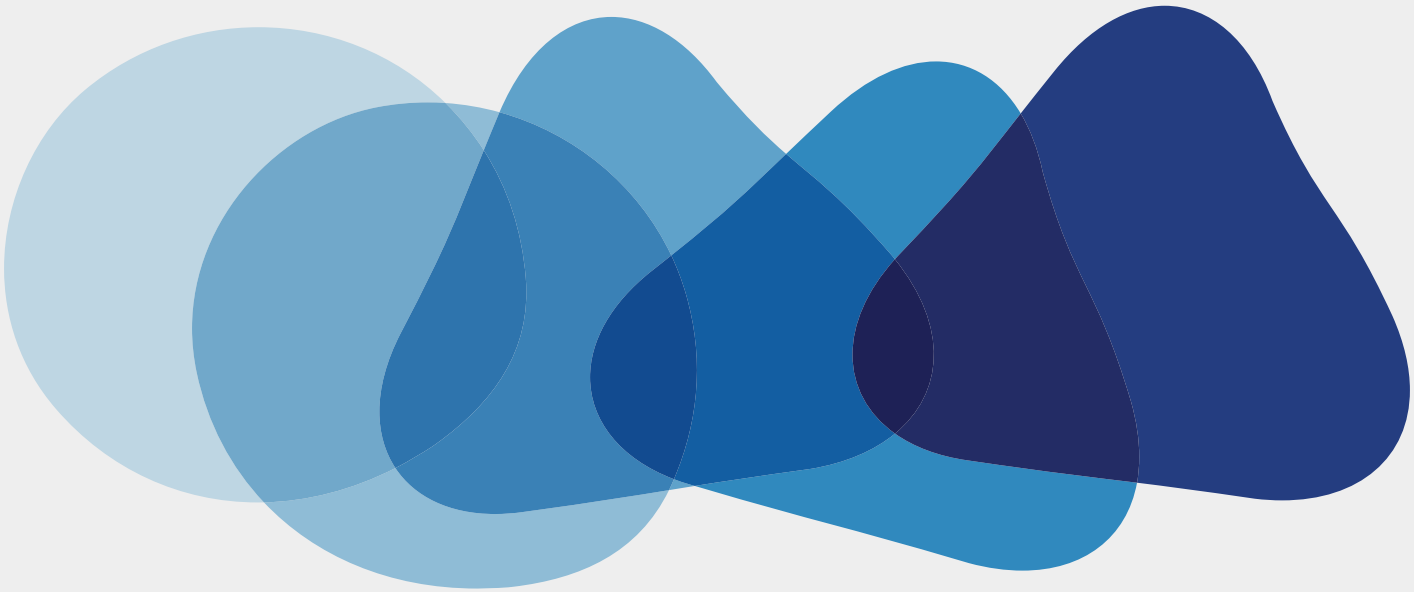
ThoughtWorks®

TECHNOLOGY RADAR *VOL.18*

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

Esta edição do Technology Radar da ThoughtWorks se baseou em uma reunião do Conselho Consultivo de Tecnologia, que se reuniu em Sydney em março de 2018.

Tradução: Glauco Vinicius, Luiz Felipe Dias, Marco Valtas, Marcos Brizeno, Paula Ribas e Ricardo Cavalcanti

O QUE HÁ DE NOVO?

Estes são os temas em destaque nessa edição:

NAVEGADORES MAIS ROBUSTOS, SERVIDORES MAIS LEVES

O navegador continua a expandir suas capacidades como destino de implantação para lógica de aplicação. À medida que as plataformas assumem mais preocupações transversais e requisitos não-funcionais, vemos uma tendência de redução da complexidade na lógica de back-end. A introdução de WebAssembly abre novas opções de linguagem para criar lógica para aplicações da Web e torna o processamento mais próximo do metal (e da GPU). A Web Bluetooth permite que os navegadores lidem com funcionalidades anteriormente reservadas para aplicações nativas, e cada vez mais vemos padrões abertos como CSS Grid Layout e CSS Modules suplantando bibliotecas personalizadas. A busca por melhores experiências de uso estimula a tendência de levar funcionalidade ao navegador e, como resultado, muitos serviços de back-end se tornam mais leves e menos complexos.

CRESCENTE COMPLEXIDADE DE NUVEM

Embora a AWS continue a avançar com uma gama vertiginosa de novos serviços, vemos cada vez mais Google Cloud Platform (GCP) e Microsoft Azure amadurecerem como alternativas viáveis. Camadas de abstração como Kubernetes e práticas como entrega contínua e infraestrutura como código facilitam a transição entre nuvens ao suportarem uma mudança evolutiva mais fácil. Mas as estratégias de nuvem tornam-se necessariamente mais complexas com o advento da Polycloud (que permite que as organizações selecionem vários provedores com base em recursos diferenciados) e com o aumento da regulamentação e das preocupações com privacidade. Muitos países da UE, por exemplo, agora exigem legalmente a localidade de dados, convertendo a jurisdição do armazenamento de dados e as políticas de hospedagem subjacentes em uma nova dimensão de diferenciação para quem avalia nuvens. A variedade de opções para ambientes de computação também está aumentando, com a AWS Fargate oferecendo contêineres como serviço (CaaS) como um meio-termo intrigante entre funções como serviço e gerenciamento de clusters de vida útil mais longa. Embora os recursos de nuvem continuem a amadurecer dentro das organizações, uma progressiva e inevitável complexidade sempre acompanha a construção de soluções reais com essas novas peças.

CONFIE NOS TIMES, MAS VERIFIQUE

A segurança continua sendo a preocupação primordial de virtualmente todo o desenvolvimento de software. Mas vemos uma mudança na abordagem tradicional de "bloquear tudo globalmente" para uma abordagem mais localizada e com mais nuances. Muitos sistemas agora gerenciam a confiança em domínios menores e usam mecanismos modernos para criar confiança transitiva entre sistemas diferentes. A filosofia mudou de "nunca confie em nada" fora do domínio e "nunca verifique nada" dentro do domínio para "confie, mas verifique" em todos os lugares — isto é, presuma interações bem intencionadas com outras partes do sistema, mas verifique a confiança no nível local. Isso permite que os times desfrutem de um alto grau de controle sobre sua própria infraestrutura, equipamentos e stacks de aplicações, levando a uma alta visibilidade e, quando necessário, alta proteção para acesso. Ferramentas como Scout2 e técnicas como BeyondCorp refletem o amadurecimento dessa perspectiva de confiança. Recebemos com bons olhos essa mudança em direção à autonomia localizada, especialmente quando ferramentas e automação podem garantir uma conformidade igual ou melhor.

AS `coisas` EVOLUEM

O ecossistema da Internet das Coisas (IoT) continua a evoluir em um ritmo forte e constante, incluindo fatores críticos de sucesso como segurança e práticas de engenharia em amadurecimento. Observamos crescimento em todo o ecossistema de IoT, de sistemas operacionais em dispositivos a padrões de conectividade, e mais fortemente no gerenciamento de dispositivos e processamento de dados baseados em nuvem. Vemos maturidade em ferramentas e estruturas que suportam boas práticas de engenharia, como entrega contínua, implantação e uma série de outras necessidades para uma eventual difusão de uso. Além dos principais provedores de nuvem, incluindo Google IoT Core, AWS IoT e Microsoft Azure Hub IoT — empresas como Alibaba e Aliyun também estão investindo fortemente em soluções IoT PaaS. Nossos blips EMQ e Mongoose OS fornecem uma visão rápida dos principais recursos do ecossistema de IoT de hoje e ilustram que as coisas estão evoluindo muito bem.

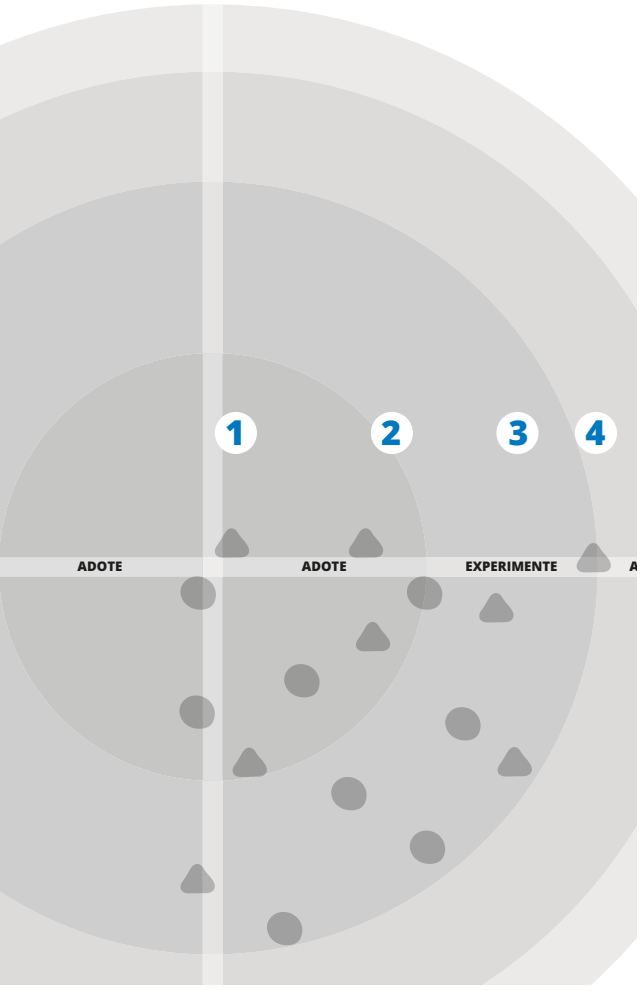
SOBRE O RADAR

'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
3. Função de aptidão arquitetural
4. Padrão de bolha autônoma
5. Engenharia de Caos
6. Eventos com escopo de domínio *NOVO*
7. Gerenciamento de identidades hospedado como serviço *NOVO*
8. Micro front-ends
9. Pipelines para infraestrutura como código
10. Polycloud

AVALIE
11. BeyondCorp *NOVO*
12. Mocks móveis integrados *NOVO*
13. Ethereum para aplicações descentralizadas
14. Streaming de eventos como fonte única da verdade
15. GraphQL para agregação de recursos do lado do servidor *NOVO*
16. Analisador automatizado da configuração de infraestrutura *NOVO*
17. Jupyter para testes automatizados *NOVO*
18. Nível de log por solicitação *NOVO*
19. Engenharia de Caos de Segurança *NOVO*
20. Malha de serviços
21. Sidecars para segurança de endpoints
22. Os três Rs de segurança

EVITE
23. Uso genérico da nuvem *NOVO*
24. Recriando anti-padrões ESB com Kafka

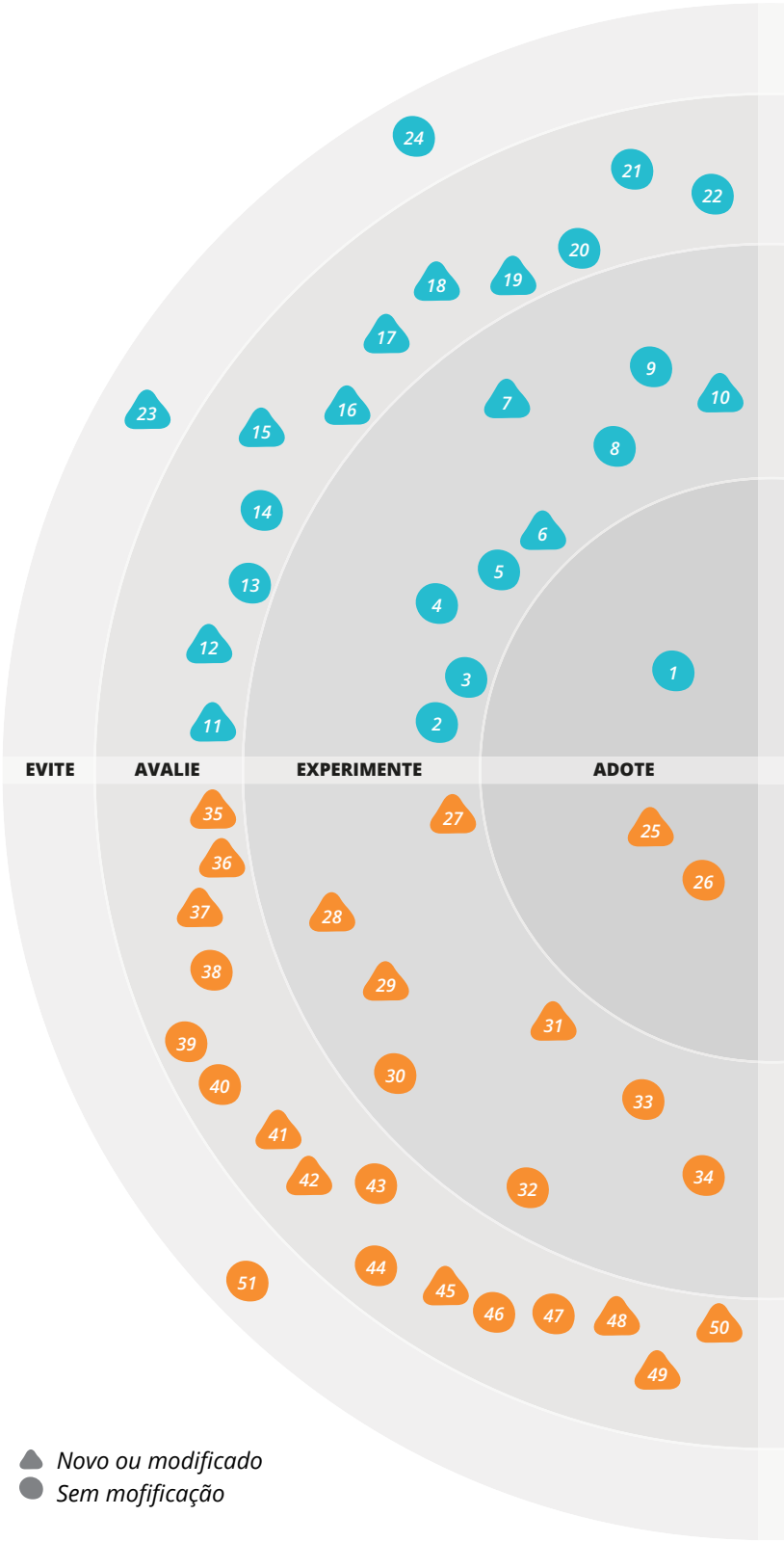
PLATAFORMAS

ADOTE
25. .NET Core
26. Kubernetes

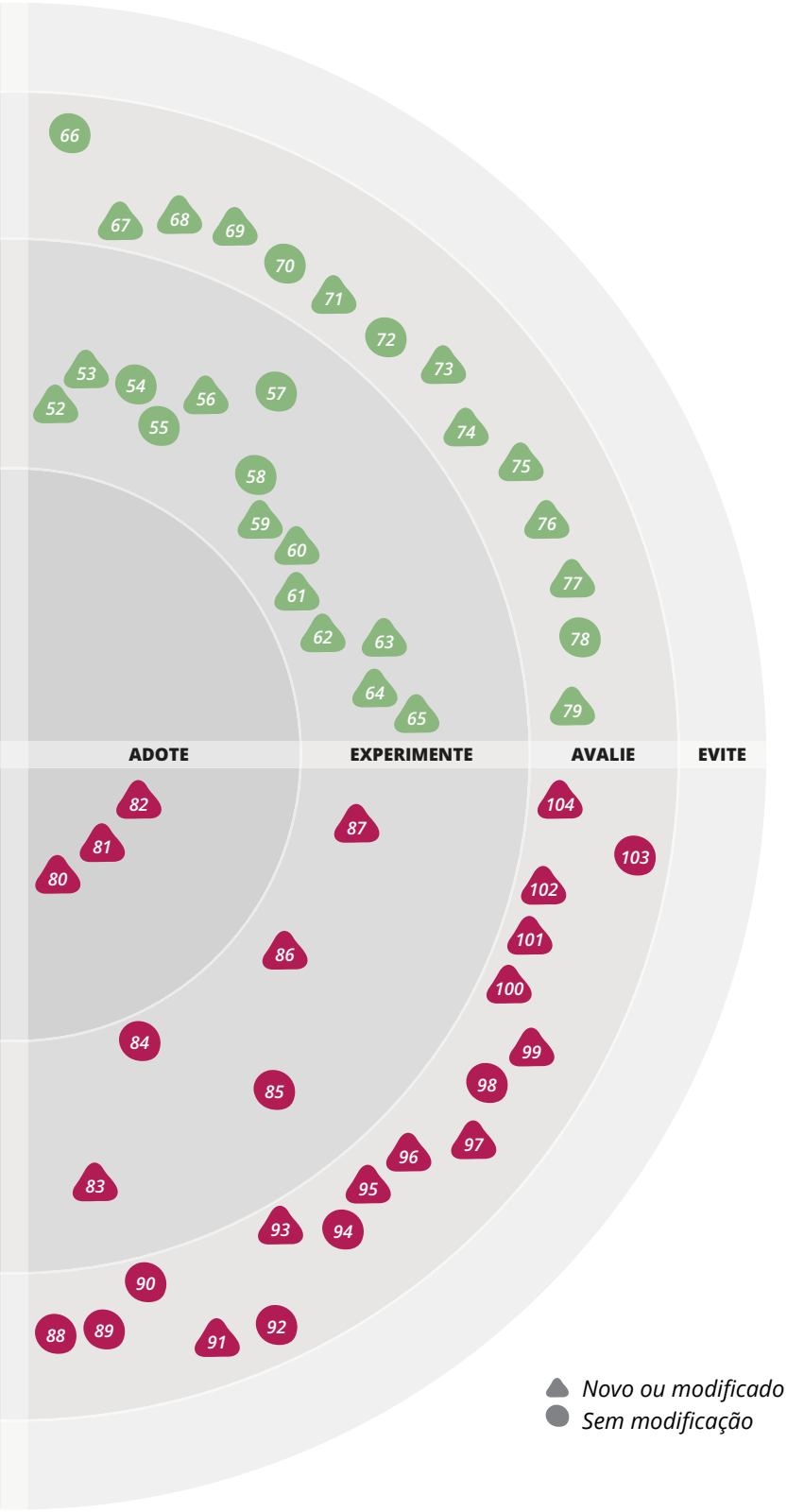
EXPERIMENTE
27. Azure
28. Contentful *NOVO*
29. EMQ *NOVO*
30. Flood IO
31. GKE
32. Google Cloud Platform
33. Keycloak
34. WeChat

AVALIE
35. AWS Fargate *NOVO*
36. Azure Service Fabric
37. Azure Stack *NOVO*
38. Cloud Spanner
39. Corda
40. Cosmos DB
41. Godot *NOVO*
42. Interledger *NOVO*
43. Language Server Protocol
44. LoRaWAN
45. Mongoose OS *NOVO*
46. Netlify
47. TensorFlow Serving
48. TICK Stack *NOVO*
49. Web Bluetooth *NOVO*
50. Contêineres do Windows

EVITE
51. API Gateways excessivamente ambiciosos



O RADAR



FERRAMENTAS

ADOTE

EXPERIMENTE
52. Appium Test Distribution *NOVO*
53. BackstopJS *NOVO*
54. Buildkite
55. CircleCI
56. CVXPY *NOVO*
57. gopass
58. Headless Chrome para testes de front-end
59. Helm *NOVO*
60. Jupyter
61. Kong API Gateway
62. kops
63. Patroni *NOVO*
64. WireMock *NOVO*
65. Yarn

AVALIE
66. Apex
67. ArchUnit *NOVO*
68. cfn_nag *NOVO*
69. Conduit *NOVO*
70. Cypress
71. Dependabot *NOVO*
72. Flow
73. Headless Firefox *NOVO*
74. nsp *NOVO*
75. Parcel *NOVO*
76. Scout2 *NOVO*
77. Sentry *NOVO*
78. Sonobuoy
79. Swashbuckle for .NET Core *NOVO*

EVITE

LINGUAGENS & FRAMEWORKS

ADOTE
80. AssertJ
81. Enzyme
82. Kotlin

EXPERIMENTE
83. Apollo *NOVO*
84. CSS Grid Layout
85. CSS Modules
86. Hyperledger Composer *NOVO*
87. OpenZeppelin *NOVO*

AVALIE
88. Android Architecture Components
89. Atlas and BeeHive
90. Clara rules
91. Flutter *NOVO*
92. Gobot
93. Hyperapp *NOVO*
94. PyTorch
95. Rasa *NOVO*
96. Reactor *NOVO*
97. RIBs *NOVO*
98. Solidity
99. SwiftNIO *NOVO*
100. Tensorflow Eager Execution *NOVO*
101. TensorFlow Lite *NOVO*
102. troposphere *NOVO*
103. Truffle
104. WebAssembly *NOVO*

EVITE

TÉCNICAS

ADOTE
1. Registros de decisões de arquiteturas leves

EXPERIMENTE
2. Aplicar gestão de produtos a plataformas internas
3. Função de aptidão arquitetural
4. Padrão de bolha autônoma
5. Engenharia de Caos
6. Eventos com escopo de domínio ***NOVO***
7. Gerenciamento de identidades hospedado como serviço ***NOVO***
8. Micro front-ends
9. Pipelines para infraestrutura como código
10. Polycloud

AVALIE
11. BeyondCorp ***NOVO***
12. Mocks móveis integrados ***NOVO***
13. Ethereum para aplicações descentralizadas
14. Streaming de eventos como fonte única da verdade
15. GraphQL para agregação de recursos do lado do servidor ***NOVO***
16. Analisador automatizado da configuração de infraestrutura ***NOVO***
17. Jupyter para testes automatizados ***NOVO***
18. Nível de log por solicitação ***NOVO***
19. Engenharia de Caos de Segurança ***NOVO***
20. Malha de serviços
21. Sidecars para segurança de endpoints
22. Os três Rs de segurança

EVITE
23. Uso genérico da nuvem ***NOVO***
24. Recriando anti-padrões ESB com Kafka

É importante lembrar que o encapsulamento se aplica a eventos e arquiteturas orientadas a eventos da mesma forma que se aplica a outras áreas do software. Especificamente, pense no escopo de um evento e se é esperado que ele seja consumido na mesma aplicação, no mesmo domínio ou em toda uma organização. Um **EVENTO COM ESCOPO DE DOMÍNIO** será consumido no mesmo domínio em que foi publicado, assim sendo, espera-se que o consumidor tenha acesso a um determinado contexto, recursos ou referências para agir no evento. Se o consumo estiver ocorrendo de forma mais ampla dentro de uma organização, o conteúdo do evento pode precisar ser diferente, e será preciso tomar cuidado para que não "vazem" detalhes de implementação dos quais outros domínios venham a depender.

O gerenciamento de identidade é um componente crítico da plataforma. Pessoas usuárias externas em aplicativos móveis precisam ser autenticadas, quem desenvolve precisa ter acesso aos componentes da infraestrutura de entrega e os microsserviços podem precisar se identificar com outros microsserviços. Você deve se perguntar se o gerenciamento de identidade deve ser "auto-hospedado". Em nossa experiência, é preferível uma solução de **GERENCIAMENTO DE IDENTIDADES HOSPEDADO COMO SERVIÇO** (SaaS). Acreditamos que provedores hospedados de primeira linha, como Auth0 e Okta podem fornecer melhores tempos de atividade e SLAs de segurança. Dito isso, algumas vezes a auto-hospedagem da solução é uma decisão realista, especialmente para empresas que têm disciplina operacional e recursos para fazer isso com segurança. As soluções de identidade corporativa de grande porte geralmente oferecem uma gama muito mais ampla de recursos, como direitos centralizados, relatórios de governança e separação de gerenciamento de tarefas, entre outros. No entanto, essas preocupações são tipicamente mais relevantes para identidades de pessoas funcionárias, especialmente em empresas regulamentadas com sistemas legados.

As organizações estão cada vez mais à vontade com a estratégia de **POLYCLOUD** — em vez de "apostar tudo" em um provedor, elas passam diferentes tipos de carga de trabalho para diferentes provedores com base em sua própria estratégia. Algumas delas aplicam a abordagem da melhor opção, por exemplo: colocar serviços padrão na AWS, mas usar o Google para aprendizagem de máquina e aplicações orientadas a dados e Azure para aplicações Microsoft Windows. Para algumas organizações, esta é uma decisão cultural e de negócio. Empresas de varejo, por exemplo, muitas vezes se recusam a armazenar seus dados na Amazon e distribuem carga em diferentes provedores com base em seus dados. Isso é diferente de uma estratégia agnóstica de nuvem de visar à 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 provedor de nuvem oferece.

Algumas organizações estão acabando com as intranets implicitamente confiáveis e tratando toda a comunicação como se estivesse sendo transmitida pela Internet pública.

(BeyondCorp)

Anteriormente no Radar, discutimos a ascensão das organizações sem fronteiras. Agora, algumas organizações estão acabando com as intranets implicitamente confiáveis e tratando toda a comunicação como se estivesse sendo transmitida pela Internet pública. Um conjunto de práticas, coletivamente rotuladas como **BEYONDCORP**, foi descrito por pessoas engenheiras do Google em uma série de publicações. Juntas, essas práticas — incluindo dispositivos gerenciados, redes 802.1x e proxies de acesso padrão que protegem serviços individuais — tornam esta uma abordagem viável para a segurança de rede em grandes empresas.

Ao desenvolver aplicações móveis, nossos times frequentemente se veem sem um servidor externo para testar os aplicativos. Configurar um mock over-the-wire pode ser uma boa solução para esse problema específico. Desenvolver mocks HTTP e compilá-los no binário móvel para testes — **MOCKS MÓVEIS INTEGRADOS** — permite que os times testem seus aplicativos móveis quando desconectados e sem dependências externas. Essa técnica pode exigir a criação de uma biblioteca opinativa baseada na biblioteca de rede usada pelo aplicativo móvel e no seu uso da biblioteca subjacente.

Um padrão que surge constantemente ao criar arquiteturas no estilo de microsserviços é como lidar com a agregação de muitos recursos do lado do servidor. Nos últimos anos, vimos o surgimento de vários padrões, como Back-end para Front-ends (BFF) e de ferramentas, como Falcor, para resolver isso. Nossos times começaram a usar **GRAPHQL PARA AGREGAÇÃO DE RECURSOS DO LADO DO SERVIDOR**. Isso difere do modo usual de usar GraphQL, no qual clientes consultam diretamente um servidor GraphQL. Ao usar essa técnica, os serviços continuam a expor APIs RESTful, mas os serviços agregados ocultos usam resolvedores GraphQL como implementação para costurar recursos de outros serviços. Essa técnica simplifica a implementação interna de serviços agregados ou BFFs usando GraphQL.

Já há algum tempo recomendamos um maior controle do time de desenvolvimento sobre toda sua stack, incluindo a infraestrutura. Isso significa maior responsabilidade do próprio time de desenvolvimento para configurar a infraestrutura de maneira segura, protegida e complacente. Ao adotar estratégias de nuvem, a maioria das organizações usa como padrão uma configuração bem controlada e centralizada para reduzir riscos, mas isso também cria gargalos substanciais de produtividade. Uma abordagem alternativa é permitir que os times gerenciem sua própria configuração, e usar um **ANALISADOR AUTOMATIZADO DA CONFIGURAÇÃO DE INFRAESTRUTURA** para garantir que a configuração seja definida de maneira segura e protegida. O Watchmen é uma ferramenta interessante, criada para prover garantia orientada por regras para configurações de contas da AWS que são controladas e operadas de forma independente pelos times de desenvolvimento. O Scout2 é outro exemplo desses analisadores para dar suporte à adequação aos princípios de segurança.

Temos visto alguns relatos interessantes sobre o uso de **JUPYTER PARA TESTES AUTOMATIZADOS**. A capacidade de misturar código, comentários e saída no mesmo documento nos lembra FIT, FitNesse e Concordion. Essa abordagem flexível é particularmente útil se seus testes precisam de muitos dados ou se basearem em análises estatísticas como testes de desempenho. Python fornece todo o poder que você precisa, mas à medida que os testes crescem em complexidade, uma forma de gerenciar suítes de notebooks seria útil.

Um problema com a observabilidade em uma arquitetura de microsserviços altamente distribuída é a escolha entre registrar tudo — e ocupar grandes quantidades de espaço de armazenamento — ou usar amostras de logs aleatoriamente e potencialmente perder eventos importantes. Recentemente, notamos uma técnica que oferece um meio-termo entre essas duas soluções. Definir o **NÍVEL DE LOG POR SOLICITAÇÃO** por meio de um parâmetro passado pelo cabeçalho de rastreamento. Usando um framework de rastreamento, possivelmente baseado no padrão OpenTracing, você pode passar uma id de correlação de serviço a serviço em uma única transação. Você pode até mesmo injetar outros dados, como o nível de log desejado, na transação

inicial e passá-lo junto com as informações de rastreio. Isso garante que os dados adicionais coletados correspondam a uma única transação de usuário enquanto ela flui pelo sistema. Essa também é uma técnica útil para depuração, já que os serviços podem ser pausados ou modificados transação por transação.

Nós deliberadamente introduzimos falsos positivos em redes de produção e outras infraestruturas para verificar se os procedimentos em vigor são capazes de identificar falhas de segurança sob condições controladas.

(Engenharia de Caos de Segurança)

Falamos anteriormente sobre a técnica de Engenharia de Caos no Radar e na suíte de ferramentas do Simian Army da Netflix, que usamos para executar experimentos para testar a resiliência da infraestrutura de produção. A **ENGENHARIA DE CAOS DE SEGURANÇA** amplia o escopo dessa técnica para o campo da segurança. Nós deliberadamente introduzimos falsos positivos em redes de produção e outras infraestruturas — dependências de tempo de compilação, por exemplo — para verificar se os procedimentos em vigor são capazes de identificar falhas de segurança sob condições controladas. Embora útil, essa técnica deve ser usada com cuidado para evitar a dessensibilização de times em relação a problemas de segurança.

Cada vez mais, estamos vendo organizações se prepararem para usar várias nuvens — porém não para se beneficiar dos pontos fortes individuais dos provedores, mas para evitar a todo custo o "aprisionamento" a um fornecedor.

(Uso Genérico da Nuvem)

Os principais provedores de nuvem continuam adicionando novos recursos às suas nuvens em um ritmo acelerado e, sob o rótulo de Polycloud, sugerimos o uso de várias nuvens em paralelo, para misturar e combinar serviços com base nos pontos fortes das ofertas de cada provedor. Cada vez mais, estamos vendo organizações se prepararem para usar várias nuvens — porém não para se beneficiar dos pontos fortes individuais dos provedores, mas para evitar a todo custo o "aprisionamento" a um fornecedor. Isso, obviamente, leva ao **USO GENÉRICO DA NUVEM**, utilizando apenas recursos presentes em todos os provedores, o que nos lembra o cenário de menor denominador comum que vimos há 10 anos quando as empresas evitavam muitos recursos avançados em bancos de dados relacionais em um esforço para permanecerem neutras em relação a fornecedores. O problema do aprisionamento é real. No entanto, em vez de tratá-lo com uma abordagem que não lida diretamente com o problema, recomendamos que você o analise do ponto de vista dos custos de saída, relacionando-os aos benefícios de usar recursos específicos da nuvem.

PLATAFORMAS

ADOTE
25. .NET Core
26. Kubernetes

EXPERIMENTE
27. Azure
28. Contentful *NOVO*
29. EMQ *NOVO*
30. Flood IO
31. GKE
32. Google Cloud Platform
33. Keycloak
34. WeChat

AVALIE
35. AWS Fargate *NOVO*
36. Azure Service Fabric
37. Azure Stack *NOVO*
38. Cloud Spanner
39. Corda
40. Cosmos DB
41. Godot *NOVO*
42. Interledger *NOVO*
43. Language Server Protocol
44. LoRaWAN
45. Mongoose OS *NOVO*
46. Netlify
47. TensorFlow Serving
48. TICK Stack *NOVO*
49. Web Bluetooth *NOVO*
50. Contêineres do Windows

EVITE
51. API Gateways excessivamente ambiciosos

Nossos times confirmaram que o **.NET CORE** atingiu um nível de maturidade que o faz padrão para aplicações do servidor .NET. A estrutura .NET Core de código aberto permite o desenvolvimento e a implantação de aplicações .NET no Windows, macOS e Linux com ferramentas multiplataforma de primeira classe. A Microsoft fornece imagens blessed do Docker, que facilitam a implantação de aplicativos .NET Core em um ambiente de contêiner. Direções positivas na comunidade e feedbacks dos nossos projectos indicam que o .NET Core é o futuro para o desenvolvimento .NET.

A Microsoft aprimorou consistentemente o **AZURE** e hoje não não há muita distinção na experiência principal de nuvem fornecida pelos principais provedores de nuvem – Amazon, Google e Microsoft. Os provedores de nuvem parecem concordar e buscam se diferenciar em outras áreas, como recursos, serviços e estrutura de custos. A Microsoft é a provedora que demonstra interesse real nos requisitos legais das empresas européias. Ela possui uma estratégia diferenciada e plausível, que inclui ofertas exclusivas, como Azure Alemanha e Azure Stack, e que dá alguma certeza às empresas européias se antecipando ao GDPR e possíveis mudanças legislativas nos Estados Unidos.

Sistemas de gerenciamento de conteúdo headless (CMSes) estão se tornando um componente comum de plataformas digitais. **CONTENTFUL** é um CMS headless moderno que nossos times integraram com êxito em seus fluxos de desenvolvimento. Gostamos particularmente de sua abordagem API primeiro e da implementação de CMS como código. Ele suporta primitivas poderosas de modelagem de conteúdo como código e scripts de evolução de modelo de conteúdo que permitem tratá-lo como outros esquemas de armazenamento de dados e aplicar práticas de design de banco de dados evolucionário ao desenvolvimento do CMS. Outros recursos notáveis que gostamos incluem a adição de duas CDNs por padrão para entregar ativos de mídia e documentos JSON, bom suporte para localização e a capacidade — embora com algum esforço — de integração com Auth0.

EMQ é um broker MQTT multiplataforma de código aberto e escalável. É escrito em Erlang/OTP para melhor desempenho, lidando com milhões de conexões simultâneas. Ele suporta vários protocolos, incluindo MQTT, MQTT para Redes de Sensores, bem como WebSockets, tornando-o apropriado para dispositivos IoT e móveis. Nós começamos a usar EMQ em nossos

projetos e gostamos de sua facilidade de instalação e uso, sua capacidade de rotear mensagens para diferentes destinos, incluindo Kafka e PostgreSQL, bem como sua abordagem orientada a API para monitoramento e configuração.

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. Google Kubernetes Engine (**GKE**) é uma solução Kubernetes gerenciada para implantação de aplicações em contêineres que aprimora 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 abordagem API-first do Google exibindo os recursos da plataforma, bem como o uso de 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 o público consumidor das mudanças subjacentes, pode impactar você. Esperamos uma melhoria contínua na maturidade da Infraestrutura como código com Terraform no GKE e ferramentas similares.

AWS FARGATE é uma entrada recente no espaço de Docker como serviço, atualmente limitada à região EUA-Leste-1. Para os times que usam AWS Elastic Container Service (ECS), AWS Fargate é uma boa alternativa sem a necessidade de gerenciar, provisionar e configurar quaisquer instâncias ou clusters subjacentes do EC2. Fargate permite definir tarefas (ECS ou EKS - ECS para Kubernetes) como um tipo Fargate, e elas serão executadas na infraestrutura do AWS Fargate. Se você gosta do foco na funcionalidade de negócio que o AWS Lambda oferece, Fargate é o mais próximo que você pode chegar quando as aplicações não podem ser implantadas como funções únicas.

A **AZURE SERVICE FABRIC** é uma plataforma de sistemas distribuídos construída para microsserviços e contêineres. O que distingue a Service Fabric, entretanto, são modelos de programação como Reliable Actors, construídos em cima de serviços confiáveis. Quando se trata de casos de uso de IoT, por exemplo, o Reliable Actors oferece algumas vantagens convincentes — além dos benefícios de confiabilidade e plataforma por estar na Service Fabric, você também desfruta de seus recursos de gerenciamento de estado e replicação. Mantendo o foco contínuo no software de código aberto (OSS), a Microsoft fará a transição da Service Fabric para um processo de desenvolvimento aberto no Github. Tudo isso faz valer a pena testar a Azure Service Fabric — principalmente para organizações que investem no framework .NET.

A Microsoft adiciona uma oferta interessante que funciona como um meio termo entre nuvens públicas completas em recursos e virtualização local simples.

(Azure Stack)

A computação em nuvem traz benefícios significativos em relação a soluções auto-hospedadas virtualizadas, mas às vezes os dados simplesmente não podem sair das instalações físicas de uma organização, geralmente em razão de latência ou regulamentação. Para empresas européias, o clima político atual também gera mais preocupações sobre colocar dados nas mãos de entidades sediadas nos EUA. Com **AZURE STACK**, a Microsoft adiciona uma oferta interessante que funciona como um meio-termo entre nuvens públicas completas em recursos e virtualização local simples: uma versão enxuta do software que executa a nuvem Azure Global da Microsoft é combinada com um rack de hardware commodity pré-configurado dos fornecedores de sempre, como HP e Lenovo, provendo a uma organização a experiência básica do Azure localmente. Por padrão, o suporte é dividido entre a Microsoft e as fornecedoras de hardware (e elas prometem cooperar), mas integradores de sistemas também podem oferecer soluções completas do Azure Stack.

À medida que RA e RV continuam a ganhar força, continuamos a explorar ferramentas com as quais podemos criar mundos virtuais imersivos. Nossa experiência positiva com Unity, um dos dois principais mecanismos de jogos, nos levou a apresentá-lo em Radares anteriores. Ainda gostamos de Unity, mas também nos animamos com **GODOT**, um participante relativamente novo na área. Godot é um software de código aberto e, embora não tão completo quanto os grandes motores comerciais, ele vem com um design de software mais moderno e menos desorganizado. O fato de oferecer C# e Python reduz ainda mais a resistência à adoção de pessoas desenvolvedoras fora do setor de jogos. A versão 3 do Godot, lançada no início deste ano, adiciona suporte a RV, e há perspectiva de suporte a RA.

A maioria das pessoas deve conhecer a "Internet do dinheiro" por meio da Bitcoin. De fato, essa ideia pode ser encontrada já nos estágios iniciais da Web. O HTTP até reservou um código de status para pagamento digital.
(Interledger)

A maioria das pessoas deve conhecer a "Internet do dinheiro" por meio da Bitcoin. De fato, essa ideia pode ser encontrada já nos estágios iniciais da Web. O HTTP até reservou um código de status para pagamento digital. A parte desafiadora dessa ideia é transferir valor entre diferentes registros em diferentes entidades. A tecnologia de Blockchain promove essa ideia por meio da criação de um registro compartilhado distribuído. O desafio atual é como alcançar a interoperabilidade entre diferentes registros de blockchain e a interoperabilidade com os tradicionais registros centralizados. O **INTERLEDGER** é um protocolo para conectar diferentes registros. Esse protocolo usa conectores e um mecanismo de criptografia, como o HTLC, para encaminhar pagamentos seguros entre registros. Não é difícil ingressar na rede de pagamento por meio de suas suítes. O Interledger foi iniciado pela Ripple e agora é constantemente desenvolvido por um grupo da comunidade W3C.

Com um crescimento acelerado de dispositivos embarcados conectados e maior acessibilidade do hardware, **MONGOOSE OS** preenche uma notável lacuna para quem desenvolve software embarcado: a lacuna entre firmware Arduino adequado para prototipagem e SDKs nativos de microcontroladores metal puro. O Mongoose OS é um sistema operacional de microcontrolador que vem com um conjunto de bibliotecas e um framework de desenvolvimento para suportar aplicações típicas de Internet das Coisas (IoT) com conectividade a servidores MQTT e plataformas populares de nuvem IoT, como Google Cloud IoT Core e AWS IoT por padrão. Na verdade, o Google recomenda um kit inicial do Mongoose para o Cloud IoT Core. Tivemos uma experiência perfeita usando Mongoose OS em nossos projetos incorporados construindo espaços de trabalho conectados. Gostamos especialmente de sua segurança integrada no nível de dispositivo individual e das atualizações de firmware OTA, entre outros recursos. No momento em que este texto foi escrito, apenas um número limitado de microcontroladores e placas era compatível, com os microcontroladores baseados em ARM mais populares ainda em desenvolvimento.

TICK STACK é uma plataforma composta por componentes de código aberto, o que facilita a coleta, armazenamento, geração de gráficos e alertas de dados de série na hora, como métricas e eventos. Os componentes da TICK Stack são: Telegraf, um agente servidor para coletar e reportar métricas; InfluxDB, um banco de dados de séries temporais de alto desempenho; Chronograf, uma interface de usuário para a plataforma; e Kapacitor, um mecanismo de processamento de dados que pode processar, transmitir e agrupar dados do InfluxDB. Ao contrário de Prometheus, que é baseado no modelo Pull, TICK Stack se baseia no modelo Push de coleta de dados. O coração do sistema é o componente InfluxDB, que é um dos melhores bancos de dados de séries temporais. Essa stack é apoiada pelo InfluxData e precisa da versão corporativa para recursos como clusterização de bancos de dados, mas ainda é uma boa opção para monitoramento. Estamos usando em alguns locais em produção e tivemos boas experiências.

WEB BLUETOOTH nos permite controlar qualquer dispositivo Bluetooth de baixa energia diretamente pelo navegador. Isso nos permite concentrar em cenários que anteriormente só poderiam ser resolvidos com um aplicativo nativo. A especificação é publicada pelo Web Bluetooth Community Group e descreve uma API para descoberta e comunicação com dispositivos pelo padrão Bluetooth 4 sem fio. No momento, o Chrome é o único entre os principais navegadores a suportar essa especificação. Com a Web física e a Web Bluetooth, agora temos outros caminhos para fazer pessoas usuárias interagirem com os dispositivos sem precisar instalar outro aplicativo em seu telefone. Este é um espaço promissor que vale a pena ficar de olho.

A Microsoft está se recuperando no espaço dos contêineres com **CONTÊINERES DO WINDOWS** permitindo a execução de aplicações do Windows como contêineres em ambientes baseados em Windows. No momento em que esse texto foi escrito, a Microsoft fornecia duas imagens do sistema operacional Windows como contêineres do Docker — Windows Server 2016 Server Core e Windows Server 2016 Nano Server — que podem ser executadas como Contêineres do Windows Server com o Docker. Nossos times começaram a usar contêineres do Windows em cenários nos quais os agentes de compilação e contêineres semelhantes têm funcionado com êxito. A Microsoft está ciente de que há melhoras a serem feitas, como reduzir os tamanhos de imagem e enriquecer o suporte e a documentação do ecossistema.

FERRAMENTAS

ADOTE

EXPERIMENTE
52. Appium Test Distribution *NOVO*
53. BackstopJS *NOVO*
54. Buildkite
55. CircleCI
56. CVXPY *NOVO*
57. gopass
58. Headless Chrome para testes de front-end
59. Helm *NOVO*
60. Jupyter
61. Kong API Gateway
62. kops
63. Patroni *NOVO*
64. WireMock *NOVO*
65. Yarn

AVALIE
66. Apex
67. ArchUnit *NOVO*
68. cfn_nag *NOVO*
69. Conduit *NOVO*
70. Cypress
71. Dependabot *NOVO*
72. Flow
73. Headless Firefox *NOVO*
74. nsp *NOVO*
75. Parcel *NOVO*
76. Scout2 *NOVO*
77. Sentry *NOVO*
78. Sonobuoy
79. Swashbuckle for .NET Core *NOVO*

Incluímos Appium no Radar no passado. É um dos frameworks de automação de testes para dispositivos móveis mais populares. À medida que escalamos nossa suíte de testes, a capacidade de executar nossos testes em paralelo para uma variedade de dispositivos é fundamental para ter ciclos de feedback curtos. **APPIUM TEST DISTRIBUTION** resolve este problema de forma muito eficaz com a sua capacidade de executar testes em paralelo, bem como executar os mesmos testes em múltiplos dispositivos. Entre outras coisas, ele se distingue por sua capacidade de adicionar e remover dispositivos nos quais os testes são executados sem a necessidade de qualquer configuração manual e por sua capacidade de executar testes em dispositivos remotos. Nós o usamos em alguns projetos na ThoughtWorks nos últimos anos e ele nos atendeu muito bem.

Estamos gostando de **BACKSTOPJS** para testes visuais de regressão de aplicações da web. As janelas de visualização configuráveis e a capacidade de ajustar as tolerâncias são particularmente úteis, assim como a ferramenta de comparação visual, que facilita a identificação de pequenas variações. A ferramenta possui boa capacidade de script e oferece a opção de executar Headless Chrome, PhantomJS e SlimerJS. Consideramos particularmente útil quando executada com guias de estilo de componentes vivos.

É surpreendente como muitos problemas podem ser expressos como problemas de otimização matemática e muitas vezes problemas convexos, que podem ser eficientemente resolvidos.

(CVXPY)

É surpreendente como muitos problemas podem ser expressos como problemas de otimização matemática e muitas vezes problemas convexos, que podem ser eficientemente resolvidos. **CVXPY** é uma linguagem de modelagem de código aberto incorporada em Python para problemas de otimização convexa. Ela é mantida por pessoas acadêmicas da Universidade de Stanford e oferece uma instalação que dispensa o uso de bibliotecas e pacotes externos para vários

solucionadores de código aberto e comerciais. A documentação inclui muitos exemplos que devem inspirar pessoas desenvolvedoras a usá-la. É particularmente útil para soluções de prototipagem, ainda que solucionadores comercialmente licenciados, como Gurobi ou IBM CPLEX, possam ser necessários. Na maioria dos casos, porém, é suficiente por si só. Entretanto, o mesmo grupo escreveu diversos pacotes de extensão, como DCCP, e softwares relacionados, como CVXOPT, tendo como base os recentes avanços na otimização.

HELM é um gerenciador de pacotes para Kubernetes. O conjunto de recursos do Kubernetes que definem uma aplicação é empacotado como uma série de gráficos. Esses gráficos podem descrever um único recurso, como um Redis pod, ou uma stack completa de uma aplicação da Web: servidores HTTP, bancos de dados e caches. O Helm, por padrão, vem com um repositório de aplicações curadas do Kubernetes que são mantidas no repositório de gráficos oficial. É fácil também configurar um repositório de gráficos privado para uso interno. O Helm tem dois componentes: um utilitário de linha de comando chamado Helm e um componente de cluster chamado Tiller. Proteger um cluster do Kubernetes é um tópico amplo e cheio de nuances, mas é altamente recomendável configurar o Tiller em um ambiente de controle de acesso baseado em função (RBAC). Usamos Helm em inúmeros projetos de clientes e seu gerenciamento de dependências, templates e mecanismo de gancho simplificou muito o gerenciamento do ciclo de vida de aplicações no Kubernetes.

Nos últimos anos, notamos um aumento constante na popularidade de notebooks analíticos. São aplicações inspiradas em Mathematica que combinam texto, visualização e código em um documento computacional vivo.

(Jupyter)

Nos últimos anos, notamos um aumento constante na popularidade de notebooks analíticos. São aplicações inspiradas em Mathematica que combinam texto, visualização e código em um documento computacional vivo. O crescimento do interesse em aprendizagem de máquina — junto com o advento de Python como linguagem de programação de escolha para profissionais dessa área — voltou as atenções especificamente para notebooks Python, dos quais **JUPYTER** parece estar ganhando mais tração entre os times da ThoughtWorks. As pessoas parecem continuar encontrando usos criativos para Jupyter, que vão além de uma simples ferramenta de análise. Veja, por exemplo, Jupyter para testes automatizados.

Kong é um API gateway de código aberto que também é oferecido como um produto corporativo, integrando com analytics de API proprietária e com um portal de desenvolvimento. Kong pode ser implantado em uma variedade de configurações, como um API gateway de borda, como um proxy de API interno ou até mesmo como um sidecar em uma configuração de malha de serviço. O OpenResty, por meio de seus módulos Nginx, fornece uma base sólida e de alto desempenho, com plugins Lua para extensões. Kong pode usar PostgreSQL para implantações de região única ou o Cassandra para configurações multirregionais. Nossas pessoas desenvolvedoras têm gostado do alto desempenho do Kong, de sua abordagem API primeiro (que permite a automação de sua configuração) e de sua facilidade de implantação como contêiner. **KONG API GATEWAY**, ao contrário de API gateways excessivamente ambiciosos, 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.

KOPS é uma ferramenta de linha de comando para criar e gerenciar clusters de produção do Kubernetes de alta disponibilidade. O kops tornou-se nossa ferramenta de escolha para autogerenciar clusters do Kubernetes em AWS, não apenas por sua crescente comunidade de código aberto. Ele também suporta a instalação, a atualização e o gerenciamento de clusters do Kubernetes na Google Cloud. Nossa experiência com kops no Google, no entanto, é muito limitada devido à nossa preferência por GKE, a oferta gerenciada do Kubernetes. Recomendamos o uso de kops em scripts reutilizáveis para criar a infraestrutura como código. Temos interesse em ver como o kops vai continuar a evoluir para suportar clusters gerenciados do Kubernetes, como o EKS, o serviço de Kubernetes gerenciado da Amazon.

PATRONI é um modelo para alta disponibilidade de PostgreSQL. Criado a partir da necessidade de fornecer falha automática para PostgreSQL, o Patroni é um controlador PostgreSQL baseado em Python que utiliza um armazenamento distribuído de configuração (como etcd, ZooKeeper ou Consul) para gerenciar o estado do cluster de PostgreSQL. O Patroni suporta tanto modelos de streaming quanto de replicação síncrona e fornece um conjunto avançado de APIs REST para configuração dinâmica do cluster de PostgreSQL. Se você deseja obter alta disponibilidade em uma configuração distribuída de PostgreSQL, é preciso considerar muitos casos de borda, e gostamos do fato de o Patroni fornecer um modelo para executar a maioria dos casos de uso comuns.

Um fator chave para arquiteturas baseadas em microsserviços é a adaptabilidade evolutiva independente dos serviços. Por exemplo, quando dois serviços dependem um do outro, o processo de teste de um geralmente envolve stubs e mocks do outro. Eles podem ser escritos à mão, mas assim como acontece com mocking em testes unitários, um framework ajuda quem está desenvolvendo a se concentrar no cenário de testes de fato. Sabemos do **WIREMOCK** há algum tempo, mas preferimos executar testes com mountebank. Durante o último ano, porém, o WireMock realmente se equiparou e agora o recomendamos como uma boa alternativa.

YARN é um gerenciador de pacotes rápido, confiável e seguro para JavaScript. Usando um arquivo de bloqueio e um algoritmo determinístico, Yarn é capaz de garantir que uma instalação que funcionou em um sistema funcionará exatamente da mesma maneira em qualquer outro sistema. Ao enfileirar solicitações de forma eficiente, o Yarn maximiza a utilização da rede e, como resultado, observamos downloads de pacotes mais rápidos. Yarn continua sendo nossa ferramenta de escolha para o gerenciamento de pacotes JavaScript, apesar das últimas melhorias no npm (versão 5).

ARCHUNIT é uma biblioteca de testes Java para checagem de características de arquitetura, como dependências de pacotes e classes, verificação de anotações e até mesmo consistência de camadas. O fato de ser executada como teste de unidade, dentro de sua configuração de teste existente, nos agrada, embora esteja disponível apenas para arquiteturas Java. O conjunto de testes da ArchUnit pode ser incorporado em um ambiente de CI ou em um pipeline de implantação, facilitando a implementação de funções de fitness em uma arquitetura evolucionária.

Nuvem e entrega contínua tiveram um efeito dramático na segurança da infraestrutura. Ao seguir a infraestrutura como código, toda a infraestrutura — o que inclui redes, firewalls e contas — é definida em scripts e arquivos de configuração, e com Servidores e Ambientes Fênix a infraestrutura é recriada em cada implantação, com frequência várias vezes por dia. Nesse cenário, testar a infraestrutura depois de criada não é suficiente nem viável. Uma ferramenta que ajuda a resolver este problema é **CFN-NAG**. Ela verifica os templates da CloudFormation usados com AWS para padrões que podem indicar uma infraestrutura insegura, e faz isso antes que a infraestrutura seja criada. Executar uma ferramenta como cfn-nag em um pipeline de complicação é rápido e pode ajudar a detectar vários problemas antes mesmo que eles cheguem a um ambiente de nuvem.

CONDUIT é uma malha de serviços leve para Kubernetes. O Conduit abrange a arquitetura fora do processo com um proxy de plano de dados escrito em Rust e um plano de controle em Go. O proxy do plano de dados é executado como um sidecar para todo o tráfego TCP no cluster do Kubernetes e o plano de controle é executado em um namespace separado no Kubernetes, expondo APIs REST para controlar o comportamento do proxy do plano de dados. Ao aplicar o proxy em todas as solicitações, o Conduit fornece uma grande variedade de métricas para monitoramento e observabilidade de interações na malha de serviço para tráfego HTTP, HTTP/2 e gRPC. Embora o Conduit seja relativamente novo nesse espaço, o recomendamos pelo fato de ser simples de instalar e operar.

Manter as dependências atualizadas é uma tarefa trabalhosa, mas é importante gerenciar atualizações com frequência e de maneira incremental. Queremos que o processo seja tão indolor e automatizado quanto possível. Nossos times costumam executar scripts manuais para automatizar partes do processo; Agora, no entanto, integramos ofertas comerciais para fazer esse trabalho. **DEPENDABOT** é um serviço que se integra aos repositórios do GitHub e verifica automaticamente as dependências do seu projeto para novas versões. Quando necessário, o Dependabot abrirá um pull request com dependências atualizadas. Usando recursos do seu servidor de CI, você pode testar automaticamente as atualizações para compatibilidade e mesclar automaticamente as atualizações compatíveis com a árvore principal de desenvolvimento. Existem alternativas para o Dependabot, incluindo Renovate para projetos

JavaScript e Depfu para projetos em JavaScript e Ruby. Nossos times, no entanto, recomendam Dependabot devido ao seu suporte multilinguagem e facilidade de uso.

Ao desenvolver aplicações front-end, mencionamos Headless Chrome como uma melhor alternativa ao Phantom JS para testes de front-end em uma edição anterior do Radar. Agora, sugerimos que você avalie o **HEADLESS FIREFOX** como uma opção viável nessa área. Da mesma forma que o Headless Chrome, o Firefox em modo headless executa o navegador sem os componentes visíveis da interface de usuário, executando o conjunto de testes da IU com muito mais rapidez.

NSP é uma ferramenta de linha de comando para identificar vulnerabilidades conhecidas em aplicações Node.js. Ao executar o comando de verificação na raiz de um projeto Node.js, o nsp gera o relatório de vulnerabilidades verificando os alertas publicados. O nsp fornece uma maneira de personalizar o comando de verificação para ocultar todas as vulnerabilidades abaixo de uma determinada pontuação CVSS ou sair com um código de erro, caso pelo menos um achado tiver uma pontuação CVSS acima do valor determinado. Uma vez que os alertas forem salvos por meio do comando gather, o nsp também pode ser usado no modo offline.

PARCEL é um empacotador de aplicações Web semelhante ao Webpack e Browserify. Apresentamos o Webpack anteriormente em nosso Radar e ele continua sendo uma ótima ferramenta. Parcel se distingue de seus concorrentes pela experiência de desenvolvimento e pela velocidade. Ele tem todos os recursos de empacotamento padrão e fornece uma real experiência de configuração zero, facilitando seu uso. Possui tempos de empacotamento rápidos e supera seus concorrentes em muitas benchmarks. Parcel despertou muito interesse da comunidade e vale a pena ficar de olho.

SCOUT2 é uma ferramenta de auditoria de segurança para ambientes AWS. Em vez de navegar manualmente por páginas da Web, você pode confiar no Scout2 para buscar todos os dados de configuração de um ambiente AWS para você. Ele até mesmo gera um relatório de superfície de ataque. O Scout2 vem com regras pré-configuradas e pode ser facilmente estendido para suportar mais serviços e casos de teste. Como o Scout2 só executa chamadas à API da AWS para buscar dados de configuração e identificar lacunas de segurança, não é necessário preencher e enviar o Formulário de Solicitação de Teste de Vulnerabilidade / Penetração da AWS.

SENTRY é uma ferramenta de rastreamento de erros que ajuda a monitorar e corrigir erros em tempo real. Ferramentas de rastreamento e gerenciamento de erros como Sentry distinguem-se das soluções de registro tradicionais, como ELK Stack, por seu foco na descoberta, investigação e correção de erros. Sentry já existe há algum tempo e é bastante popular — ferramentas de rastreamento de erros são cada vez mais úteis com o foco atual em "tempo médio de recuperação". Sentry — com suas opções de integração com Github, Hipchat, Heroku, Slack, entre outras plataformas — nos permite ficar de olho em nossos aplicativos. Pode fornecer notificações de erro após uma implantação, permite rastrear se novos commits realmente solucionam o problema e nos alerta se um problema retornar devido a uma regressão.

No estado atual dos serviços de tecnologia, a exposição das APIs RESTFul é cada vez mais adotada e a documentação da API é muito importante para as pessoas consumidoras.

(Swashbuckle for .NET Core)

No estado atual dos serviços de tecnologia, a exposição das APIs RESTFul é cada vez mais adotada e a documentação da API é muito importante para as pessoas consumidoras. Neste espaço, o Swagger tem sido amplamente usado entre os times e gostaríamos de destacar **SWASHBUCKLE PARA .NET CORE**. Swashbuckle para .NET Core é uma ferramenta que gera uma documentação viva da API no Swagger, baseada no código dos projetos .NET Core. Ao usá-la, você também pode explorar e testar operações de APIs por meio de sua interface de usuário.

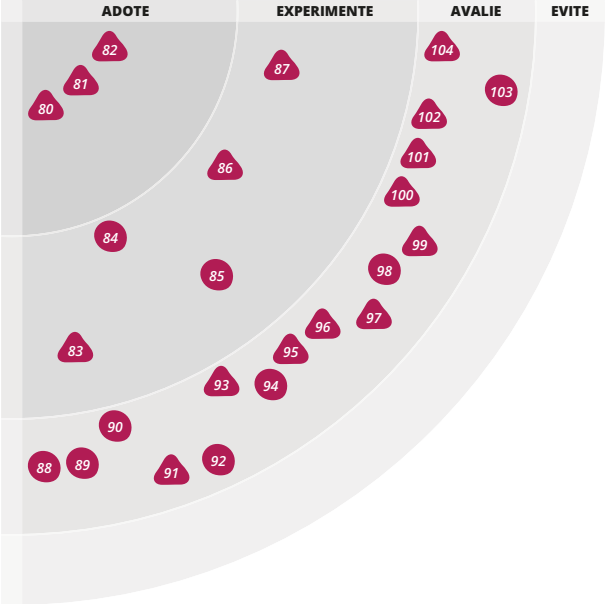
LINGUAGENS & FRAMEWORKS

ADOTE
80. AssertJ
81. Enzyme
82. Kotlin

EXPERIMENTE
83. Apollo *NOVO*
84. CSS Grid Layout
85. CSS Modules
86. Hyperledger Composer *NOVO*
87. OpenZeppelin *NOVO*

AVALIE
88. Android Architecture Components
89. Atlas and BeeHive
90. Clara rules
91. Flutter *NOVO*
92. Gobot
93. Hyperapp *NOVO*
94. PyTorch
95. Rasa *NOVO*
96. Reactor *NOVO*
97. RIBs *NOVO*
98. Solidity
99. SwiftNIO *NOVO*
100. Tensorflow Eager Execution *NOVO*
101. TensorFlow Lite *NOVO*
102. troposphere *NOVO*
103. Truffle
104. WebAssembly *NOVO*

EVITE



ASSERTJ é uma biblioteca Java que fornece uma interface fluente para asserções, o que facilita a transmissão de intenção dentro do código de teste. AssertJ fornece mensagens de erro legíveis e asserções suaves, além de coleções e suporte a exceções aprimorados. Muitos de nossos times escolhem AssertJ como sua biblioteca de asserção padrão ao invés de JUnit combinada com Java Hamcrest.

ENZYME se tornou a escolha padrão para testes unitários de componentes da interface de usuário do React. Ao contrário de muitos outros utilitários de teste baseados em snapshot, Enzyme permite que você teste sem fazer renderização no dispositivo, o que resulta em testes mais rápidos e mais granulares. Esse é um fator que contribui para a nossa capacidade de reduzir massivamente a quantidade de testes funcionais que temos que fazer em aplicações React. Em muitos dos nossos projetos, ele é usado em um framework de testes de unidade, como Jest.

KOTLIN passou por uma taxa acelerada de adoção e crescimento rápido do suporte de ferramentas. Algumas das razões por trás de sua popularidade são sua sintaxe concisa, segurança contra nulidade, facilidade de transição de Java e interoperabilidade com outras linguagens baseadas em JVM em geral, além do fato de também funcionar como uma ótima linguagem introdutória à programação funcional. Com o JetBrains adicionando a capacidade de compilar Kotlin para binários nativos em múltiplas plataformas, bem como transpile para JavaScript, acreditamos que ela tenha um potencial de uso muito mais amplo pela grande comunidade de desenvolvimento de aplicações móveis e nativas. Embora no momento em que esse texto foi escrito algumas das ferramentas, como análise estática e de código de cobertura, ainda não tenham amadurecido, dada a nossa experiência com o uso de Kotlin em muitas aplicações de produção, acreditamos que esteja pronta para adoção geral.

Desde que foi incluído pela primeira vez no Radar, abservamos uma adoção consistente de GraphQL, principalmente como interface remota para a técnica Back-end para Front-ends (BFF). À medida que ganharam mais experiência, nossos times chegaram a um consenso sobre o Apollo, um cliente GraphQL, como a maneira preferida de acessar dados do GraphQL de uma aplicação React. Embora o projeto **APOLLO** também forneça uma estrutura de servidor e um gateway GraphQL, o cliente Apollo simplifica o problema de vincular componentes da interface de usuário a dados servidos por qualquer back-end do GraphQL. De forma notável, o Apollo é usado pela Amazon AWS em seu recente lançamento do novo serviço AWS AppSync.

O projeto Hyperledger tornou-se uma colaboração mais ampla e agora contém uma série de subprojetos. Ele suporta implementações Blockchain para diferentes propósitos; por exemplo, Burrow se dedica a criar um Ethereum com permissões, enquanto Indy tem maior foco na identidade digital. Entre essas plataformas, Fabric é a mais madura. Na maioria das vezes, quando as pessoas falam sobre a adoção de Hyperledger, elas estão na verdade pensando em Hyperledger Fabric. No entanto, a abstração de programação de chaincode é relativamente baixa, pois manipula diretamente o estado do registro. Além disso, sempre leva muito tempo para configurar a infraestrutura antes de escrever a primeira linha do código de blockchain. **HYPERLEDGER COMPOSER**, que se baseia em Fabric, acelera o processo de transformar ideias em software. O Composer fornece DSLs para modelar ativos de negócios, definir o controle de acesso e construir uma rede de negócios. Ao usar o Composer, você pode validar rapidamente sua ideia por meio de um navegador sem configurar qualquer infraestrutura. Basta lembrar que o Compositor em si não é Blockchain — você ainda precisa implantá-lo em Fabric.

A segurança é a base da economia de blockchain.
(OpenZeppelin)

A segurança é a base da economia de blockchain. Na última edição do Radar, destacamos a importância de testar e auditar dependências de contratos inteligentes. **OPENZEPPELIN** é um framework para ajudar a criar contratos inteligentes seguros em Solidity. A equipe por trás do OpenZeppelin resumiu uma série de armadilhas e melhores práticas em torno de segurança de contratos inteligentes e incorporou essas experiências no código-fonte. A estrutura é bem revisada e validada pela comunidade de código aberto. Recomendamos o uso do OpenZeppelin ao invés de escrever sua própria implementação do token ERC20/ERC721. O OpenZeppelin também se integra com Truffle.

FLUTTER é um framework multiplataforma que permite escrever aplicativos móveis nativos em Dart. Ele se beneficia do Dart e pode ser compilado em código nativo, se comunicando com a plataforma de destino sem ponte e alternância de contexto — algo que pode causar gargalos de desempenho em estruturas como React Native ou Weex. O recurso de recarga do Flutter é impressionante e fornece feedback visual super rápido ao editar o código. No momento, o Flutter ainda está em versão beta, mas continuaremos de olho nele para ver como seu ecossistema amadurece.

Dado o número de frameworks de aplicações JavaScript que incluímos no Radar ao longo dos anos, nos perguntamos: precisamos realmente trazer outro? Decidimos que **HYPERAPP** merece uma atenção devido à sua abordagem minimalista. Tem um footprint muito pequeno, menos de 1 KB, e ainda cobre todas as funcionalidades essenciais para se escrever uma aplicação web. Isso só é possível com um design elegante que reduz tudo ao mínimo absoluto, o que facilita a compreensão e o uso do framework. Apesar de ser relativamente novo, atraiu uma comunidade de bom tamanho e recomendamos pelo menos considerá-lo ao escolher um framework para uma nova aplicação.

RASA é um novo estreante na área de chatbots. Em vez de usar uma árvore de decisão simples, ele usa redes neurais para mapear intenção e estado interno para uma resposta. Rasa se integra com soluções de processamento de linguagem natural, como spaCy e, ao contrário de outras soluções que apresentamos no Radar, Rasa é um software de código aberto e pode ser auto-hospedado, o que o torna uma solução viável quando o controle dos dados é uma preocupação. Nossas experiências com o uso de Rasa Stack para uma aplicação interna foram positivas.

REACTOR é uma biblioteca para criar aplicações sem bloqueio em JVM — versões 8 e superiores — com base na especificação de Reactive Streams. A programação reativa enfatiza a mudança da lógica imperativa para o código de estilo assíncrono, não-bloqueador e funcional, especialmente quando se lida com recursos externos. O Reactor implementa a

especificação de fluxo reativo e fornece duas APIs de publicação — Flux (elementos 0 a N) e Mono (elemento 0 ou 1) — para modelar efetivamente o processamento de fluxo baseado em push. O projeto do Reactor é bem adequado para a arquitetura de microsserviços e oferece mecanismos de rede prontos para para tráfego HTTP, WebSockets, TCP e UDP.

RIBS — que é uma sigla para roteador, interator e compilador (builder) — é um framework de arquitetura multiplataforma móvel da Uber. A ideia principal do RIBs é dissociar a lógica de negócios da árvore de visões e assim garantir que o aplicativo seja orientado pela lógica de negócios. Esta é de fato uma aplicação de Arquitetura Limpa no desenvolvimento de aplicações móveis. Ao aplicar padrões de arquitetura consistentes em Android e iOS nativos, RIBs fornece gerenciamento claro de instruções e boa testabilidade. Aconselhamos colocar a lógica de negócios no serviço de back-end em vez de vazá-la na exibição, portanto, se você tiver realmente uma aplicação móvel complicada, RIBs pode ajudar a gerenciar essa complexidade.

Somos a favor de estilos de programação reativos e assíncronos, principalmente para sistemas distribuídos com limite de E/S de rede. Bibliotecas reativas frequentemente se encontram em uma estrutura de comunicação não-bloqueante de nível mais baixo, como Netty. Recentemente o **SWIFTNIO**, um framework de rede de código aberto da Apple, chamou nossa atenção. O SwiftNIO é semelhante ao Netty, mas escrito em Swift. No momento, é compatível com macOS e Ubuntu e implementa HTTP como um protocolo de nível superior. Estamos felizes em ver o uso desse framework e sua integração em frameworks de aplicações de nível superior e outros protocolos.

Na última edição, apresentamos o PyTorch, um framework de modelagem de aprendizagem profunda que permite um estilo de programação imperativo. Agora **TENSORFLOW EAGER EXECUTION** fornece esse estilo imperativo em TensorFlow, habilitando a execução de instruções de modelagem fora do contexto de uma sessão. Essa melhoria pode proporcionar a facilidade de depuração e controle de modelo mais refinado do PyTorch com a ampla popularidade e desempenho dos modelos do TensorFlow. O recurso ainda é novo, por isso ansiamos para observar seu desempenho e como ele será recebido pela comunidade do TensorFlow.

TENSORFLOW LITE é o sucessor escolhido do TensorFlow Mobile, que mencionamos em nosso Radar anterior. Assim como o Mobile, é uma solução leve ajustada e otimizada para dispositivos móveis (Android e iOS). Esperamos que o caso de uso padrão seja a implantação de modelos pré-treinados em aplicativos móveis, mas o TensorFlow Lite também suporta aprendizagem no dispositivo, o que abre outras áreas de aplicação.

Estamos testando **TROPOSPHERE** como forma de definir infraestrutura como código na AWS para nossos projetos nos quais AWS CloudFormation é usada no lugar de Terraform. troposphere é uma biblioteca Python que nos permite escrever código em Python para gerar descrições JSON em CloudFormation. O que mais gostamos em troposphere é que ela facilita a detecção rápida de erros JSON, aplicando verificação de tipo, testes de unidade e composição DRY de recursos da AWS.

WebAssembly é um formato de compilação binária projetado para ser executado no navegador em velocidades quase nativas. Ele amplia a variedade de linguagens que você pode usar para escrever funcionalidades front-end.

(WebAssembly)

WEBASSEMBLY é um grande avanço em termos de recursos de navegador como ambiente de execução de código. Suportado por todos os principais navegadores e compatível com versões anteriores, é um formato de compilação binária projetado para ser executado no navegador em velocidades quase nativas. Ele amplia a variedade de linguagens que você pode usar para escrever funcionalidades front-end, com foco inicial em C, C ++ e Rust, e também é um destino de compilação LLVM. Quando executado na sandbox, ele pode interagir com JavaScript e compartilhar as mesmas permissões e modelo de segurança. Quando usado com o novo compilador de streaming do Firefox, também resulta em inicialização de página mais rápida. Embora ainda seja cedo, este padrão W3C é definitivamente um padrão a se explorar.

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

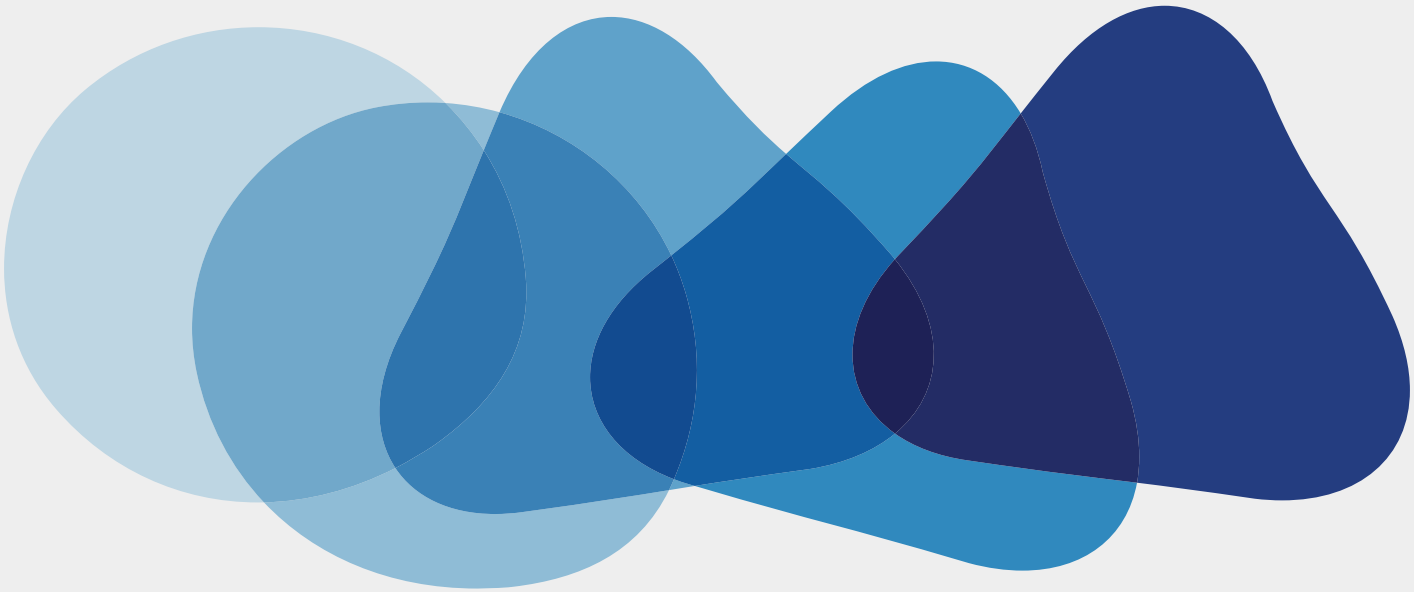
INSCREVA-SE

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 14 países: Alemanha, Austrália, Brasil, Canadá, Chile, China, Equador, Espanha, Estados Unidos, Índia, Itália, Reino Unido, Singapura e Tailândia.



thoughtworks.com/pt

#TWTechRadar