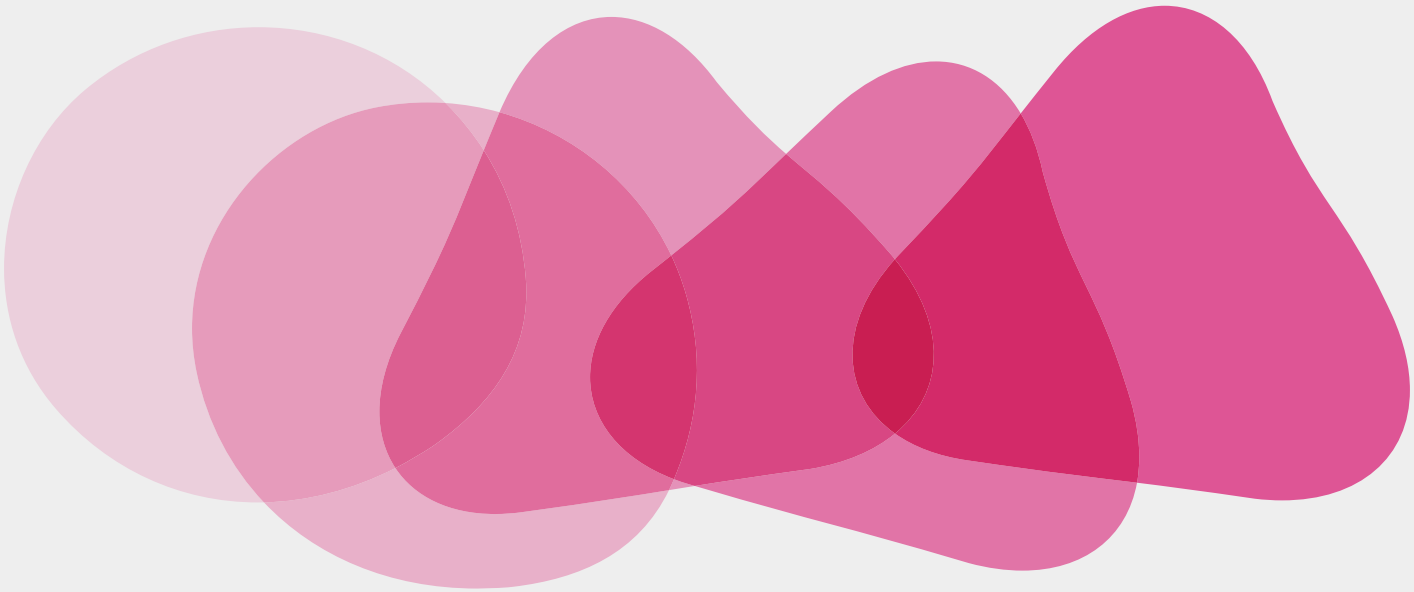
ThoughtWorks®

TECHNOLOGY RADAR *VOL.19*

Nossas ideias sobre tecnologias e tendências que estão moldando o futuro



thoughtworks.com/pt/radar

#TWTechRadar

CONTRIBUIÇÕES

O Technology Radar é produzido 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 | Ni Wang | Rachel Laycock | Scott Shaw | Shangqi Liu | Zhamak Dehghani

Tradução: Alexey Villas Bôas, Camilla Crispim, Marco Valtas, Paula Ribas e Ricardo Cavalcanti

Esta edição do Technology Radar da ThoughtWorks teve como base um encontro do Conselho Consultivo de Tecnologia, que se reuniu em Atlanta em outubro de 2018

O QUE HÁ DE NOVO?

Estes são os temas em destaque nessa edição:

NUVENS GRUDENTAS

Os provedores de nuvem sabem que estão em um mercado altamente competitivo e que, para terem sucesso, precisam adquirir e reter clientes de longo prazo. Assim, para se manterem diferenciados, correm para adicionar novas funcionalidades, e os vemos atingindo uma paridade de recursos, o que se reflete no fato de colocarmos AWS, Google Cloud Platform e Azure no anel *Experimente* nesta edição. No entanto, uma vez que conquistam os clientes, esses provedores tendem a criar uma relação o mais "grudenta" possível com seus clientes para desencorajar a migração para outro provedor. Frequentemente isso se manifesta em uma forte dependência da sua suíte específica de serviços e ferramentas, oferecendo uma melhor experiência de desenvolvimento, desde que os clientes permaneçam com eles. Algumas empresas são surpreendidas quando esse grude se torna aparente, geralmente ao optar por mover partes ou todas as suas cargas de trabalho para outra nuvem, ou ao descobrir que o uso da nuvem e seu custo estão fora de controle. Encorajamos nossos clientes a usar a técnica de custo de execução como função de aptidão arquitetural para monitorar o custo de operação, que é um indicador de dependência de determinado provedor, ou Kubernetes e contêineres para aumentar a portabilidade da carga de trabalho e reduzir o custo de mudança para outra nuvem por meio de infraestrutura como código. Nesta edição do Radar, também apresentamos duas novas ferramentas de automação de infraestrutura em nuvem, Terragrunt e Pulumi. Apesar de apoiarmos que se considere dependência ao avaliar novas ofertas do seu provedor de nuvem, advertimos contra o uso genérico da nuvem. Na nossa experiência, o custo de criar e manter camadas de abstração agnósticas de um provedor específico de nuvem supera o custo de saída de um provedor específico.

ANTIPADRÕES CORPORATIVOS PERSISTENTES

Não importa a velocidade de mudança da tecnologia, as empresas ainda encontram maneiras de reimplementar antipadrões do passado. Muitos de nossos registros no anel *Evite* denunciam um lobo velho disfarçado de ovelha nova: comportamento de Enterprise Service Bus (ESB) implementado em plataformas de fluxo de eventos — Recriando antipadrões ESB com Kafka, Arquitetura de microsserviços em camadas, Pacotes com fome de dados, Gateways de API excessivamente ambiciosos, Plataformas de baixo código e outras práticas antigas nocivas. O problema fundamental, como sempre, é o equilíbrio entre isolamento e acoplamento: isolamos as coisas para torná-las gerenciáveis a partir de uma perspectiva técnica, mas posteriormente precisamos adicionar coordenação para torná-las úteis na resolução de problemas de negócios, resultando em alguma forma de acoplamento. Dessa forma, esses velhos padrões continuam reemergindo. Novas arquiteturas e ferramentas fornecem meios apropriados para resolver esses problemas, mas isso requer um esforço deliberado para entender como usá-las apropriadamente, e não apenas voltar a reimplementar padrões antigos com novas tecnologias.

O QUE HÁ DE NOVO?

Estes são os temas em destaque nessa edição:

PRÁTICAS PERENES DE ENGENHARIA

Um efeito colateral do ritmo acelerado da inovação tecnológica é um padrão repetitivo de expansão e contração. Quando surge uma nova inovação que muda fundamentalmente a maneira como pensamos sobre algum aspecto do desenvolvimento de software, a indústria corre para adotar: conteinerização, front-ends reativos, aprendizagem de máquina e assim por diante. Essa é a fase de expansão. No entanto, para tornar esse elemento novo realmente eficaz, é necessário descobrir como aplicar práticas de engenharia perenes: entrega contínua, testes, colaboração e assim por diante. A fase de contração ocorre quando determinamos como usar esse novo recurso de forma eficaz, criando uma base firme para permitir a próxima expansão explosiva. Durante essa fase, aprendemos a aplicar práticas como testes automatizados abrangentes e automatização de sequências de etapas recorrentes dentro do contexto da nova tecnologia. Frequentemente, isso caminha de mãos dadas com a criação de novas ferramentas de desenvolvimento. Embora possa parecer que a introdução de uma nova inovação tecnológica por si só avança nossa indústria, é a combinação dessa inovação com práticas de engenharia perenes que sustenta nosso progresso contínuo.

RITMO = DISTÂNCIA / TEMPO

Nossos temas geralmente destacam um padrão que observamos em várias entradas no Radar atual, mas este se refere a todas as entradas durante a vida útil do Radar. Percebemos (e fizemos algumas pesquisas para comprovar) que o tempo que nossos blips permanecem no Radar está caindo. Quando começamos o Radar uma década atrás, o padrão para as entradas era permanecer por duas edições do Radar (aproximadamente um ano) sem nenhum movimento antes que elas desaparecessem automaticamente. No entanto, como indicado pela fórmula no título deste tema, *ritmo = distância sobre tempo*: a mudança no ecossistema de desenvolvimento de software continua a acelerar. O tempo permanece constante (ainda criamos o Radar duas vezes por ano), mas a distância percorrida em termos de inovação tecnológica aumentou consideravelmente, fornecendo ainda mais evidências ao que é óbvio para qualquer pessoa observadora: o ritmo da mudança tecnológica continua a aumentar. Vemos o aumento do ritmo em todos os nossos quadrantes do Radar e também no apetite de nossos clientes para adotar novas e diversas opções tecnológicas. Consequentemente, modificamos nosso padrão tradicional para este Radar: agora, cada entrada deve aparecer no Radar com base em seu mérito atual—não permitimos mais que elas permaneçam por padrão. Fizemos essa mudança após uma análise cuidadosa, sentindo que isso nos permite capturar melhor o ritmo frenético de mudança sempre presente no ecossistema de tecnologia.

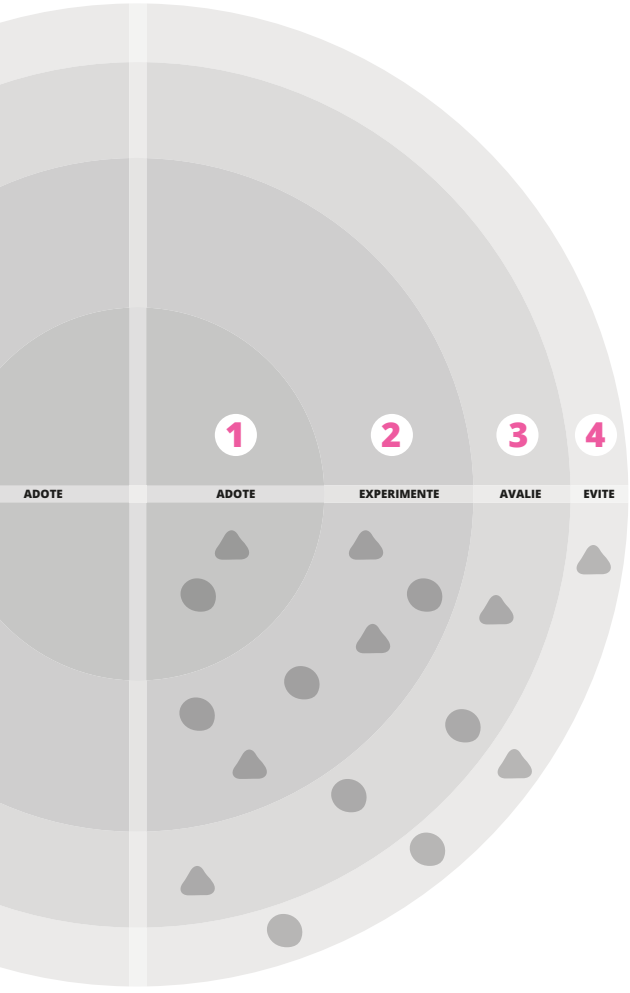
SOBRE O RADAR

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 software. 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 ser 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. Event Storming

EXPERIMENTE
2. 1% canário *NOVO*
3. Compra Delimitada *NOVO*
4. Destruição criptografada *NOVO*
5. Quatro métricas fundamentais *NOVO*
6. Configuração de nuvem para múltiplas contas *NOVO*
7. Observabilidade como código *NOVO*
8. Estratégia de fornecedor com risco proporcional *NOVO*
9. Custo de execução como função de aptidão arquitetural *NOVO*
10. Segredos como serviço *NOVO*
11. Engenharia de Caos de Segurança
12. Dados versionados para análises reproduzíveis *NOVO*

AVALIE
13. Katas de Caos *NOVO*
14. Imagens Docker sem distribuição *NOVO*
15. Entrega incremental com COTS *NOVO*
16. Analisador automatizado da configuração de infraestrutura
17. Verificações de pré-commit de compilação downstream *NOVO*
18. Malha de serviços

EVITE
19. Inicialização manual de clusters Hadoop usando ferramentas de gerenciamento de configuração *NOVO*
20. Uso genérico da nuvem
21. Arquitetura de microsserviços em camadas *NOVO*
22. Gerenciamento de dados mestre *NOVO*
23. Inveja de microsserviços
24. Eventos de solicitação-resposta em fluxos de trabalho voltados ao usuário *NOVO*
25. RPA *NOVO*

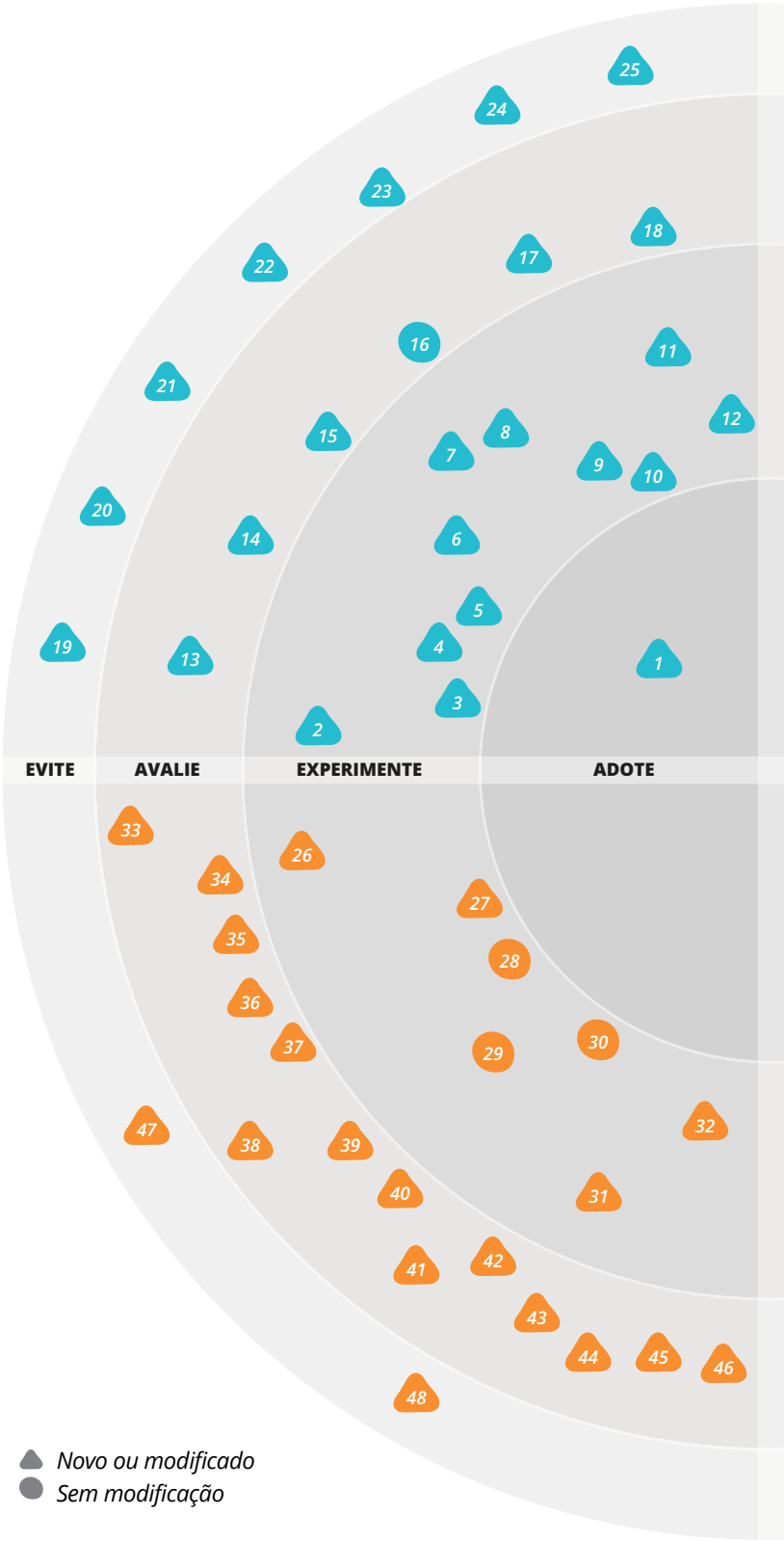
PLATAFORMAS

ADOTE

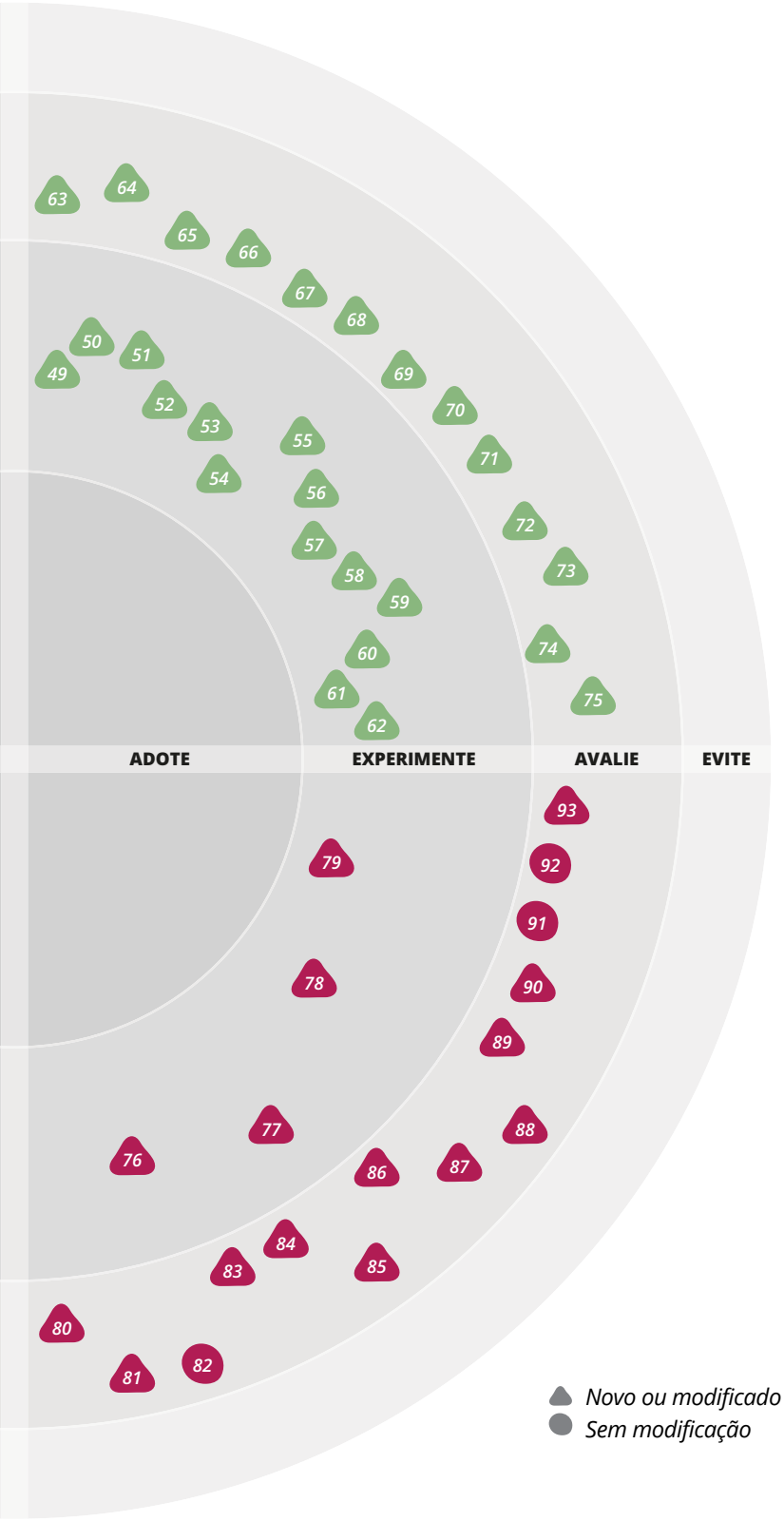
EXPERIMENTE
26. Apache Atlas *NOVO*
27. AWS
28. Azure
29. Contentful
30. Google Cloud Platform
31. VPC compartilhada *NOVO*
32. TICK Stack

AVALIE
33. Azure DevOps *NOVO*
34. CockroachDB *NOVO*
35. Debezium *NOVO*
36. Glitch *NOVO*
37. Google Cloud Dataflow *NOVO*
38. gVisor *NOVO*
39. IPFS *NOVO*
40. Istio *NOVO*
41. Knative *NOVO*
42. Pulumi *NOVO*
43. Quorum *NOVO*
44. Resin.io *NOVO*
45. Rook *NOVO*
46. SPIFFE *NOVO*

EVITE
47. Pacotes com fome de dados *NOVO*
48. Plataformas de baixo código *NOVO*



O RADAR



FERRAMENTAS

ADOTE

EXPERIMENTE
49. acs-engine *NOVO*
50. Archery *NOVO*
51. ArchUnit
52. Cypress
53. git-secrets *NOVO*
54. Headless Firefox
55. LocalStack *NOVO*
56. Mermaid *NOVO*
57. Prettier *NOVO*
58. Rider *NOVO*
59. Snyk *NOVO*
60. Ambientes de desenvolvimento de UI *NOVO*
61. Visual Studio Code
62. VS Live Share *NOVO*

AVALIE
63. Bitrise *NOVO*
64. Codefresh *NOVO*
65. Grafeas *NOVO*
66. Heptio Ark *NOVO*
67. Jaeger *NOVO*
68. kube-bench *NOVO*
69. Ocelot *NOVO*
70. Optimal Workshop *NOVO*
71. Stanford CoreNLP *NOVO*
72. Terragrunt *NOVO*
73. TestCafe *NOVO*
74. Traefik *NOVO*
75. Wallaby.js *NOVO*

EVITE

LINGUAGENS & FRAMEWORKS

ADOTE

EXPERIMENTE
76. Jepsen
77. MMKV *NOVO*
78. MockK *NOVO*
79. TypeScript

AVALIE
80. Apache Beam *NOVO*
81. Camunda *NOVO*
82. Flutter
83. Ktor *NOVO*
84. Nameko *NOVO*
85. Polly.js *NOVO*
86. PredictionIO *NOVO*
87. Puppeteer *NOVO*
88. Q# *NOVO*
89. SAFE stack *NOVO*
90. Spek *NOVO*
91. troposphere
92. WebAssembly
93. WebFlux *NOVO*

EVITE

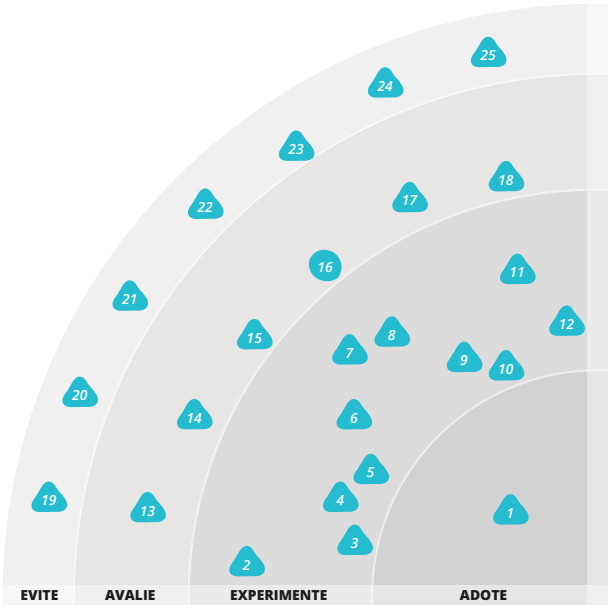
TÉCNICAS

ADOTE
1. Event Storming

EXPERIMENTE
2. 1% canário *NOVO*
3. Compra Delimitada *NOVO*
4. Destruição criptografada *NOVO*
5. Quatro métricas fundamentais *NOVO*
6. Configuração de nuvem para múltiplas contas *NOVO*
7. Observabilidade como código *NOVO*
8. Estratégia de fornecedor com risco proporcional *NOVO*
9. Custo de execução como função de aptidão arquitetural *NOVO*
10. Segredos como serviço *NOVO*
11. Engenharia de Caos de Segurança
12. Dados versionados para análises reproduzíveis *NOVO*

AVALIE
13. Katas de Caos *NOVO*
14. Imagens Docker sem distribuição *NOVO*
15. Entrega incremental com COTS *NOVO*
16. Analisador automatizado da configuração de infraestrutura
17. Verificações de pré-commit de compilação downstream *NOVO*
18. Malha de serviços

EVITE
19. Inicialização manual de clusters Hadoop usando ferramentas de gerenciamento de configuração *NOVO*
20. Uso genérico da nuvem
21. Arquitetura de microsserviços em camadas *NOVO*
22. Gerenciamento de dados mestre *NOVO*
23. Inveja de microsserviços
24. Eventos de solicitação-resposta em fluxos de trabalho voltados ao usuário *NOVO*
25. RPA *NOVO*



Quando as organizações migram para os microsserviços, um dos principais impulsionadores é a esperança de um tempo de ida a mercado mais rápido. No entanto, essa aspiração só tende a acontecer quando os serviços (e seus times de apoio) são distribuídos de forma homogênea entre os limites do domínio de negócios de longa duração. Caso contrário, funcionalidades significativas naturalmente exigirão uma coordenação rigorosa entre vários times e serviços, introduzindo atrito natural na priorização de roadmaps concorrentes. A solução para esse problema é uma boa modelagem de domínio. **EVENT STORMING** rapidamente se tornou um dos nossos métodos favoritos para identificar com rapidez os principais conceitos em um espaço de problema e alinhar várias partes interessadas da melhor maneira para distribuir soluções potenciais.

O feedback rápido é um dos nossos valores fundamentais para construir software. Por muitos anos, usamos a abordagem de implantação canário para incentivar o feedback rápido de novas versões de software, reduzindo o risco por meio de lançamentos incrementais para usuários selecionados. Uma das questões relacionadas a essa técnica é como segmentar usuários. Implantações canário para um segmento muito pequeno (por exemplo, 1%) de usuários podem ser um catalisador para mudanças. Enquanto começar com um segmento muito pequeno de usuários permite que os times se sintam confortáveis com a técnica, a captação de feedback rápido do usuário permite que diversos times observem o impacto de novas implantações, aprendam e ajustem o curso conforme necessário – uma mudança inestimável na cultura de engenharia. Nós chamamos isso de poder do **1% CANÁRIO**.

A maioria das organizações que não possuem recursos para criar software personalizado escolhe soluções prontas para uso ou SaaS para atender a seus requisitos. No entanto, frequentemente essas soluções tendem a expandir agressivamente seu escopo para se envolver em todas as partes do seu negócio. Isso ofusca os limites de integração e torna a mudança menos previsível e lenta. Para mitigar esse risco, recomendamos que as organizações desenvolvam um modelo objetivo de priorização de recursos e em seguida implantem uma estratégia que chamamos de **COMPRA DELIMITADA**, ou seja, escolher apenas produtos de fornecedores que sejam modulares e desacoplados e possam ficar

contidos no Contexto Delimitado de um único recurso do negócio. Essa modularidade e capacidade de entrega independente deve ser incluída nos critérios de aceitação de um processo de escolha de fornecedores.

Manter o controle adequado sobre dados confidenciais é difícil, especialmente quando – para fins de backup e recuperação – os dados são copiados fora de um sistema mestre de registro. A **DESTRUIÇÃO CRIPTOGRAFADA** é a prática de tornar os dados confidenciais ilegíveis, sobrescrevendo ou excluindo deliberadamente as chaves de criptografia usadas para proteger esses dados. Por exemplo, uma tabela inteira de detalhes pessoais do cliente pode ser criptografada usando chaves aleatórias para cada registro, com uma tabela diferente armazenando as chaves. Se um cliente exercitou seu "direito de ser esquecido", podemos simplesmente excluir a chave apropriada, efetivamente "destruindo" os dados criptografados. Essa técnica pode ser útil quando temos confiança em manter o controle adequado de um conjunto menor de chaves de criptografia, mas temos menos confiança em relação ao controle sobre um conjunto de dados maior.

O relatório State of DevOps e o livro subsequente Accelerate destacam um fato surpreendente: para prever e melhorar o desempenho de um time, precisamos apenas medir lead time, frequência de implantação, tempo médio de restauração (MTTR) e alterar a porcentagem de falhas.

(Quatro métricas fundamentais)

O relatório State of DevOps, publicado pela primeira vez em 2014, afirma que times de alto desempenho criam organizações de alto desempenho. Recentemente, o time por trás do relatório publicou Accelerate, que descreve o método científico usado no relatório. Uma das principais conclusões de ambos os materiais são as **QUATRO MÉTRICAS FUNDAMENTAIS** para sustentar o desempenho de entrega de software: lead time, frequência de implantação, tempo médio de restauração (MTTR) e porcentagem de falha de alteração. Como uma consultoria que ajudou na transformação de muitas organizações, vemos essas métricas surgirem frequentemente como uma forma de ajudar as organizações a determinar se estão melhorando o desempenho geral. Cada métrica cria um ciclo virtuoso e direciona o foco dos times à melhoria contínua: para reduzir o lead time, você reduz as atividades desnecessárias que, por sua vez, permitem implantar com mais frequência; a frequência de implantação força seus times a melhorar suas práticas e automação; sua velocidade para se recuperar de falhas é aprimorada por melhores práticas, automação e monitoramento, que reduzem a frequência de falhas.

O autoatendimento sob demanda é uma característica fundamental (e benéfica) da computação em nuvem. Quando cenários de serviços de larga escala são implantados usando uma única conta, regras e processos relacionados ao uso dessa conta se tornam necessários, geralmente envolvendo etapas de aprovação que aumentam o tempo de resposta. Uma abordagem melhor é a de **CONFIGURAÇÃO DE NUVEM PARA MÚLTIPLAS CONTAS**, em que várias contas são usadas; em casos exgtremos uma conta por equipe. Isso de fato adiciona custos em outros lugares, por exemplo, para garantir o faturamento compartilhado, permitir a comunicação entre VPCs e gerenciar o relacionamento com o provedor de nuvem. No entanto, isso geralmente acelera o desenvolvimento e melhora a segurança, porque as contas de serviço único são mais fáceis de auditar e, no caso de uma violação, o impacto é bastante reduzido. Ter várias contas também reduz a dependência de provedor de novem, porque uma conta fornece um bom limite para serviços que podem ser movidos em bloco para outro provedor de nuvem.

A observabilidade é parte integrante da operação de uma arquitetura de microsserviços distribuída. Dependemos de diferentes saídas do sistema, como rastreamento distribuído, registros agregados e métricas para inferir o estado interno dos componentes distribuídos, diagnosticar onde estão os problemas e chegar à raiz da causa. Um aspecto importante de um ecossistema de observabilidade é o monitoramento – visualização e análise da saída do sistema – e o alerta quando condições inesperadas são detectadas. Tradicionalmente, a configuração de painéis de monitoramento e a configuração de alertas são feita por meio de sistemas de apontar e clicar baseados em GUI. Essa abordagem leva a configurações de painel não repetíveis, incapacidade de testar e ajustar continuamente os alertas para evitar a fadiga ou a perda de alertas importantes, além do distanciamento de boas práticas das organizações. É altamente recomendável tratar suas configurações do ecossistema de **OBSERVABILIDADE COMO CÓDIGO**, além de adotar infraestrutura como código para sua infraestrutura de monitoramento e alertas. Escolha produtos de observabilidade que suportam configuração

por meio de código controlado por versão e execução de APIs, ou comandos por meio de pipelines de CD de infraestrutura. A observabilidade como código é um aspecto muitas vezes esquecido da infraestrutura como código, e acreditamos que seja crucial o suficiente para ser destacada.

A observabilidade é parte integrante da operação de uma arquitetura distribuída e baseada em microsserviços. Recomendamos tratar suas configurações de ecossistema de observação como código..

(Observabilidade como código)

Muitas vezes, em um esforço para terceirizar o risco para seus fornecedores, as empresas buscam um único fornecedor para suas implementações de sistema mais críticas e arriscadas. Infelizmente, isso proporciona menos opções de solução e menos flexibilidade. Ao invés disso, as empresas devem procurar manter uma maior independência de fornecedores nos pontos onde a exposição ao risco do negócio é maior. Vemos uma **ESTRATÉGIA DE FORNECEDOR COM RISCO PROPORCIONAL** emergente que incentiva o investimento para manter a independência do fornecedor para sistemas de negócio altamente críticos. Funções de negócios menos críticas podem tirar proveito da entrega simplificada de uma solução nativa do fornecedor, porque permitem absorver mais facilmente o impacto da perda desse fornecedor. Esse trade-off tornou-se aparente à medida que os principais provedores de nuvem expandiram sua gama de ofertas de serviços. Por exemplo, usar o AWS Secret Management Service pode acelerar o desenvolvimento inicial e tem o benefício da integração do ecossistema, mas também adicionará mais inércia se você precisar migrar para um provedor de nuvem diferente do que teria se tivesse implementado, por exemplo, Vault.

Ainda vemos times que não monitoram o custo de execução de suas aplicações com a devida atenção, à medida que sua arquitetura de software ou uso evolui. Isso acontece principalmente quando se utiliza uma arquitetura sem servidor, que leva as pessoas desenvolvedoras a presumirem que os custos serão mais baixos, já que você não está pagando por ciclos de servidor não utilizados. No entanto, os principais provedores de nuvem são bastante experientes em definir seus modelos de cálculo de preço e funções sem servidor muito usadas, e embora sejam muito úteis para a iteração rápida, podem se tornar caros rapidamente quando comparados com servidores dedicados na nuvem (ou locais). Aconselhamos os times a estruturar o **CUSTO DE EXECUÇÃO COMO FUNÇÃO DE APTIDÃO ARQUITETURAL**, o que significa monitorar o custo de execução de seus serviços em relação ao valor entregue. Quando você observar desvios em relação ao que era esperado ou aceitável, discuta se é hora de evoluir sua arquitetura.

Durante muito tempo, alertamos as pessoas sobre a tentação de incluir segredos em seus repositórios de código-fonte. Anteriormente, recomendamos desacoplar gerenciamento de segredos do código-fonte. No entanto, agora estamos vendo um conjunto de boas ferramentas que oferecem **SEGREDOS COMO SERVIÇO**. Com essa abordagem, em vez de escrever segredos ou configurá-los como parte do ambiente, as aplicações os recuperam a partir de um processo separado. Ferramentas como Vault, da HashiCorp, permitem que você gerencie segredos separadamente da aplicação, e aplique políticas, como rotação frequente externamente.

Embora tenhamos apresentado novos blips nesta edição do Radar, achamos que vale a pena continuar destacando a utilidade da **ENGENHARIA DE CAOS DE SEGURANÇA**. Nós a movemos para o anel Experimente porque os times que usam essa técnica confiam que as políticas de segurança implementadas são robustas o suficiente para lidar com os modos comuns de falha de segurança. Ainda assim, tenha cuidado ao usar essa técnica - não queremos que nossos times se dessensibilizem para essas questões.

Os dados versionados para análises reproduzíveis em larga escala ou problemas de inteligência de máquina nos permitem reproduzir diferentes versões de análises feitas em diferentes conjuntos de dados e parâmetros, e isso tem imenso valor.

(Dados versionados para análises reproduzíveis)

Quando se trata de análise de dados em larga escala ou problemas relacionados à inteligência de máquina, a capacidade de reproduzir diferentes versões de análises feitas com diferentes conjuntos de dados e parâmetros

é extremamente valioso. Para obter uma análise reproduzível, tanto os dados quanto o modelo (incluindo escolha de algoritmo, parâmetros e hiperparâmetros) precisam ser versionados. Os **DADOS VERSIONADOS PARA ANÁLISES REPRODUZÍVEIS** são um problema relativamente mais complicado do que os modelos de versão, devido ao tamanho dos dados. Ferramentas como DVC ajudam na criação de versões de dados ao permitir que os usuários executem commit e push de arquivos de dados para um repositório de armazenamento remoto em nuvem, usando um fluxo de trabalho parecido com o do git. Isso facilita a obtenção de uma versão específica de dados para reproduzir uma análise.

KATAS DE CAOS é uma técnica que nossos times desenvolveram para treinar e aperfeiçoar pessoas engenheiras de infraestrutura e de plataforma. Eles combinam técnicas de Engenharia de Caos – que é a criação de falhas e interrupções em um ambiente controlado – com a abordagem sistemática de ensino e treinamento Kata. Aqui, Kata refere-se a padrões de código que acionam falhas controladas, permitindo que as pessoas engenheiras descubram o problema, recuperem-se da falha, realizem um postmortem e encontrem a causa raiz. A execução repetida dos Katas ajuda as pessoas engenheiras a internalizar suas novas habilidades.

Ao criar imagens do Docker para nossas aplicações, geralmente temos duas preocupações: a segurança e o tamanho da imagem. Tradicionalmente, usamos ferramentas de varredura de segurança de contêiner para detectar e corrigir vulnerabilidades e exposições comuns e pequenas distribuições como Alpine Linux, para endereçar o tamanho da imagem e o desempenho da distribuição. Neste edição do Radar, estamos felizes em abordar a segurança e o tamanho dos contêineres com uma nova técnica chamada **IMAGENS DOCKER SEM DISTRIBUIÇÃO**, desenvolvida pelo Google. Com essa técnica, o footprint da imagem é reduzido para a aplicação, seus recursos e dependências de tempo de execução de linguagem, sem a distribuição do sistema operacional. As vantagens dessa técnica incluem redução do ruído dos scanners de segurança, menor superfície de ataque à segurança, sobrecarga reduzida de vulnerabilidades de patch e tamanho de imagem ainda menor para melhor desempenho. O Google publicou um conjunto de imagens de contêiner sem distribuição para diferentes linguagens. Você pode criar imagens de aplicativos sem distribuição usando a ferramenta de criação do Google Bazel, que tem regras para criar contêineres sem distribuição ou simplesmente usar Dockerfiles em vários estágios. Observe que os contêineres sem distribuição, por padrão, não possuem um shell para depuração. No entanto, você pode encontrar facilmente versões de depuração de contêineres sem distribuição online, incluindo busybox shell.

Ao criar imagens do Docker para nossas aplicações, muitas vezes nos preocupamos com duas coisas: a segurança e o tamanho da imagem. Com essa técnica, o footprint da imagem é reduzido para o aplicativo, seus recursos e dependências de tempo de execução de linguagem, sem a distribuição do sistema operacional.

(Imagens Docker sem distribuição)

A ThoughtWorks, como pioneira e líder no espaço ágil, tem proposto a prática da entrega incremental. Também aconselhamos muitos clientes que examinem software pronto para uso por meio de uma abordagem "isso pode ser liberado de forma incremental?". Tem sido difícil devido à abordagem "big-bang" da maioria dos fornecedores, que geralmente envolve a migração de grandes quantidades de dados. Recentemente, no entanto, também obtivemos sucesso ao usar a **ENTREGA INCREMENTAL COM COTS** (Commercial off-the-shelf), lançando processos de negócios específicos de forma incremental para subconjuntos menores de usuários. Recomendamos que você avalie se pode aplicar essa prática ao software do fornecedor de sua escolha, para ajudar a reduzir os riscos envolvidos nas entregas big-bang.

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.

Em arquiteturas e implementações mais complexas, pode não ser imediatamente óbvio que uma compilação dependente do código atualmente sendo checked-in está quebrada. As pessoas desenvolvedoras que tentam consertar uma compilação quebrada podem se encontrar trabalhando contra um alvo em movimento, já que a compilação é continuamente acionada por dependências upstream. **VERIFICAÇÕES DE PRÉ-COMMIT DE COMPILAÇÃO DOWNSTREAM** é uma técnica muito simples: um script pré-commit ou pré-push verifica o status dessas compilações downstream e alerta a pessoa desenvolvedora com antecedência que uma compilação está quebrada.

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 um 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 em uma diferente instância de execução. 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.

Quando as organizações escolhem uma distribuição Hadoop vanilla ou Spark em vez de uma das distribuições de fornecedor, elas precisam decidir como querem provisionar e gerenciar o cluster. Ocasionalmente, vemos uma **INICIALIZAÇÃO MANUAL DE CLUSTERS HADOOP USANDO FERRAMENTAS DE GERENCIAMENTO DE CONFIGURAÇÃO** como Ansible, Chef e outras. Embora essas ferramentas sejam ótimas no provisionamento de componentes de infraestrutura imutáveis, elas não são muito úteis quando você precisa gerenciar sistemas dinâmicos e podem frequentemente levar a um esforço significativo para tentar gerenciar e desenvolver clusters. Em vez disso, recomendamos o uso de ferramentas como Ambari para provisionar e gerenciar seus clusters Hadoop ou Spark dinâmicos.

Cada vez mais, vemos organizações se preparando para usar "qualquer nuvem" e evitar o aprisionamento a fornecedores a todo custo. Essa estratégia limita o uso da nuvem aos únicos recursos comuns a todos os provedores de nuvem, abrindo mão dos benefícios exclusivos dos provedores.

(Uso genérico da nuvem)

Os principais provedores de nuvem tem se tornado cada vez mais competitivos em seus preços e no ritmo acelerado de lançamento de novos recursos. Isso deixa os consumidores em uma posição difícil ao escolher e se comprometer com um provedor. Cada vez mais, estamos vendo organizações se prepararem para usar "qualquer nuvem" e evitar o aprisionamento de fornecedor a todo custo. Isso, é claro, leva ao **USO GENÉRICO DA NUVEM**. Vemos organizações limitando seu uso da nuvem aos únicos recursos comuns em todos os provedores de nuvem – perdendo, assim, os benefícios exclusivos dos provedores. Vemos organizações fazendo grandes investimentos em camadas de abstração desenvolvidas localmente, que são muito complexas para serem construídas e caras demais para serem mantidas para permanecerem agnósticas de nuvem. O problema do aprisionamento é real. Recomendamos abordar esse problema com uma estratégia de várias nuvens que avalia o custo da migração e o esforço dos recursos de uma nuvem para outra, em comparação com os benefícios do uso de recursos específicos da nuvem. Recomendamos aumentar a portabilidade das cargas de trabalho enviando as aplicações como contêineres Docker amplamente adotados, usando protocolos de identidade e segurança de software livre para migrar facilmente a identidade das cargas de trabalho, através de uma estratégia de fornecedor com risco proporcional

para manter a independência da nuvem somente quando necessário, e usando Polycloud para misturar e combinar serviços de diferentes fornecedores onde isso faz sentido. Em resumo, mude sua abordagem de um uso genérico da nuvem para uma estratégia sensata de várias nuvens.

Uma das características que define uma arquitetura de microsserviços é que os componentes e serviços do sistema são organizados em torno de capacidades de negócio. Independentemente do tamanho, os microsserviços encapsulam um agrupamento significativo de funcionalidade e informação para permitir a entrega independente de valor de negócio. Isso contrasta com abordagens anteriores na arquitetura de serviços que os organizavam de acordo com características técnicas. Observamos uma série de empresas adotando uma **ARQUITETURA DE MICROSSERVIÇOS EM CAMADAS**, que, em alguns aspectos, é uma contradição. Essas empresas recorreram à organização de serviços principalmente de acordo com uma função técnica, por exemplo, APIs de experiência, APIs de processo ou APIs do sistema. É muito fácil atribuir times de tecnologia por camada, de modo que entregar qualquer alteração de negócio importante requer uma coordenação lenta e cara entre vários times. Nós nos prevenimos contra os efeitos dessa divisão por camadas e recomendamos a organização de serviços e times essencialmente de acordo com capacidades de negócio.

O **GERENCIAMENTO DE DADOS MESTRE** (MDM) é um exemplo clássico da solução empresarial "bala de prata": ela promete resolver uma classe de problemas aparentemente relacionados de uma só vez. Por meio da criação de um único ponto centralizado de mudança, coordenação, teste e implantação, as soluções de MDM afetam negativamente a capacidade de uma organização de responder às mudanças nos negócios. As implementações tendem a ser longas e complexas, pois as organizações tentam capturar e mapear todos os dados "mestre" no MDM, integrando a solução MDM em todos os sistemas consumidores ou produtores.

Os microsserviços surgiram como uma das principais técnicas de arquitetura em sistemas modernos baseados em nuvem, mas ainda acreditamos que os times devem proceder com cuidado ao fazer essa escolha. A **INVEJA DE MICROSSERVIÇOS** tenta os times a complicarem sua arquitetura com muitos serviços, simplesmente porque é uma opção de arquitetura moderna. Plataformas como o Kubernetes facilitam muito a implementação de conjuntos complexos de microsserviços, e os fornecedores estão empurrando suas soluções para gerenciamento de microsserviços, potencialmente influenciando times a seguirem nesse caminho. É importante lembrar que os microsserviços trocam a complexidade de desenvolvimento para complexidade operacional, e exigem uma base sólida de testes automatizados, entrega contínua e cultura de DevOps.

Em várias ocasiões, vimos designs de sistema que usam **EVENTOS DE SOLICITAÇÃO-RESPOSTA EM FLUXOS DE TRABALHO VOLTADOS AO USUÁRIO**. Nesses casos, a interface do usuário é bloqueada ou o usuário precisa aguardar o carregamento de uma nova página até que uma mensagem de resposta correspondente a uma mensagem de solicitação seja recebida. Os principais motivos citados para designs como este são o desempenho ou uma abordagem unificada para comunicação entre back-ends para casos de uso síncronos e assíncronos. Acreditamos que o aumento da complexidade – em desenvolvimento, testes e operações – supera em muito o benefício de ter uma abordagem unificada, e sugerimos fortemente o uso de solicitações HTTP síncronas quando a comunicação síncrona entre os serviços de back-end for necessária. Quando bem implementada, a comunicação usando HTTP raramente é um gargalo em um sistema distribuído.

A automação de processos robóticos (**RPA**) é uma parte essencial de muitas iniciativas de transformação digital, pois promete proporcionar redução de custos sem a necessidade de modernizar a arquitetura e os sistemas subjacentes. O problema com essa abordagem, que se concentrar apenas na automação de processos de negócios, sem abordar os sistemas ou recursos de software subjacentes, é que isso pode dificultar ainda mais a alteração dos sistemas subjacentes, introduzindo acoplamentos adicionais. Isso torna ainda mais difícil qualquer tentativa futura de endereçar o cenário de TI legado. Poucos sistemas podem se dar ao luxo de ignorar mudanças e, portanto, os esforços de RPA precisam ser associados a estratégias apropriadas de modernização de sistemas legados.

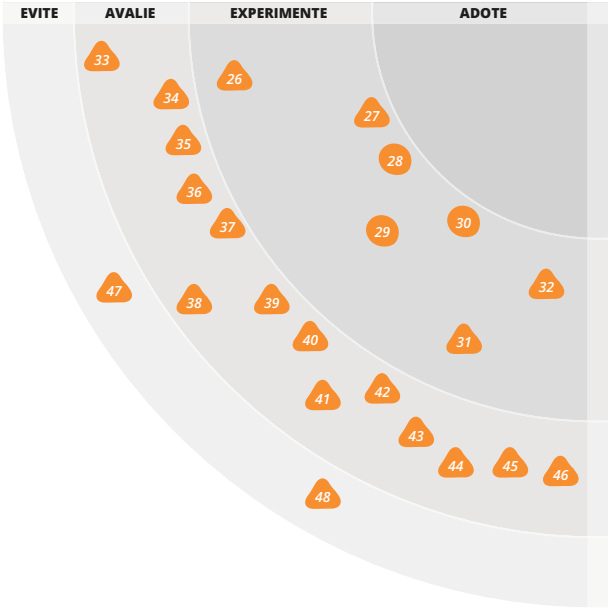
PLATAFORMAS

ADOTE

EXPERIMENTE
26. Apache Atlas *NOVO*
27. AWS
28. Azure
29. Contentful
30. Google Cloud Platform
31. VPC compartilhada *NOVO*
32. TICK Stack

AVALIE
33. Azure DevOps *NOVO*
34. CockroachDB *NOVO*
35. Debezium *NOVO*
36. Glitch *NOVO*
37. Google Cloud Dataflow *NOVO*
38. gVisor *NOVO*
39. IPFS *NOVO*
40. Istio *NOVO*
41. Knative *NOVO*
42. Pulumi *NOVO*
43. Quorum *NOVO*
44. Resin.io *NOVO*
45. Rook *NOVO*
46. SPIFFE *NOVO*

EVITE
47. Pacotes com fome de dados *NOVO*
48. Plataformas de baixo código *NOVO*



As necessidades de gerenciamento de metadados estão crescendo. O Atlas fornece a capacidade de modelar tipos de metadados, classificar ativos de dados, rastrear linhagem de dados e ativar a descoberta de dados. Isso se ajusta às necessidades de governança de dados das empresas.

(Apache Atlas)

Com as crescentes e diversas necessidades de dados das empresas, surge uma necessidade crescente de gerenciamento de metadados. **APACHE ATLAS** é uma estrutura de gerenciamento de metadados que atende às necessidades de governança de dados das empresas. O Atlas fornece recursos para modelar tipos de metadados, classificar ativos, rastrear a linhagem e habilitar a descoberta de dados. No entanto, ao criar uma plataforma de gerenciamento de metadados, precisamos ter cuidado para não repetir os erros do gerenciamento de dados mestre.

Colocamos a **AWS** inicialmente no anel Adote há sete anos, e a abrangência, a profundidade e a confiabilidade de seus serviços melhoraram em ritmo acelerado desde então. No entanto, estamos movendo a AWS agora para o anel Experimente, não por deficiências em sua oferta, mas porque seus concorrentes, GCP e Azure amadureceram consideravelmente e a escolha de um provedor de nuvem tornou-se cada vez mais complexa. Nós reservamos o Adote para quando vemos um favorito óbvio em um campo. Por muitos anos, a AWS foi a escolha padrão, mas agora sentimos que as organizações devem fazer uma escolha equilibrada entre provedores de nuvem, que leve em conta sua footprint geográfica e regulatória, seu alinhamento estratégico com provedores (ou a falta dele) e, é claro, a ligação entre suas necessidades mais importantes e os produtos que diferenciam os provedores de nuvem.

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.

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.

Um padrão recomendado é usar uma rede de nuvem privada virtual gerenciada no nível organizacional e dividida em sub-redes menores sob o controle de cada time de entrega. A VPC compartilhado torna organizações, projetos, VPCs e sub-redes entidades de primeira classe em configurações de rede – isso simplifica a configuração e torna a segurança e o controle de acesso mais transparentes.

(VPC compartilhada)

À medida que ganhamos mais experiência com a nuvem pública em organizações grandes e pequenas, surgiram alguns padrões. Um deles é uma rede de nuvem privada virtual gerenciada no nível organizacional e dividida em sub-redes menores sob o controle de cada time de entrega. Isso está intimamente relacionado à idéia de configuração de múltiplas contas na nuvem e ajuda a partição de infraestrutura alinhada com limites do time. Depois de realizar muitas vezes essa configuração explicitamente usando VPCs, sub-redes, grupos de segurança e NACLs, gostamos muito da noção de **VPC COMPARTILHADA** do Google. A VPC compartilhada transforma organizações, projetos e sub-redes em entidades de primeira classe em configurações de rede. As VPCs podem ser gerenciadas por administradores de uma organização, que podem delegar a administração de sub-redes aos projetos. Os projetos podem ser explicitamente associados a sub-redes na VPC. Isso simplifica a configuração e torna a segurança e o controle de acesso mais transparentes.

TICK STACK é uma plataforma composta por componentes de código aberto, que facilita a coleta, armazenamento, geração de gráficos e alertas de dados de séries temporais, 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, embora seja necessária a versão corporativa para recursos como clusterização de bancos de dados, ainda é uma boa opção para monitoramento. Estamos usando em alguns locais em produção e tivemos boas experiências.

O **AZURE DEVOPS** Services inclui um conjunto de serviços gerenciados, como repositórios Git hospedados, pipelines CI e CD e repositório de artefatos. O Azure DevOps Services substiuiu o Visual Studio Team Services. Tivemos uma boa experiência ao iniciar projetos rapidamente com os serviços do Azure DevOps – gerenciando, construindo e implantando aplicações para o Azure. Também nos deparamos com

alguns desafios – como a falta de suporte completo para pipelines de CI e CD como código, lentidão no tempo de inicialização de agente de compilação, separação de compilação e implantação em diferentes pipelines – e experimentamos algumas paralisações. Esperamos que o Azure DevOps Services melhore com o tempo para fornecer uma boa experiência de desenvolvimento ao hospedar aplicações no Azure, com uma experiência sem fricções na integração com outros serviços do Azure.

Inspirado no relatório do Google sobre Spanner – seu banco de dados distribuído baseado em relógios atômicos –, o CockroachDB é uma alternativa de código aberto que fornece transações distribuídas e particionamento geolocalizado, ainda suportando SQL.

(CockroachDB)

COCKROACHDB é um banco de dados distribuído de código aberto inspirado pelo relatório Spanner: o banco de dados distribuído do Google. No CockroachDB, os dados são automaticamente divididos em intervalos, normalmente de 64MB, e distribuídos entre nós no cluster. Cada intervalo tem um grupo de consenso e, pelo fato da plataforma usar o algoritmo de consenso Raft, os dados são sempre sincronizados. Com seu design único, o CockroachDB fornece transações distribuídas e particionamento geolocalizado, suportando ainda SQL. Ao contrário do Spanner, que se baseia em TrueTime com relógio atômico para linearizabilidade, o CockroachDB usa NTP para sincronização do relógio e fornece serialização como nível de isolamento padrão. Se você estiver trabalhando com dados estruturados que se encaixam em um único nó, escolha um banco de dados relacional tradicional. No entanto, se seus dados precisam ser escalados nos nós, ser consistentes e sobreviver a falhas, recomendamos que você analise com atenção o CockroachDB.

DEBEZIUM é uma plataforma de change data capture (CDC) que pode transmitir alterações de banco de dados como tópicos do Kafka. O CDC é uma técnica popular com vários usos, incluindo a replicação de dados para outros bancos, alimentação de sistemas analíticos, extração de microsserviços de monolitos e invalidação de caches. Estamos sempre à procura de ferramentas ou plataformas nessa área (falamos sobre Bottled Water em uma edição anterior) e Debezium é uma excelente escolha. A plataforma usa uma abordagem de CDC baseada em log, o que significa que ela reage a alterações nos arquivos de log do banco de dados. Debezium usa o Kafka Connect, que a torna altamente escalável e resiliente a falhas, e possui conectores CDC para vários bancos de dados, incluindo Postgres, Mysql e MongoDB. Estamos usando em alguns projetos e tem funcionado muito bem para nós.

GLITCH, um ambiente de desenvolvimento colaborativo online que permite copiar e adaptar (ou "remixar") aplicações web existentes ou criar as suas próprias aplicações com facilidade, despertou nosso interesse. Enraizado no ethos "tinkerer", Glitch é ideal para pessoas aprendendo a programar, mas é capaz de suportar aplicações mais complexas. O foco principal é em JavaScript e Node.js, mas há suporte, embora limitado, para outras linguagens. Com edição em tempo real integrada, hospedagem, compartilhamento e controle de versão de código de forma instantânea, Glitch oferece uma visão inovadora e diferenciada de programação colaborativa.

GOOGLE CLOUD DATAFLOW é útil em cenários ETL tradicionais para ler dados de uma fonte, transformando-os e, em seguida, armazenando-os em um dissipador, com configurações e dimensionamento gerenciados pelo Dataflow. O Dataflow suporta Java, Python e Scala e fornece empacotadores para conexões com vários tipos de fontes de dados. No entanto, a versão atual não permite adicionar outras bibliotecas, o que pode torná-lo inadequado para determinadas manipulações de dados. Não é possível também alterar o DAG do fluxo de dados dinamicamente. Portanto, se seu ETL tiver fluxos condicionais de execução com base em parâmetros, talvez você não consiga usar o fluxo de dados sem soluções alternativas.

GVISOR é um kernel do espaço de usuário para contêineres. Ele limita a superfície do kernel de hospedagem acessível à aplicação sem eliminar o acesso a todos os recursos esperados. Ao contrário das tecnologias de sandbox existentes, como hardware virtualizado (KVM e Xen) ou execução baseada em regras (seccomp, SELinux e AppArmor), o gVisor adota uma abordagem distinta para o sandboxing de contêineres ao interceptar chamadas de sistema da aplicação e atuar como kernel convidado sem a necessidade de tradução através de hardware virtualizado. O gVisor inclui uma Open Container Initiative (OCI) chamada runsc que se integra com

Docker e fornece suporte experimental a Kubernetes. O gVisor é um projeto relativamente novo e recomendamos que ele seja avaliado para o panorama de segurança do seu contêiner.

Na maioria dos casos, blockchain não é o lugar certo para armazenar um arquivo blob (imagem ou áudio, por exemplo). Quando as pessoas desenvolvem aDApp, uma opção é colocar arquivos blob em algum armazenamento de dados centralizado, o que geralmente sinaliza falta de confiança . Outra opção é armazená-los no InterPlanetary File System (**IPFS**), que é um sistema de arquivos content addressed, ponto-a-ponto e com versionamento. Ele foi projetado para distribuir grandes volumes de dados com alta eficiência e removidos de qualquer autoridade centralizada. Os arquivos são armazenados em pontos que não precisam confiar uns nos outros. O IPFS mantém todas as versões de um arquivo para que você nunca perca arquivos importantes. Nós vemos o IPFS como um bom complemento para a tecnologia blockchain. Além de sua aplicação com blockchain, o IPFS tem uma meta ambiciosa de descentralizar a infraestrutura da Internet.

Ao criar e operar um ecossistema de microsserviços, uma das primeiras perguntas a responder é como implementar funcionalidades transversais como descoberta de serviço, segurança de serviço-a-serviço e de origem-a-serviço, observabilidade (incluindo telemetria e rastreamento distribuído), rolling releases e resiliência. Nos últimos dois anos, nossa resposta padrão a essa pergunta foi usar uma técnica de malha de serviço. Uma malha de serviços oferece a implementação desses recursos transversais como uma camada de infraestrutura configurada por código. As configurações de políticas podem ser consistentemente aplicadas a todo o ecossistema de microsserviços; imposta tanto no tráfego interno à malha como externo (através do proxy de malha como um gateway), bem como no tráfego em cada serviço (através do mesmo proxy de malha como um contêiner de sidecar). Apesar de acompanharmos de perto o progresso de diferentes projetos de malhas de serviços de código aberto, como Linkerd, usamos com sucesso **ISTIO** em produção com um modelo operacional surpreendentemente fácil de configurar.

Como pessoas desenvolvedoras de aplicações, gostamos de nos concentrar na solução de problemas centrais de negócios, deixando que a plataforma subjacente lide com as tarefas chatas, porém difíceis, de implantar, escalar e gerenciar aplicações. Embora a arquitetura sem servidor seja um passo nessa direção, a maioria das ofertas populares está vinculada a uma implementação proprietária, o que significa dependência de fornecedores. **KNATIVE** tenta resolver isso por ser uma plataforma sem servidor de código aberto que se integra bem com o popular ecossistema Kubernetes. Com Knative você pode modelar cálculos sob demanda em uma estrutura suportada de sua escolha (incluindo Ruby on Rails, Django e Spring, entre outros); subscrever, entregar e gerenciar eventos; integrar com ferramentas conhecidas de CI e CD; e construir contêineres a partir da fonte. Fornecendo um conjunto de componentes de middleware para criar aplicações centradas na fonte e baseadas em contêiner, que podem ser escaladas elasticamente, Knative é uma plataforma atraente que merece ser avaliada para suas necessidades sem servidor.

Concentrando-se fortemente nas arquiteturas nativas da nuvem, Pulumi é uma ferramenta de automação de infraestrutura que se destaca ao permitir que as configurações sejam escritas em TypeScript/JavaScript, Python e Go.

(Pulumi)

PULUMI, um estreante promissor na automação de infraestrutura em nuvem, tem nos interessado muito. Pulumi se distingue ao permitir que as configurações sejam escritas em TypeScript/JavaScript, Python e Go — sem necessidade de YAML. O Pulumi é fortemente focado em arquiteturas nativas da nuvem — incluindo contêineres, funções sem servidor e serviços de dados — e fornece um bom suporte para Kubernetes.

Ethereum é o principal ecossistema de desenvolvimento de tecnologia blockchain. Temos soluções emergentes visando disseminar essa tecnologia em ambientes empresariais que normalmente exigem permissão de rede e privacidade de transação, além de maior taxa de transferência e menor latência. **QUORUM** é uma dessas soluções. Originalmente desenvolvida por J.P. Morgan, Quorum se posiciona como "uma versão do Ethereum com foco em empresas". Ao contrário do nó Hyperledger Burrow, que cria uma nova máquina virtual Ethereum (EVM), Quorum faz o fork do código a partir do cliente oficial do Ethereum, permitindo que ele evolua juntamente com o Ethereum. Embora mantenha a maioria das funcionalidades do ledger do Ethereum, Quorum altera o protocolo de consenso de

prova de trabalho (PoW) para outros mais eficientes, além de adicionar suporte a transações privadas. Com Quorum, pessoas desenvolvedoras podem aproveitar seu conhecimento da Ethereum usando, por exemplo, contratos Solidity e Truffle para construir aplicações blockchain empresariais. No entanto, com base em nossa experiência, o Quorum ainda não está pronto para empresas. Ele não tem, por exemplo, controle de acesso para contratos privados, não funciona bem com balanceadores de carga e tem apensas suporte parcial a bancos de dados, o que levará a uma sobrecarga significativa de implantação e design. Recomendamos que você tenha cautela ao implementar Quorum, com atenção ao progresso futuro dessa solução.

RESIN.IO é uma plataforma de Internet das Coisas (IoT) que faz uma coisa e faz bem: implanta contêineres em dispositivos. As pessoas desenvolvedoras usam um portal de software como serviço (SaaS) para gerenciar dispositivos e atribuir aplicações definidas por Dockerfiles a eles. A plataforma pode construir contêineres para vários tipos de hardware e implanta as imagens remotamente. Para os contêineres, o Resin.io usa balena, um mecanismo baseado no framework Moby criado por Docker. A plataforma ainda está em desenvolvimento, tem algumas arestas não aparadas e não possui alguns recursos (por exemplo, trabalhar com registros privados), mas o conjunto de recursos atual, incluindo conexão ssh em um contêiner em um dispositivo do portal da Web, aponta para um futuro promissor.

ROOK é um orquestrador de armazenamento em nuvem nativa de código aberto para Kubernetes. Rook se integra com Ceph e traz os sistemas de armazenamento File, Block e Object para o cluster do Kubernetes, executando-os perfeitamente bem junto a outras aplicações e serviços que estão consumindo o armazenamento. Ao usar os operadores do Kubernetes, Rook orquestra o Ceph no plano de controle e fica fora do caminho dos dados entre as aplicações e o Ceph. Armazenamento é um dos componentes importantes da computação em nuvem nativa e acreditamos que Rook, embora ainda seja um projeto em fase de incubação na CNCF, nos coloca um passo mais perto da auto-suficiência e portabilidade entre implantações local e na nuvem pública.

Tornar elementos-chave da plataforma inovadora e de grande escala do Google disponíveis como ofertas de código aberto parece ter se tornado uma tendência. Da mesma forma que o HBASE utilizou BigTable e o Kubernetes utilizou Borg, o **SPIFFE** agora está se baseando no LOAS do Google para dar vida a um conceito crítico de nuvem nativa chamado identidade de carga de trabalho. Os padrões SPIFFE são respaldados pelo OSS SPIFFE Runtime Environment (SPIRE), que fornece automaticamente identidades criptograficamente verificáveis para cargas de trabalho de software. Embora o SPIRE ainda não esteja pronto para uso em produção, vemos enorme valor em uma forma independente de plataforma de fazer fortes asserções de identidade entre cargas de trabalho em infraestruturas de TI distribuídas e modernas. SPIRE suporta muitos casos de uso, incluindo conversão de identidade, autenticação de cliente OAuth, "criptografia em todos os lugares" do mTLS e observabilidade da carga de trabalho. O Istio usa SPIFFE por padrão

PACOTES COM FOME DE DADOS são soluções que exigem a absorção de dados em si próprios para funcionar. Em alguns casos, eles podem até exigir que se tornem "mestres" destes dados. Depois que os dados passam a ser controlados pelo pacote, esse software se torna a única maneira de atualizar, alterar ou acessar os dados. O pacote que tem fome de dados pode resolver um problema comercial específico, como ERP. No entanto, as "demandas de dados" de estoque ou finanças que surgem em uma organização geralmente exigirão integração e alterações complexas em sistemas que estão fora do escopo original.

PLATAFORMAS DE BAIXO CÓDIGO usam interfaces de usuário e configuração gráficas para criar aplicações. Infelizmente, os ambientes de baixo código são promovidos com a ideia de que você não precisa mais de times de desenvolvimento com habilidades específicas. Tais sugestões ignoram o fato de que escrever código é apenas uma pequena parte do que precisa acontecer para criar software de alta qualidade – práticas como controle de versão, testes e design meticuloso de soluções são tão importantes quanto. Embora essas plataformas tenham seus usos, sugerimos abordá-las com cautela, especialmente quando elas são acompanhadas de alegações extravagantes de menor custo e maior produtividade.

FERRAMENTAS

ADOTE

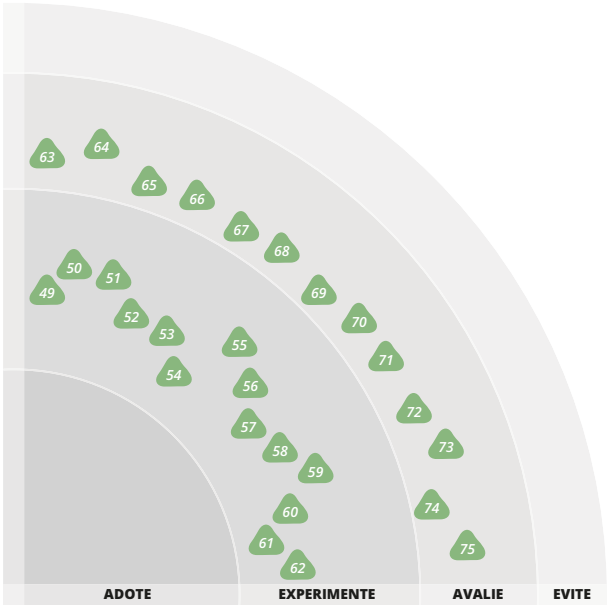
EXPERIMENTE

49. acs-engine *NOVO*
50. Archery *NOVO*
51. ArchUnit
52. Cypress
53. git-secrets *NOVO*
54. Headless Firefox
55. LocalStack *NOVO*
56. Mermaid *NOVO*
57. Prettier *NOVO*
58. Rider *NOVO*
59. Snyk *NOVO*
60. Ambientes de desenvolvimento de UI *NOVO*
61. Visual Studio Code
62. VS Live Share *NOVO*

AVALIE

63. Bitrise *NOVO*
64. Codefresh *NOVO*
65. Grafeas *NOVO*
66. Heptio Ark *NOVO*
67. Jaeger *NOVO*
68. kube-bench *NOVO*
69. Ocelot *NOVO*
70. Optimal Workshop *NOVO*
71. Stanford CoreNLP *NOVO*
72. Terragrunt *NOVO*
73. TestCafe *NOVO*
74. Traefik *NOVO*
75. Wallaby.js *NOVO*

EVITE



Azure Container Service Engine (**ACS-ENGINE**) é um gerador de templates para o Azure Resource Manager (ARM). As configurações necessárias do cluster são definidas em um arquivo JSON, o acs-engine lê essas definições de cluster e gera vários arquivos que podem ser consumido pelo ARM. A ferramenta também oferece flexibilidade para escolher diferentes orquestradores, incluindo Kubernetes, DC/OS, OpenShift, Swarm mode e Swarm – e para configurar recursos e agentes do cluster. Temos usado o acs-engine em diversos projetos e o recomendamos para gerenciamento de clusters no Azure Container Service.

Temos visto avanços significativos na integração de ferramentas de segurança com processos modernos de entrega de software. **ARCHERY** é uma ferramenta de código aberto com uma comunidade ativa que está fazendo um bom trabalho ao reunir uma coleção de outras ferramentas, incluindo Zap. Projetado principalmente para aplicativos Web, o Archery facilita a integração de ferramentas de segurança em seus sistemas de compilação e implantação. Seus painéis também permitem rastrear vulnerabilidades, bem como realizar verificações de aplicações e de rede.

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. Gostamos do fato de ser executada como teste de unidade dentro de sua configuração de teste existente, embora suporte apenas arquiteturas baseadas em Java. O conjunto de testes de ArchUnit pode ser incorporado em um ambiente de CI ou em um pipeline de implantação, facilitando a implementação de funções de aptidão em uma arquitetura evolucionária.

A execução de testes de ponta a ponta pode apresentar desafios, como a longa duração do processo de execução, a desorganização de alguns testes e a correção de falhas em integração contínua

ao executar testes no modo headless. Nossos times tiveram experiências muito boas com **CYPRESS** resolvendo problemas comuns, como falta de desempenho e longo tempo de espera para carregar respostas e recursos. Cypress é uma ferramenta útil que ajuda pessoas desenvolvedoras a criar testes de ponta a ponta, e registra todas as etapas do teste como vídeo em um arquivo MP4 para facilitar a identificação de erros.

Uma ferramenta simples que impede que você faça commit de senhas e outras informações confidenciais em um repositório git, verificando também todas as revisões históricas antes de tornar um repositório público.

(git-secrets)

A segurança continua a ser primordial e a verificação negligente de credenciais e outros segredos no controle de códigos fonte é um importante vetor de ataque. **GIT-SECRETS** é uma ferramenta simples que impede que você faça commit de senhas e outras informações confidenciais a um repositório git. Ela também pode verificar todas as revisões históricas antes de tornar um repositório público, caso você queira garantir que nunca fez o check-in acidental de uma credencial. O git-secrets vem com suporte integrado para chaves e credenciais comuns da AWS e pode ser configurado rapidamente para outros provedores também.

HEADLESS FIREFOX tem o mesmo nível de maturidade do Headless Chrome para testes de front-end. Semelhante ao Headless Chrome, com o Firefox no modo headless, agora podemos aproveitar os testes do navegador sem os componentes visíveis da interface de usuário (UI), executando o conjunto de testes de UI com muito mais rapidez.

Um dos desafios de usar serviços em nuvem é poder desenvolver e testar localmente usando esses serviços. **LOCALSTACK** resolve esse problema para AWS ao fornecer implementações locais de test doubles para uma ampla variedade de serviços da AWS, incluindo S3, Kinesis, Dynamodb e Lambda. Ela se baseia em ferramentas existentes, como o Kinesalite, Dynalite e Moto e adiciona processos isolados e funcionalidade de injeção de erros. LocalStack é muito fácil de usar e vem com um runner JUnit Moto e uma extensão JUnit 5. Estamos usando em alguns de nossos projetos e ficamos com boa impressão.

MERMAID permite gerar diagramas a partir de uma linguagem de marcação com markdown. Criada pela necessidade de simplificar a documentação, Mermaid tornou-se um ecossistema maior com plugins para Confluence, Visual Studio Code e Jekyll, para citar alguns. Para entender como funciona, você pode usar o Live Editor no GitHub. Mermaid também possui uma conveniente interface de linha de comando que permite gerar arquivos SVG, PNG e PDF como saída dos arquivos de definição. Temos usado Mermaid em muitos projetos e gostamos da simplicidade de descrever gráficos e fluxogramas com markdown e armazenar os arquivos de definição com o repositório de código.

PRETTIER é um formador de código automatizado para JavaScript (com crescente suporte para outras linguagens). Ao impor seu próprio estilo de formatação opinativa, aumenta a consistência e a legibilidade e reduz o esforço de pessoas desenvolvedoras tanto na formatação quanto em debates desnecessários no time sobre o estilo de código. Ainda que você discorde das escolhas de estilo impostas pelo Prettier, achamos que os benefícios para o time geralmente superam os pequenos problemas de estilo. Prettier pode ser usado com um hook pré-commit ou com um plugin IDE. Como acontece com qualquer formatador, uma reformatação única de sua base de código pode confundir seu histórico de controle de versão, mas consideramos isso uma desvantagem secundária. Nós particularmente gostamos da maneira como Prettier inverte a abordagem baseada em linter e, a exemplo de gofmt, em vez de validar seu código, ele garante que seu código sempre será válido.

Prettier é um formatador de código automatizado para JavaScript que aumenta a consistência e a legibilidade e reduz o esforço de desenvolvimento na formatação e os debates desnecessários sobre estilo de código.

(Prettier)

Incluímos o Visual Studio Code no Radar desde 2015, mas ele não é mais a única novidade em IDEs multiplataforma .NET Core. Recentemente, o **RIDER**, que faz parte da plataforma IDEA desenvolvida pela JetBrains, ganhou adoção, especialmente por quem se acostumou com a velocidade e destreza fornecidas pelo ReSharper, que impulsiona a refatoração em Rider. O Rider, no entanto, faz mais do que o ReSharper para

levar a plataforma IDEA completa ao .NET e aumentar a produtividade de quem está desenvolvendo. Independentemente da sua plataforma preferida, vale a pena explorar o Rider, uma vez que atualmente ele leva vantagem na produtividade em relação ao Visual Studio Code. Também é ótimo ver o ecossistema funcionando bem, já que a competição garante que essas ferramentas continuem melhorando.

SNYK te ajuda a encontrar, corrigir e monitorar vulnerabilidades conhecidas em árvores de dependência npm, Ruby, Python, Scala, Golang, .NET, PHP, Java e Docker. Quando adicionado ao seu pipeline de compilação, Snyk monitora e testa continuamente a árvore de dependências de bibliotecas em relação a um banco de dados de vulnerabilidades hospedado e sugere a atualização mínima de versão de dependência direta necessária para a correção.

À medida que mais times adotam DesignOps, as práticas e ferramentas nessa área amadurecem também. Muitos de nossos times agora trabalham com o que poderíamos chamar de **AMBIENTES DE DESENVOLVIMENTO DE UI**, que fornecem um ambiente abrangente para iterar rapidamente os componentes de interface de usuário, concentrando-se na colaboração entre designers de experiência do usuário e pessoas desenvolvedoras. Agora temos algumas opções neste espaço: Storybook, react-styleguidist, Compositor e MDX. Você pode usar essas ferramentas de forma autônoma no desenvolvimento de bibliotecas de componentes ou de sistemas de design, bem como incorporadas em um projeto de aplicação da Web. Em vez de criar o aplicativo, além de um BFF e serviços para adicionar um recurso a um componente, você pode iniciar o servidor dev do Storybook.

VISUAL STUDIO CODE é o editor/IDE gratuito da Microsoft, disponível para várias plataformas. Tivemos uma boa experiência ao usá-lo para desenvolvimento front-end com React, TypeScript e linguagens de back-end como GoLang, sem precisar alternar entre diferentes editores. O conjunto de ferramentas, o suporte a linguagens e as extensões do Visual Studio Code continuam crescendo e melhorando. Gostaríamos de destacar especificamente o VS Live Share para colaboração em tempo real e pareamento remoto. Embora projetos complexos em linguagens estaticamente tipadas, como Java, .NET ou C ++, provavelmente encontrem melhor suporte em IDEs mais maduros da Microsoft ou Jetbrains, acreditamos que o Visual Studio Code está se tornando cada vez mais uma ferramenta de escolha entre times de infraestrutura e desenvolvimento front-end.

A colaboração em tempo real com VS Live Share facilita o emparelhamento remoto. Particularmente, nós gostamos de que isso permita às pessoas desenvolvedoras colaborar usando seus próprios editores pré-configurados.
(VS Live Share)

VS LIVE SHARE é um conjunto de extensões para Visual Studio Code e Visual Studio. A colaboração em tempo real para edição e depuração de código, chamadas de voz, compartilhamento de terminal e exposição de portas locais reduziram alguns dos obstáculos que encontraríamos, caso contrário, ao parear remotamente. Em particular, gostamos do fato de o Live Share permitir que pessoas desenvolvedoras colaborem umas com as outras, enquanto continuam usando seu editor pré-configurado, que inclui temas, mapas de teclas e extensões.

Criar, testar e implantar aplicações móveis envolve várias etapas complexas, especialmente para um pipeline que vai dos repositórios de código-fonte até as lojas de aplicativos. Essa ferramenta fácil de configurar e específica de domínio pode reduzir a complexidade e a sobrecarga de manutenção.
(Bitrise)

Criar, testar e implantar aplicações móveis envolve várias etapas complexas, especialmente quando consideramos um pipeline que vai de repositórios de código-fonte até lojas de aplicativos. Todas essas etapas podem ser automatizadas com scripts e pipelines de compilação em ferramentas genéricas de CI/CD. No entanto, para times que se focam no desenvolvimento móvel e têm pouca ou nenhuma necessidade de integração com pipelines de compilação para sistemas back-end, uma ferramenta específica de domínio pode reduzir a sobrecarga de complexidade e manutenção. **BITRISE** é fácil de configurar e fornece um conjunto abrangente de etapas pré-configuradas para a maioria das necessidades de desenvolvimento móvel.

CODEFRESH é um servidor de CI hospedado semelhante a CircleCI ou Buildkite. Ele é centrado em contêineres, tornando Dockerfiles e clusters de hospedagem de contêiner entidades de primeira classe. Gostamos do fato de a ferramenta incentivar uma abordagem de entrega com pipeline e oferecer suporte a branching e merging. Os primeiros relatórios de nossos times são positivos, mas ainda precisamos ver como isso funciona para projetos maiores e pipelines complexos.

Estamos sempre à procura de ferramentas e técnicas que permitam que os times de entrega trabalhem de forma independente do restante de uma organização maior, mantendo-se dentro de suas margens de segurança e risco. Grafeas é uma dessas ferramentas.

(Grafeas)

Estamos constantemente à procura de ferramentas e técnicas que permitam que os times de entrega trabalhem de forma independente do resto de uma organização maior, mantendo-se dentro de suas margens de segurança e risco. **GRAFEAS** é uma dessas ferramentas. Ela permite que as organizações publiquem metadados autoritativos sobre artefatos de software – imagens do Docker, bibliotecas, pacotes – que podem ser acessados a partir de scripts de compilação ou outros controles de conformidade automatizados. Os mecanismos de controle de acesso permitem uma separação de responsabilidades entre os times que publicam aprovações ou vulnerabilidades e os times que criam e implantam software. Embora várias organizações, incluindo Google e JFrog, usem Grafeas em seus fluxos de trabalho, vale notar que a ferramenta ainda está em versão alfa.

HEPTIO ARK é uma ferramenta para gerenciar recuperação de desastres para clusters Kubernetes e volumes persistentes. O Ark é fácil de usar e configurar e permite que você faça backup e restaure seus clusters por meio de uma série de pontos de verificação. Com o Ark, você pode reduzir significativamente o tempo de recuperação no caso de uma falha de infraestrutura, migrar facilmente os recursos do Kubernetes de um cluster para outro e replicar o ambiente de produção para testes e soluções de problemas. Ark suporta os principais provedores de armazenamento de backup (incluindo AWS, Azure e Google Cloud) e a partir da versão 0.6.0, um sistema de plugins que adiciona compatibilidade para plataformas adicionais de armazenamento de backup e volume. Os ambientes Kubernetes gerenciados, como GKE, fornecem esses serviços prontos para uso. No entanto, se você estiver operando Kubernetes localmente ou na nuvem, avalie com atenção o Heptio Ark para recuperação de desastres.

JAEGER é um sistema de rastreamento distribuído de código aberto. Semelhante ao Zipkin, foi inspirado pelo relatório Google Dapper e está em conformidade com OpenTracing. Jaeger é um projeto de código aberto mais novo que o Zipkin, mas ganhou popularidade rapidamente devido ao maior número de linguagens suportadas pelas bibliotecas clientes, bem como fácil instalação em Kubernetes. Usamos Jaeger com sucesso com o Istio, integrando rastreamentos de aplicações com o Envoy no Kubernetes e gostmos de sua interface do usuário. Com Jaeger se juntando o CNCF, prevemos um esforço maior de engajamento da comunidade e uma integração mais profunda com outros projetos CNCF.

KUBE-BENCH é um exemplo de analisador de configuração de infraestrutura que automatiza a verificação da sua configuração do Kubernetes em relação ao CIS benchmark para K8s. A ferramenta abrange autenticação de usuário, permissões e dados seguros, entre outras áreas. Nossos times têm considerado o kube-bench valioso na identificação de configurações vulneráveis.

Ark é uma ferramenta para gerenciar a recuperação de desastres para clusters Kubernetes e volumes persistentes. É fácil de usar e configurar, e permite fazer backup e restauração de seus clusters por meio de uma série de pontos de verificação.

(Heptio Ark)

OCELOT é um gateway da API .NET. Após três anos de desenvolvimento, o Ocelot construiu um conjunto de recursos relativamente completo e uma comunidade ativa. Embora não faltem excelentes gateways de API

(por exemplo, Kong), a comunidade .NET parece preferir o Ocelot para criar microsserviços. Em parte, o motivo é que o Ocelot se integra bem ao ecossistema .NET (por exemplo, com o IdentityServer). Outra razão pode ser que a comunidade .NET tenha estendido o Ocelot para suportar protocolos de comunicação como gRPC, Orleans e WebSocket.

Pesquisa de UX exigem coleta e análise de dados para melhores tomadas de decisões sobre os produtos que precisamos construir. O **OPTIMAL WORKSHOP** é um conjunto de ferramentas que ajuda a fazer isso de forma digital. Recursos como primeiro clique ou classificação de cartões ajudam a validar os protótipos e melhorar a navegação no site e a exibição de informações. Times distribuídos, em particular, beneficiam-se do Optimal Workshop, já que ele permite conduzir pesquisas remotas.

A extração de informações de negócios significativas de dados de texto é uma técnica fundamental para o processamento de dados não-estruturados. Stanford CoreNLP, um conjunto de ferramentas de processamento de linguagem natural baseado em Java, nos ajuda a usar as pesquisas mais recentes na área de NLP para resolver vários problemas de negócios.

(Stanford CoreNLP)

Temos cada vez mais projetos que exigem processamento de dados não estruturados. Extrair informações de negócios significativas de dados de texto é uma técnica fundamental. **STANFORD CORENLP** é um conjunto de ferramentas de processamento de linguagem natural baseadas em Java. Ele suporta reconhecimento de entidades nomeadas, extração de relacionamentos, análise de sentimentos e classificação de texto, além de vários idiomas, incluindo inglês, chinês e árabe. Também encontramos ferramentas úteis para rotular corpus e modelos de treinamento para o nosso cenário. Com Stanford CoreNLP, pudemos usar as pesquisas mais recentes na área de NLP para resolver vários problemas de negócios.

Nós usamos amplamente Terraform como código para configurar infraestrutura de nuvem. **TERRAGRUNT** é um empacotador leve para Terraform que implementa as práticas defendidas pelo livro Terraform: Up and Running. Consideramos o Terragrunt útil, já que incentiva módulos versionados e reusabilidade para diferentes ambientes, com alguns recursos úteis, incluindo execução de código recursivo em subdiretórios. Gostaríamos de ver a ferramenta evoluir para suportar práticas de CD nativamente, onde todo o código pode ser empacotado, versionado e reutilizado em diferentes ambientes em pipelines de CD. Nossa equipe consegue isso hoje com soluções alternativas.

Nossos times estão relatando experiências de sucesso com **TESTCAFE**, uma ferramenta de automação de testes de navegador baseada em JavaScript. O TestCafe permite que você escreva testes em JavaScript ou TypeScript e execute testes em qualquer navegador que suporte JavaScript. O TestCafe tem vários recursos úteis, incluindo execução paralela pronta para uso e simulação de solicitação de HTTP. O TestCafe usa um modelo de execução assíncrona sem tempos de espera explícitos, o que resulta em suítes de testes muito mais estáveis.

TRAEFIK é um proxy reverso e balanceador de carga de código aberto. Se você estiver procurando por um proxy de borda que forneça roteamento simples sem todos os recursos de NGINX e HAProxy, Traefik é uma boa escolha. O roteador fornece reconfiguração sem necessidade de recarga, métricas, monitoramento e disjuntores, essenciais para a execução de microsserviços. Ele também se integra muito bem com Let's Encrypt para fornecer SSL termination. Quando comparadas ao Traefik, ferramentas como NGINX e HAProxy podem demandar o uso de ferramentas adicionais para modelar a configuração em resposta à escala, adição ou remoção de microsserviços e podem, eventualmente, exigir uma reinicialização, o que pode ser importuno em ambientes de produção.

Valorizamos muito o feedback rápido durante o desenvolvimento orientado a testes e estamos sempre procurando novas maneiras de torná-lo ainda mais rápido. **WALLABY.JS** é uma extensão comercial para editores populares que fornece execução contínua de testes de unidade JavaScript, destacando os resultados em linhas ao lado de seu código. A ferramenta identifica e executa o conjunto mínimo de testes afetados por cada alteração de código e permite que você execute testes continuamente enquanto digita.

LINGUAGENS & FRAMEWORKS

ADOTE

EXPERIMENTE
76. Jepsen
77. MMKV ***NOVO***
78. MockK ***NOVO***
79. TypeScript

AVALIE
80. Apache Beam ***NOVO***
81. Camunda ***NOVO***
82. Flutter
83. Ktor ***NOVO***
84. Nameko ***NOVO***
85. Polly.js ***NOVO***
86. PredictionIO ***NOVO***
87. Puppeteer ***NOVO***
88. Q# ***NOVO***
89. SAFE stack ***NOVO***
90. Spek ***NOVO***
91. troposphere
92. WebAssembly
93. WebFlux ***NOVO***

EVITE

Com a crescente adoção da arquitetura de microsserviços, estamos construindo mais aplicativos distribuídos do que antes. Embora haja muitos benefícios em uma arquitetura desacoplada, a complexidade e o esforço envolvidos na comprovação da correção do sistema geral aumentaram drasticamente. **JEPSEN** fornece ferramentas muito necessárias para verificar a exatidão na coordenação de agendadores de tarefas, testar consistência eventual, linearização e serialização características de bancos de dados distribuídos. Usamos Jepsen em alguns projetos e gostamos do fato de podermos simular configurações, injetar e corrigir falhas e verificar o estado do sistema após a recuperação.

MMKV é um framework de código aberto desenvolvido pelo WeChat, que fornece armazenamento rápido de valor-chave para aplicações móveis. Ele usa recursos de mapeamento de memória do iOS para evitar a necessidade de salvar explicitamente as alterações e é extremamente rápido e eficiente. No caso de uma falha inesperada, o MMKV permite que o aplicativo restaure os dados rapidamente.

MockK é uma biblioteca nativa que ajuda nossos times a escrever testes limpos e concisos para aplicativos Kotlin, ao invés de usar empacotadores incômodos do Mockito ou do PowerMock..

(MockK)

MOCKK é uma biblioteca para mocking escrita em Kotlin. Sua filosofia principal é fornecer suporte de primeira classe para recursos da linguagem Kotlin, como co-rotinas ou blocos lambda. Como uma biblioteca nativa, ele ajuda nossos times a escrever código limpo e conciso ao testar aplicações Kotlin, ao invés de usar empacotadores incômodos do Mockito ou do PowerMock.

TYPESCRIPT é uma linguagem cuidadosamente considerada, e suas ferramentas de aprimoramento constante, bem como seu suporte a IDE continuam a nos impressionar. Com um bom repositório de definições de tipo TypeScript, nós nos beneficiamos

de todas as bibliotecas ricas de JavaScript, ao mesmo tempo em que ganhamos segurança de tipagem. Isso é particularmente importante à medida que nossa base de código baseada no navegador continua a crescer. A tipagem no TypeScript permite usar IDEs e outras ferramentas para fornecer um contexto mais profundo ao seu código, além de fazer alterações e refatorar o código com segurança. O TypeScript, sendo um superconjunto de JavaScript, somado à documentação e à comunidade, ajudaram a facilitar a curva de aprendizado.

APACHE BEAM é um modelo de programação unificado de código aberto para definir e executar, em paralelo, pipelines em lote e streaming de dados. O Beam fornece uma camada de API portátil para descrever esses pipelines independentemente dos mecanismos de execução (ou runners), como Apache Spark, Apache Flink ou Google Cloud Dataflow. Diferentes runners têm diferentes capacidades e fornecer uma API portátil é uma tarefa difícil. Beam tenta encontrar um equilíbrio delicado, extraindo ativamente as inovações desses runners para o modelo Beam e também trabalhando com a comunidade para influenciar o roadmap desses runners. O Beam tem um rico conjunto de transformações de I/O integradas que cobre a maioria das necessidades de pipeline de dados. Ele também fornece um mecanismo para implementar transformações personalizadas para casos de uso específicos. A API portátil e as transformações de IO extensíveis formam um argumento convincente para avaliar o Apache Beam para necessidades de pipeline de dados.

Embora tenhamos uma tendência ao ceticismo em relação às ferramentas de modelo e notação de processos de negócios (BPMN), Camunda é fácil de testar, versionar e refatorar. A integração com Spring e Spring Boot o torna uma escolha sólida.

(Camunda)

Temos uma tendência ao ceticismo em relação às ferramentas de modelo e notação de processos de negócios (BPMN), pois geralmente elas estão associadas a ambientes de baixo código e suas desvantagens. Embora o framework OSS BPMN **CAMUNDA** forneça um pouco dessa característica, ele também oferece fluxo de trabalho e mecanismos de decisão que podem ser integrados diretamente como uma biblioteca no seu código Java. Isso facilita o teste, a versionamento e a refatoração de fluxos de trabalho. Camunda também se integra com Spring e Spring Boot, entre outros frameworks, tornando-o uma escolha sólida.

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.

Kotlin não é mais apenas uma excelente opção para o desenvolvimento de aplicativos para dispositivos móveis. Novas ferramentas e frameworks surgiram, demonstrando o valor da linguagem também para o desenvolvimento de aplicações web. **KTOR** é um desses frameworks. Ao contrário de outros frameworks web que suportam Kotlin, Ktor é escrito em Kotlin, usando recursos da linguagem como coroutines, que permite uma implementação assíncrona sem bloqueio. A flexibilidade de incorporar diferentes ferramentas para criação de logs, injeção de dependência (DI) ou um mecanismo de templates – além de sua arquitetura leve – torna Ktor uma opção interessante para nossos times na criação de serviços RESTful.

Um insight que tivemos depois de conversar com nossos times é que Python está voltando para vários domínios tecnológicos. Na verdade, está a caminho de se tornar a linguagem de programação mais usada. Em parte, isso é impulsionado por sua adoção por cientistas de dados e em aprendizado de máquina, mas também vemos times adotando para construir microsserviços. **NAMEKO** é um framework de microsserviços super leve e uma alternativa a Flask para escrever serviços. Ao contrário do Flask, o Nameko possui apenas um conjunto limitado de recursos que inclui suporte para WebSocket, HTTP e AMQP. Também gostamos do foco na testabilidade. Se você não precisa de recursos como os templates que o Flask fornece, então vale a pena analisar Nameko.

POLLY.JS é uma ferramenta simples que ajuda os times a testar sites e aplicações em JavaScript. Nossos times gostam particularmente disso, permitindo que interceptem e façam stub de interações HTTP, o que permite um teste mais fácil e rápido do código JavaScript sem precisar aumentar os serviços ou componentes dependentes.

PREDICTIONIO é um servidor de aprendizagem de máquina de código aberto. Pessoas desenvolvedoras e cientistas de dados podem usá-lo para criar aplicatições inteligentes para previsão. Como todos as aplicações inteligentes, o PredictionIO tem três partes: coleta e armazenamento de dados, treinamento de modelos, implantação de modelos e serviço de exposição. As pessoas desenvolvedoras podem se concentrar na implementação de lógica de processamento de dados, algoritmo de modelo e lógica de previsão com base nas interfaces correspondentes, e liberar-se do armazenamento de dados e da implantação de treinamento de modelo. Em nossa experiência, o PredictionIO pode suportar volumes grandes e pequenos de dados com baixa conconrrência. Utilizamos o PredictionIO principalmente para criar serviços preditivos para pequenas e médias empresas ou como prova de conceito ao criar mecanismos de previsão mais complexos e personalizados.

No Radar anterior, mencionamos o Headless Chrome para testes de front-end. Com a adoção do Chrome DevTools Protocol (CDP) por outros navegadores, um novo conjunto de bibliotecas está surgindo para automação e testes em navegadores. O CDP permite um controle refinado sobre o navegador, mesmo no modo headless. Novas bibliotecas de alto nível estão sendo criadas usando o CDP para testes e automação. **PUPPETEER** é uma dessas novas bibliotecas. Ela pode conduzir o headless Chrome por meio de uma aplicação de página única, obter rastreamento de tempo para diagnósticos de desempenho e muito mais. Nossos times acharam mais rápido e mais flexível que as alternativas baseadas no WebDriver.

Enquanto ainda esperamos o hardware chegar, podemos experimentar a computação quântica usando linguagens e simuladores. As amostras do Q# podem dar uma ideia do potencial futuro da programação.

(Q#)

A computação quântica atualmente existe em uma espécie de zona imaginária, por estar disponível para testes sem ter chegado ainda. Enquanto ainda esperamos o hardware chegar, podemos experimentar e aprender com linguagens e simuladores. Embora a IBM e outras empresas tenham feito um bom progresso, prestamos atenção especial aos esforços da Microsoft com base na linguagem **Q#** e seu simulador (32 qubits localmente e 40 no Azure). Se você quiser começar a pensar sobre o futuro da programação, confira o conjunto de exemplos no GitHub.

SAFE STACK – cujo nome deriva das iniciais de Suave, Azure, Fable e Elmish – traz uma série de tecnologias para uma stack coerente para desenvolvimento web. Ela é construído em torno da linguagem de programação F#, tanto no lado do servidor quanto no navegador, e, portanto, tem foco na programação funcional e segura de tipos, com uma abordagem

assíncrona. Ele oferece recursos de produtividade, como o recarregamento hot, e permite substituir partes da stack, por exemplo, a estrutura Web do lado do servidor ou o provedor de nuvem.

A adoção de uma nova linguagem normalmente provoca o surgimento de novas ferramentas que suportam práticas de engenharia maduras, como a automação de testes. Kotlin não é exceção. **SPEK** é um framework de testes – inspirada em ferramentas conhecidas, como Cucumber, RSpec and Jasmine – que cria testes em Gherkin e Specification, permitindo que os times tragam práticas maduras, como o desenvolvimento orientado por comportamento, para o espaço Kotlin.

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 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.

Depois de trabalhar com um estilo funcional reativo em várias aplicações, nossos times ficaram impressionados e relataram que a abordagem melhora a legibilidade do código e o rendimento do sistema.

(WebFlux)

O Spring Framework 5, lançado há mais de um ano, inclui fluxos reativos, um padrão para o processamento de fluxo assíncrono com contrapressão não-bloqueante. O módulo **WEBFLUX** apresenta um alternativa reativa ao módulo Spring MVC tradicional para escrever aplicações Web no ecossistema Spring. Depois de trabalhar com ele em várias aplicações, nossos times ficaram impressionados e relataram que a abordagem reativa (funcional) melhora a legibilidade do código e a taxa de transferência do sistema. Eles observam, no entanto, que a adoção do WebFlux requer uma mudança significativa no pensamento e recomendam que isso seja considerado na decisão de escolher o WebFlux em relação ao Spring MVC.

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

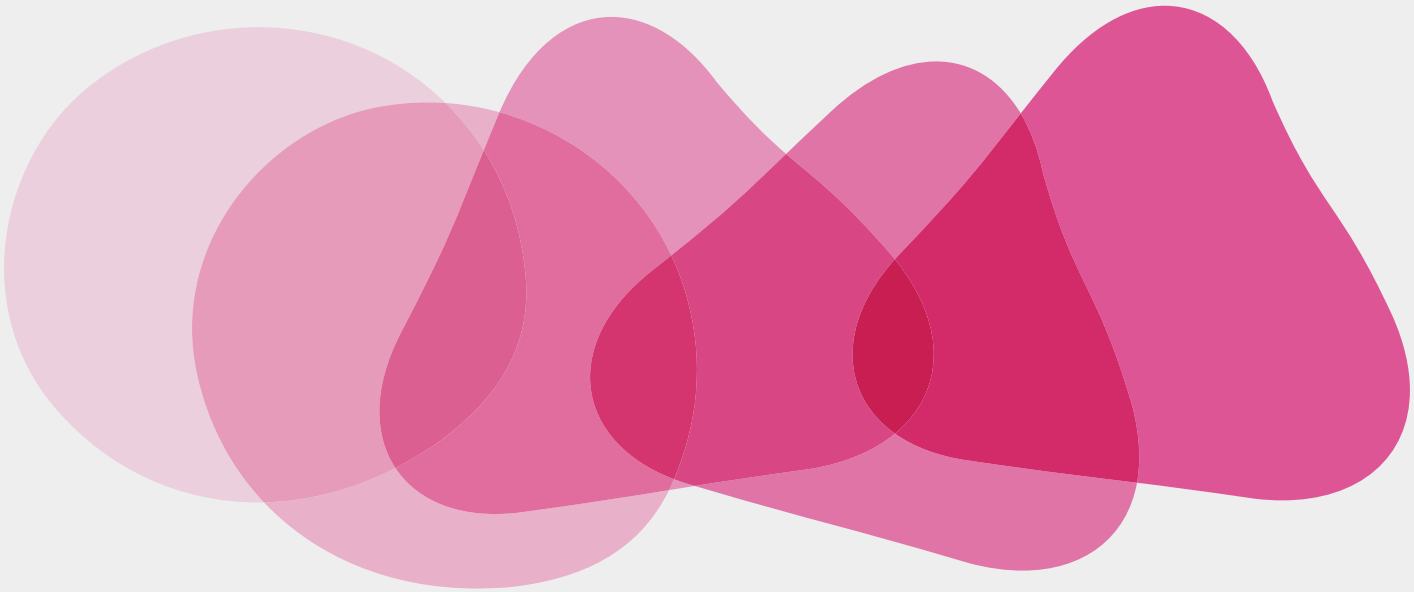
INSCREVA-SE

thght.works/Sub-PT

ThoughtWorks®

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

Fundada há 25 anos, a ThoughtWorks se tornou uma empresa com mais de 5000 pessoas, incluindo uma área de produtos que desenvolve ferramentas pioneiras para times de software. A ThoughtWorks tem 41 escritórios em 14 países: Alemanha, Austrália, Brasil, Canadá, Chile, China, Equador, Espanha, Estados Unidos, Índia, Itália, Reino Unido, Singapura e Tailândia.



thoughtworks.com/pt/radar

#TWTechRadar