ThoughtWorks®

TECHNOLOGY RADAR *VOL.16*

Ideias sobre as
tecnologias e tendências
que estão moldando o futuro

thoughtworks.com/radar

#TWTechRadar

COLABORADORES

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

Rebecca Parsons (CTO) | Martin Fowler (Chief Scientist) | Badri Janakiraman | Bharani Subramaniam | Camilla Crispim
Erik Doernenburg | Evan Bottcher | Fausto de la Torre | Hao Xu | Ian Cartwright
James Lewis | Jonny LeRoy | Marco Valtas | Mike Mason | Neal Ford
Rachel Laycock | Scott Shaw | Srihari Srinivasan | Zhamak Dehghani

TRADUÇÃO

Alexey Villas Bôas | Glauco Vinícius | Luciano Ramalho | Maitê Balhester | Marco Valtas
Marcos Brizeno | Narciso Benigno | Paula Ribas | Pedro Evangelista | Raquel Guimarães

O QUE HÁ DE NOVO?

Assuntos em destaque nessa edição:

INTERFACE CONVERSIONAL E PROCESSAMENTO DE LINGUAGEM NATURAL

assista ao vídeo (thght.works/ConUI)

Conversação—uma nova maneira de interagir com aplicações—pegou o ecossistema de surpresa com ferramentas como Siri, Cortana e Allo, e depois se estendeu aos lares com aparelhos como Amazon Echo e Google Home.

Construir interfaces de usuário conversionais e com linguagem natural, apesar de apresentar novos desafios, tem benefícios bem óbvios. O time por trás do Amazon Echo omitiu a tela intencionalmente, obrigando o próprio time a repensar muitas das interações entre ser humano e computador.

A tendência conversacional não é limitada apenas à voz; com aplicativos de mensagem crescendo e dominando tanto telefones quanto o espaço de trabalho, vemos conversas com outros humanos sendo complementadas por chatbots inteligentes. Na medida em que essas plataformas melhoram, eles vão entender o contexto e intenção das conversas, tornando as interações mais reais e portanto mais atraentes.

A explosão de interesse do mercado e da grande mídia tem como efeito um crescimento correspondente de interesse em desenvolver utilizando essa nova forma de exocortex pessoal para interações.

INTELIGÊNCIA COMO SERVIÇO

assista ao vídeo (thght.works/IntSer)

Uma família de plataformas explodiu no cenário de tecnologia recentemente, a qual chamamos de *inteligência como serviço*. Essas plataformas abrangem uma grande variedade de utilidades surpreendentemente poderosas de processamento de voz para a compreensão da linguagem natural, reconhecimento de imagem e aprendizagem profunda.

Capacidades que teriam tido um alto custo há alguns anos agora aparecem como plataformas de código aberto ou SaaS. Parece que a "guerra na nuvem" deu lugar para uma competição em armazenamento e computação para capacidades cognitivas, como sugerido pela disposição em abrir o código de ferramentas até então consideradas diferenciais, como Kubernetes e Mesos.

Todos as grandes empresas têm ofertas nesse espaço, bem como algumas empresas de nicho interessantes que valem uma avaliação. Embora ainda tenhamos reservas sobre as implicações éticas e de privacidade desses serviços, vemos um grande potencial na utilização dessas ferramentas poderosas de maneira inovadora. Nossos clientes já estão investigando quais novos horizontes eles podem alcançar combinando cognição de comódite com inteligência em seus próprios negócios.

EXPERIÊNCIA DE DESENVOLVIMENTO COMO NOVO DIFERENCIAL

assista ao vídeo (thght.works/DevExp)

O design da experiência do usuário tem sido um diferencial chave para empresas de produtos de tecnologia por muitos anos. Agora, a rápida ascensão de ferramentas e produtos voltados para quem desenvolve, combinada com a escassez de talentos em engenharia de software, está direcionando um foco semelhante na experiência do desenvolvedor.

Cada vez mais, as organizações avaliam as ofertas de nuvem com base na quantidade de atrito de engenharia que elas reduzem, tratam as APIs como produtos e montam times focados na produtividade. Na ThoughtWorks, sempre tivemos uma obsessão por práticas de engenharia eficientes e promovemos ferramentas e plataformas que tornam a vida das pessoas desenvolvedoras mais fácil, por isso nos anima ver que a indústria começa a adotar essa abordagem.

As principais técnicas incluem: tratar a infraestrutura interna como um produto que precisa ser atraente o suficiente para competir com ofertas externas, concentrando-se no auto-serviço, compreendendo a ergonomia do desenvolvedor nas APIs produzidas, contendo "código legado encaixotado" e se comprometendo com uma pesquisa empática contínua de usuário com pessoas desenvolvedoras usando seus serviços.

A ASCENSÃO DAS PLATAFORMAS

assista ao vídeo (thght.works/RiseOTP)

Os temas do radar emergem das observações e das conversas durante o processo de debate. Recentemente, ao compilar o Radar, chamou nossa atenção o número de novas entradas no quadrante das Plataformas. Achamos que isso é indicativo de uma tendência mais ampla no ecossistema de desenvolvimento de software.

Notáveis empresas do Vale do Silício têm ilustrado como construir uma plataforma adequada pode render benefícios significativos. Parte de seu sucesso vem de encontrar um nível útil de encapsulamento e capacidades. Cada vez mais, o "pensamento de plataforma" aparece em todo o ecossistema—de capacidades avançadas destacadas no Radar, como linguagem natural, até plataformas de infraestrutura, como a Amazon.

As empresas estão começando a pensar em plataformas quando expõem recursos selecionados através de APIs inspiradas em produtos. As equipes de desenvolvimento pensam mais em termos de construção de plataformas para integração e melhor experiência do desenvolvedor. Parece que a indústria tem finalmente se apegado a uma combinação razoável de embalagem, conveniência e utilidade.

Uma definição que gostamos é que as plataformas devem expor uma API de autoatendimento e ser fáceis de configurar e provisionar no ambiente utilizado pelo time—o que faz uma boa interseção com outro tema emergente, a *experiência de desenvolvimento como o novo diferencial*. Esperamos ver mais refinamento na definição e nas capacidades das plataformas no futuro próximo.

PYTHON ONIPRESENTE

assista ao vídeo (thght.works/PerPyt)

Python é uma linguagem que continua aparecendo em lugares interessantes. Sua facilidade de uso como uma linguagem de programação geral, combinada com a sua forte base na computação matemática e científica tem historicamente levado a sua adoção pelas comunidades acadêmicas e de pesquisa. Mais recentemente, as tendências da indústria em torno da comoditização e das aplicações da AI, combinadas com a maturidade de Python 3, ajudaram a trazer novas comunidades para Python.

Essa edição do Radar apresenta algumas bibliotecas Python que ajudaram a impulsionar o ecossistema, incluindo Scikit-learn no domínio de aprendizagem de máquina; TensorFlow, Keras e Airflow para gráficos inteligentes de fluxo de dados; além de spaCy, que implementa o processamento de linguagem natural para ajudar a capacitar APIs de conversação. Cada vez mais, vemos Python preenchendo a lacuna entre cientistas e engenheiros dentro das organizações, afrouxando o preconceito que existia contra suas ferramentas favoritas.

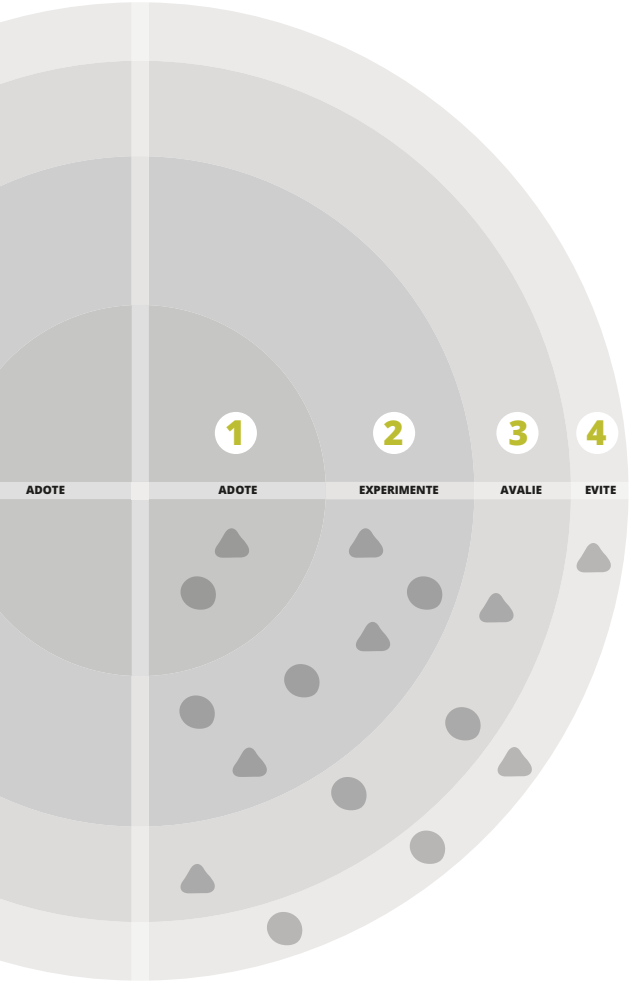
Abordagens arquitetônicas como microsserviços e contêineres facilitaram a execução de Python em ambientes de produção. Engenheiros podem agora implantar e integrar códigos especializados em Python criados por cientistas através de APIs agnósticas de linguagem e tecnologia. Esta fluidez é um grande passo em direção a um ecossistema consistente entre pesquisadores e engenheiros, em contraste com a existência de fato de traduzir linguagens especializadas como R para os ambientes de produção.

SOBRE O RADAR

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

O Radar captura o resultado das discussões do TAB em um formato que procura oferecer valor a uma ampla gama de pessoas interessadas, de CTOs a pessoas que desenvolvem. O conteúdo é concebido para ser um resumo conciso. Nós encorajamos você a explorar essas tecnologias para obter mais detalhes. O Radar é gráfico por natureza, agrupando os itens em técnicas, ferramentas, plataformas e linguagens & frameworks. Quando itens do radar puderem ser classificados em mais de um quadrante, escolhemos aquele que parece mais adequado. Além disso, agrupamos esses itens em quatro anéis para refletir a nossa posição atual em relação a eles.

Para mais informações sobre o Radar, veja:
thoughtworks.com/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. Pipelines como código

EXPERIMENTE
2. APIs como produto
3. Desacoplar gerenciamento de segredos do código-font *NOVO*
4. Hospedando dados PII na União Europeia
5. Legado encaixotado *NOVO*
6. Registros de decisões de arquiteturas leves
7. Aplicações Web progressivas *NOVO*
8. Prototipagem com InVision e Sketch *NOVO*
9. Arquitetura sem servidor

AVALIE
10. Consulta dirigida ao cliente
11. Varredura de segurança de contêiner
12. APIs aptas para conversação *NOVO*
13. Privacidade diferencial
14. Micro frontends
15. Times de produto de engenharia de plataforma *NOVO*
16. Análise social de código *NOVO*
17. Realidade Virtual além dos jogos

EVITE
18. Instância única de integração contínua para todos os times
19. REST anêmico
20. Inveja de Big Data
21. CI de faz de conta *NOVO*
22. Ambientes de teste de integração empresariais *NOVO*
23. Geração de código baseada em especificações *NOVO*

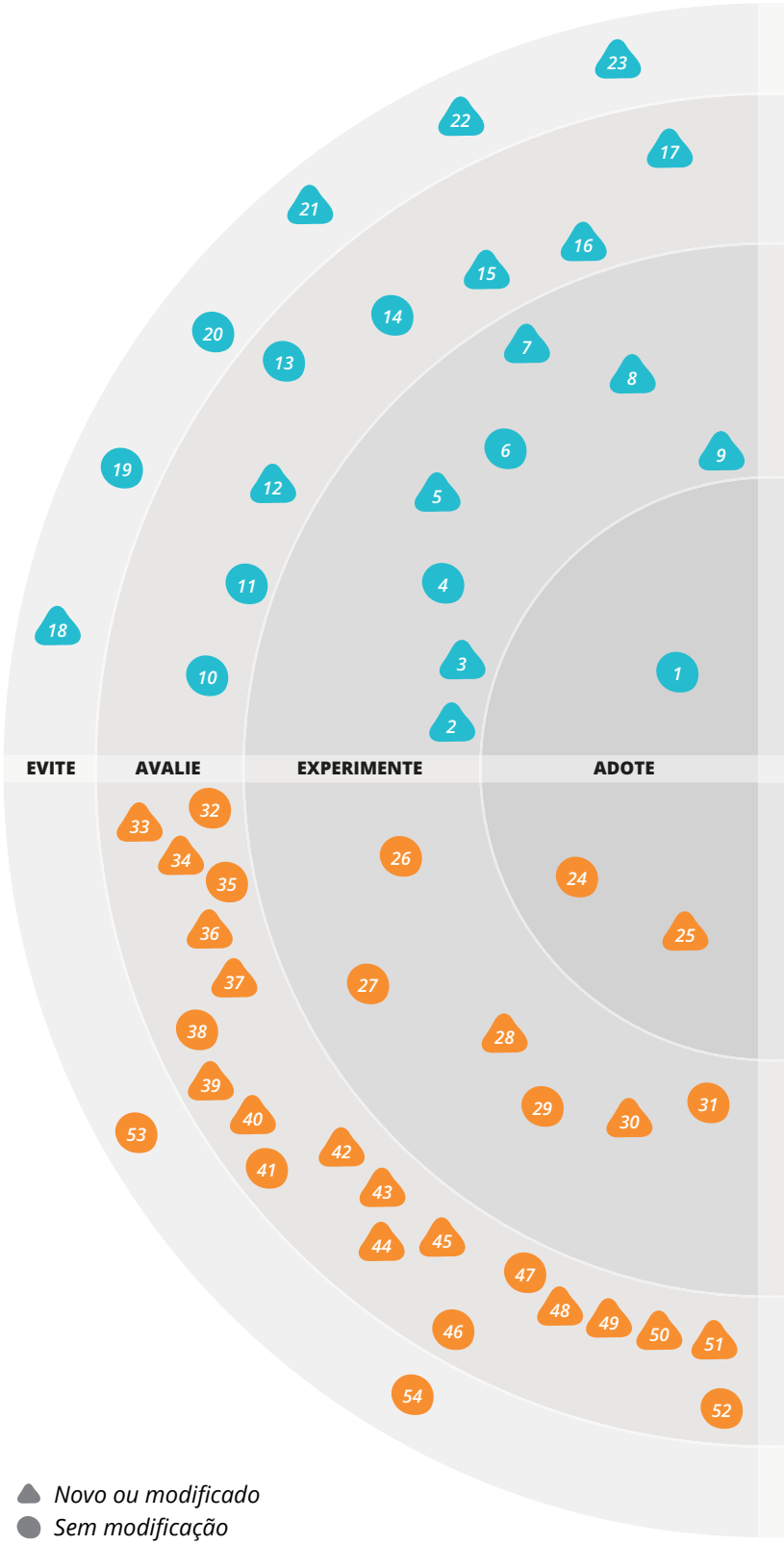
PLATAFORMAS

ADOTE
24. HSTS
25. Módulos de segurança do Linux

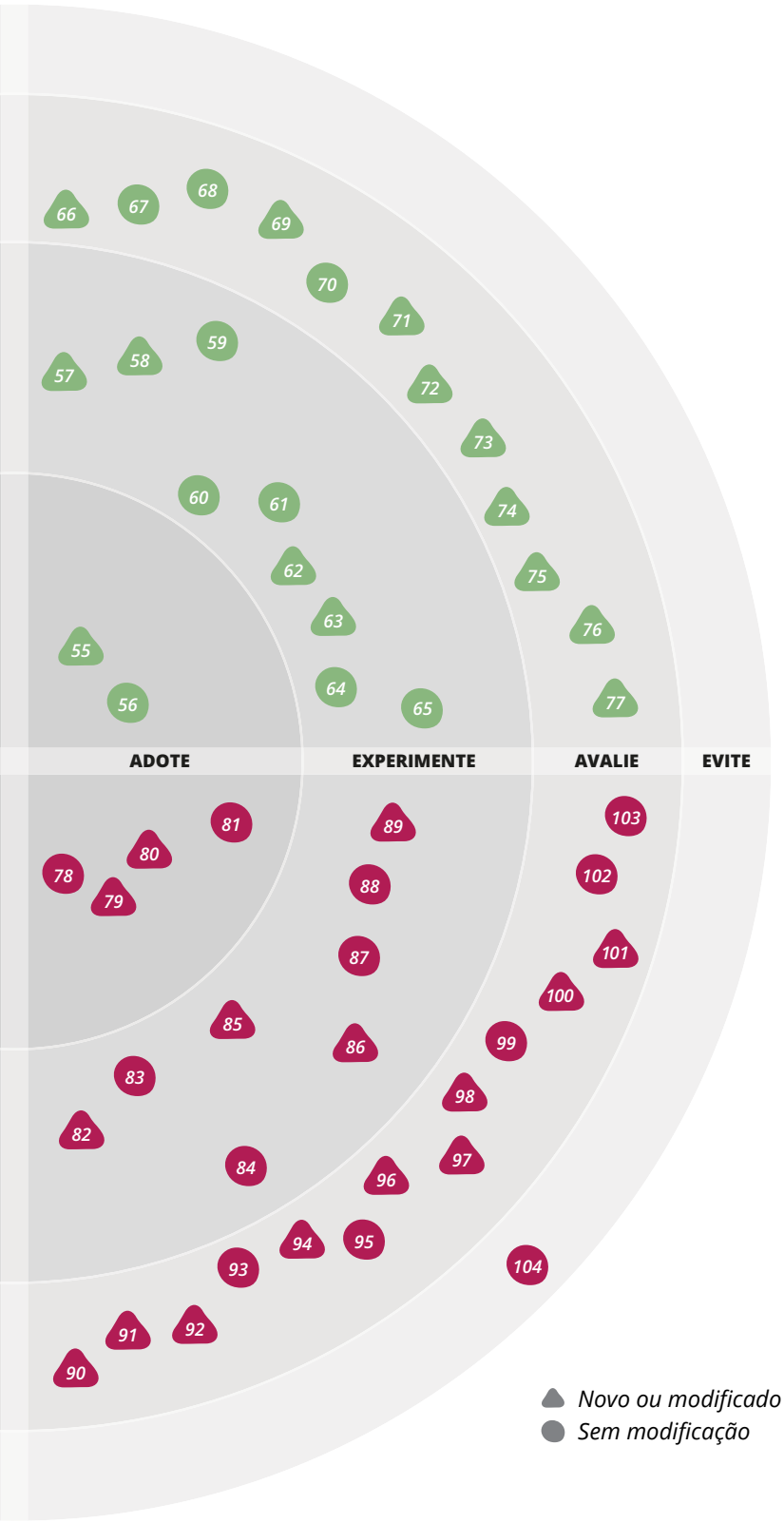
EXPERIMENTE
26. Apache Mesos
27. Auth0
28. AWS Device Farm *NOVO*
29. AWS Lambda
30. OpenTracing *NOVO*
31. Unity além dos jogos

AVALIE
32. .NET Core
33. Amazon API Gateway
34. api.ai *NOVO*
35. Cassandra com cuidado
36. Compreensão de imagem baseada na nuvem *NOVO*
37. DataStax Enterprise Graph *NOVO*
38. Electron
39. Ethereum
40. Hyperledger *NOVO*
41. IndiaStack
42. Kafka Streams *NOVO*
43. Keycloak *NOVO*
44. Mesosphere DCOS
45. Mosquitto *NOVO*
46. Nuance Mix
47. OpenVR
48. PlatformIO *NOVO*
49. Tango *NOVO*
50. Plataformas conversacionais *NOVO*
51. WebVR *NOVO*
52. wit.ai

EVITE
53. CMS como plataforma
54. API gateways excessivamente ambiciosos



O RADAR



FERRAMENTAS

ADOTE
55. fastlane
56. Grafana

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

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

EVITE

LINGUAGENS & FRAMEWORKS

ADOTE
78. Ember.js
79. Python 3
80. ReactiveX
81. Redux

EXPERIMENTE
82. Avro *NOVO*
83. Elixir
84. Enzyme
85. Hangfire *NOVO*
86. Nightwatch *NOVO*
87. Phoenix
88. Quick e Nimble
89. Vue.js

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

EVITE
104. AngularJS

TÉCNICAS

ADOTE
1. Pipelines como código

EXPERIMENTE
2. APIs como produto
3. Desacoplar gerenciamento de segredos do código-font *NOVO*
4. Hospedando dados PII na União Europeia
5. Legado encaixotado *NOVO*
6. Registros de decisões de arquiteturas leves
7. Aplicações Web progressivas *NOVO*
8. Prototipagem com InVision e Sketch *NOVO*
9. Arquitetura sem servidor

AVALIE
10. Consulta dirigida ao cliente
11. Varredura de segurança de contêiner
12. APIs aptas para conversação *NOVO*
13. Privacidade diferencial
14. Micro frontends
15. Times de produto de engenharia de plataforma *NOVO*
16. Análise social de código *NOVO*
17. Realidade Virtual além dos jogos

EVITE
18. Instância única de integração contínua para todos os times
19. REST anêmico
20. Inveja de Big Data
21. CI de faz de conta *NOVO*
22. Ambientes de teste de integração empresariais *NOVO*
23. Geração de código baseada em especificações *NOVO*

Empresas estão abraçando de coração aberto APIs como uma maneira de expor suas capacidades de negócios, tanto externa quanto internamente. APIs prometem a habilidade de fazer experimentos rápidos com novas ideias de negócio, apenas combinando as capacidades principais. Mas o que diferencia APIs de outros serviços de integração quaisquer? Uma diferença é tratar **APIS COMO PRODUTO**, mesmo quando o consumidor é um sistema interno ou uma colega desenvolvedora. Times que constroem APIs deveriam entender a necessidade dos seus clientes e tornar o produto atrativo para eles. Testes de usabilidade e pesquisa de UX podem levar a um melhor design e entendimento de padrões de uso da API, ajudando a trazer um pensamento de produto para APIs. APIs, assim como produtos, devem ser ativamente mantidas, ter suporte e serem fáceis de usar. Elas devem ter um dono que defende seus usuários e caminha em direção à melhoria contínua. Em nossa experiência, pensamento de produto é o ingrediente que falta e faz toda a diferença entre uma integração qualquer e um negócio ágil construído em cima de uma plataforma de APIs.

Em edições anteriores do Radar, citamos ferramentas como git-crypt e Blackbox, que nos permitem manter segredos com segurança dentro do código fonte. **DESACOPLAR GERENCIAMENTO DE SEGREDOS DO CÓDIGO-FONTE** é a nossa maneira de lembrar as tecnologistas que existem outras opções para armazenar segredos. Por exemplo, HashiCorp Vault, servidores de CI e ferramentas de gerenciamento de configuração fornecem mecanismos para armazenar segredos que não são vinculados ao código-fonte de uma aplicação. Ambas as abordagens são viáveis e recomendamos que você use pelo menos uma delas em seus projetos.

Times que constroem APIs deveriam entender a necessidade dos seus clientes e tornar o produto atrativo para eles. Testes de usabilidade e pesquisa de UX podem levar a um melhor design e entendimento de padrões de uso da API, ajudando a trazer um pensamento de produto para APIs.

— APIs como produto

Trabalhar com código legado, especialmente com grandes monólitos, é uma das experiências mais insatisfatórias e desgastantes para quem desenvolve. Apesar da nossa sugestão contra extender e ativamente manter monólitos legados, eles continuam a ser dependências em nossos ambientes e as pessoas desenvolvedoras muitas vezes subestimam o custo e o tempo necessários para desenvolver junto dessas dependências. Para ajudar a reduzir o desgaste, usamos imagens de máquinas virtuais ou imagens de contêiner com Docker para criar imagens imutáveis de sistemas legados e suas configurações. A intenção é manter o **LEGADO ENCAIXOTADO** para ser executado localmente, removendo a necessidade de reconstruir, reconfigurar ou compartilhar ambientes. Em um cenário ideal, os times que possuem sistemas legados geram as imagens legadas por meio de seus pipelines de compilação, e as pessoas que desenvolvem podem executar e orquestrar essas imagens em seus sandboxes alocados com mais confiabilidade. Embora essa abordagem tenha reduzido o tempo total gasto por cada pessoa desenvolvedora, o sucesso foi limitado quando times que são donos das dependências do legado relutaram em criar imagens de contêiner para uso de outras pessoas.

O aumento de **APLICAÇÕES WEB PROGRESSIVAS** (PWAs) é a última tentativa de recuperar a web móvel como resposta ao "desgaste" com aplicativos dos usuários. Originalmente propostas pelo Google em 2015, PWAs são aplicações web que usam as mais recentes tecnologias para combinar o melhor da web e aplicações móveis nativas. Usando um conjunto de tecnologias padrão abertas, como service workers, app manifest e cache e push APIs, podemos criar aplicações independentes da plataforma e oferecer experiências de usuário semelhantes às dos aplicativos. Isso traz paridade para aplicações web e nativas e ajuda quem desenvolve para dispositivos móveis a atingir usuários além dos limites impostos pelas lojas de aplicativos. Pense em PWAs como sites que agem e sentimos como aplicativos nativos.

O uso combinado de InVision e Sketch alterou a forma como algumas pessoas abordam o desenvolvimento de aplicações web. Embora sejam ferramentas, é a técnica de **PROTOTIPAGEM COM INVISION E SKETCH** que realmente torna este blip significativo. Criar protótipos ricos e clicáveis como ponto de partida para implementar o comportamento de front-end e back-end ajuda a acelerar o desenvolvimento e elimina o atrito nos detalhes de implementação. O uso combinado dessas ferramentas atinge o equilíbrio certo entre a elaboração prematura dos detalhes visuais e a antecipação de feedback dos usuários sobre a experiência interativa.

A abordagem de **ARQUITETURA SEM SERVIDOR** substitui máquinas virtuais de longa duração por poder computacional efêmero, que passa a existir sob solicitação e desaparece imediatamente após o uso. Nossos times gostam da abordagem sem servidor; tem funcionado bem para nós e a consideramos uma escolha de arquitetura válida. Observe que a abordagem sem servidor não precisa ser tudo ou nada: alguns de nossos times implantaram uma parte de seus sistemas sem servidor embora tenham mantido uma abordagem tradicional de arquitetura para outras partes. Ainda que AWS Lambda seja quase um sinônimo de sem servidor, todos os demais grandes provedores de nuvem têm ofertas similares, e recomendamos ainda avaliar opções de nicho como webtask.

Tecnologias como Amazon Alexa, Google Voice e Siri reduziram drasticamente a barreira para interação com software através de voz. No entanto, um estilo mais conversacional de entrada (voz ou texto) pode ser difícil de construir em cima de muitas APIs existentes

— APIs aptas para conversação

Tecnologias como Amazon Alexa, Google Voice e Siri reduziram drasticamente a barreira para interação com software através de voz. No entanto, um estilo mais conversacional de entrada (voz ou texto) pode ser difícil de construir em cima de muitas APIs existentes, especialmente quando se trata de um estilo de interação com mais estado, onde uma interação subsequente precisa estar ciente do contexto geral da conversa. Nesse estilo de interação, gostaríamos de perguntar, por exemplo, sobre trens de Manchester para Glasgow e, em seguida, sermos capazes de

perguntar “qual o horário do primeiro trem?” sem ter que dar o contexto da conversa novamente. Normalmente, esse contexto estaria presente na resposta inicial que enviamos de volta a um navegador, mas no caso de interfaces de voz, precisamos lidar com esse contexto em outro lugar. **APIS APTAS PARA CONVERSAÇÃO** podem ser um exemplo do padrão back-end para front-end, no qual o front-end é uma plataforma de bate-papo. Esse tipo de API pode lidar com os detalhes desse estilo de interação ao gerenciar estados de conversação enquanto chama os serviços subjacentes em nome do front-end de voz.

A adoção de nuvem e DevOps, embora tenha aumentado a produtividade dos times, que agora podem agir de forma mais rápida com dependência reduzida em equipes de operações e infraestrutura centralizadas, também restringiu equipes que não possuem habilidades para autogerenciar uma pilha completa de aplicação e operações. Algumas organizações têm abordado este desafio através da criação de **TIMES DE PRODUTO DE ENGENHARIA DE PLATAFORMA**. Esses times operam uma plataforma interna que permite que times de entrega realizem a implantação e a operação de sistemas com tempo e complexidade reduzidos. A ênfase aqui é em auto-serviço por API e ferramentas de suporte, com times de entrega ainda sendo responsáveis por dar suporte ao que eles implantam na plataforma. As organizações que consideram a criação de tal time de plataforma devem ser muito cautelosas para não criar acidentalmente um time de DevOps separado, nem devem simplesmente renomear sua estrutura de hospedagem e operações existente como uma plataforma.

A ANÁLISE SOCIAL DE CÓDIGO enriquece nossa compreensão da qualidade do código, sobrepondo o comportamento de uma pessoa desenvolvedora com a análise estrutural do código. Ela usa dados de sistemas de controle de versão, como frequência e tempo de alteração, bem como a pessoa que faz a alteração. Você pode decidir escrever seus próprios scripts para analisar tais dados ou usar ferramentas como CodeScene. CodeScene pode te ajudar a obter uma melhor compreensão de seus sistemas de software, identificando hotspots e subsistemas complexos e difíceis de manter; acoplamento entre subsistemas distribuídos por meio de acoplamento temporal, bem como a visão da lei de Conway na sua organização. Acreditamos que com tendências tecnológicas como sistemas distribuídos, microsserviços e times distribuídos, a dimensão social de nosso código seja vital para a compreensão holística da saúde de nossos sistemas.

A ideia de realidade virtual tem se feito presente por mais de 50 anos, e com sucessivas melhorias da tecnologia computacional, muitas ideias vêm sendo promovidas e exploradas. Acreditamos que agora estamos chegando a um momento decisivo. Headsets orientados ao consumidor chegaram ao mercado no ano passado com preços razoavelmente acessíveis, e placas de vídeo modernas têm poder suficiente para criar experiências imersivas com eles. Os headsets são direcionados principalmente a entusiastas de jogos eletrônicos, mas temos convicção de que eles abrirão muitas oportunidades para a **REALIDADE VIRTUAL ALÉM DOS JOGOS**. Times sem experiência com criação de jogos não devem subestimar o esforço e a habilidade necessários para criar bons modelos 3D e texturas convincentes.

A ideia de realidade virtual tem se feito presente por mais de 50 anos, e com sucessivas melhorias da tecnologia computacional, muitas ideias vêm sendo promovidas e exploradas. Acreditamos que agora estamos chegando a um momento decisivo.

— Realidade virtual além dos jogos

Sentimo-nos novamente na obrigação de alertar contra a criação de uma **INSTÂNCIA ÚNICA DE INTEGRAÇÃO CONTÍNUA (CI) PARA TODOS OS TIMES**. Embora consolidar e centralizar a infraestrutura de CI seja uma boa ideia na teoria, na prática não enxergamos maturidade suficiente nas ferramentas e produtos nessa área para atingir o resultado esperado. Times de entrega de software que precisam usar a oferta de CI centralizada regularmente têm longos atrasos por dependerem de um time central para executar tarefas de configuração pequenas, ou para resolver problemas na infraestrutura e nas ferramentas compartilhadas. Nesse estágio, continuamos recomendando que as

organizações limitem seus investimentos centralizados para estabelecer padrões, orientações e suporte para que times de entrega operem suas próprias infraestruturas de CI.

Infelizmente, muitas pessoas simplesmente configuram um servidor de CI e erroneamente assumem que estão "fazendo CI", quando na realidade estão perdendo todos os seus benefícios. Mesmo fazendo com que as pessoas se sintam bem, o "CI de faz de conta" que se segue falharia em qualquer teste de certificação de CI confiável.

— CI de faz de conta

Há muito tempo defendemos o uso de Integração Contínua (CI), e tivemos um papel pioneiro na criação de programas para servidores de CI que compilam projetos nos check-ins automaticamente. Quando bem utilizados, esses programas são executados como um processo daemon em uma linha principal compartilhada do projeto que as pessoas desenvolvedoras se comprometem a atualizar diariamente com suas modificações. O servidor de CI compila o projeto e executa testes abrangentes para garantir que todo o sistema de software esteja integrado e sempre em um estado pronto para lançamento, satisfazendo assim os princípios da entrega contínua. Infelizmente, muitas pessoas simplesmente configuram um servidor de CI e erroneamente assumem que estão "fazendo CI", quando na realidade estão perdendo todos os seus benefícios. Os modos de falha comuns incluem: executar CI em uma linha principal compartilhada mas com commits pouco frequentes, de forma que a integração não é realmente contínua; executar uma compilação com cobertura de teste ruim; permitir que a compilação fique em vermelho por longos períodos de tempo; ou executar CI fora da linha principal (em feature-branches), resultando em isolamento contínuo. Mesmo fazendo com que as pessoas se sintam bem, o "**CI DE FAZ DE CONTA**" que se segue falharia em qualquer teste de certificação de CI confiável.

Quando as implantações trimestrais ou mensais nas empresas eram considerados uma boa prática, era necessário manter um ambiente completo para executar ciclos de teste antes da implantação em produção. Estes **AMBIENTES DE TESTE DE INTEGRAÇÃO EMPRESARIAIS** (muitas vezes referidos como SIT ou Staging) são um gargalo comum para a entrega contínua hoje em dia. Os ambientes em si são frágeis e caros de se manter, muitas vezes com componentes que precisam de configuração manual por uma equipe de gerenciamento de ambiente separada. O teste nesse ambiente fornece feedback pouco confiável e lento, e o esforço do teste é duplicado com o que pode ser realizado em componentes isoladamente. Recomendamos que as organizações criem incrementalmente um caminho independente de componentes-chave para produção. Entre as técnica mais importantes estão os testes de contrato guiados pelo consumidor, o desacoplamento entre publicação e implantação, o foco no tempo médio para a recuperação e os testes em produção.

Quando SOAP dominava a indústria de software corporativo, a prática de gerar código de cliente a partir de especificações WSDL era uma prática aceita e até mesmo incentivada. Muitas vezes, infelizmente, o código gerado era complexo, não-testável, difícil de modificar e frequentemente não funcionava em todas as plataformas de implementação. Com o advento do REST, descobrimos ser melhor evoluir clientes de API utilizando o padrão de leitura tolerante para extrair e processar apenas os campos necessários. Recentemente observamos um perturbador retorno aos velhos hábitos com pessoas desenvolvedoras gerando código a partir de especificações de API escritos em Swagger ou RAML, prática que chamamos de **GERAÇÃO DE CÓDIGO BASEADA EM ESPECIFICAÇÕES**. Embora essas ferramentas sejam muito úteis para conduzir o design de APIs e para extrair documentação, nós advertimos contra o tentador atalho de simplesmente gerar código de cliente diretamente dessas especificações. São grandes as chances de que esse código será difícil de testar e manter.

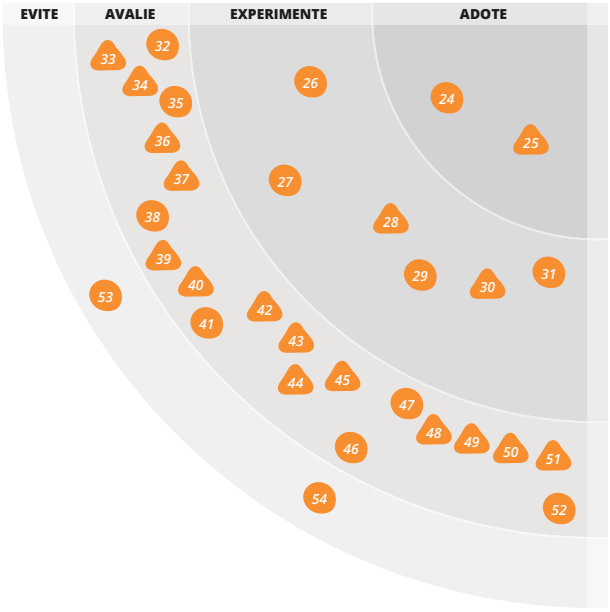
PLATAFORMAS

ADOTE
24. HSTS
25. Módulos de segurança do Linux

EXPERIMENTE
26. Apache Mesos
27. Auth0
28. AWS Device Farm *NOVO*
29. AWS Lambda
30. OpenTracing *NOVO*
31. Unity além dos jogos

AVALIE
32. .NET Core
33. Amazon API Gateway
34. api.ai *NOVO*
35. Cassandra com cuidado
36. Compreensão de imagem baseada na nuvem *NOVO*
37. DataStax Enterprise Graph *NOVO*
38. Electron
39. Ethereum
40. Hyperledger *NOVO*
41. IndiaStack
42. Kafka Streams *NOVO*
43. Keycloak *NOVO*
44. Mesosphere DCOS
45. Mosquitto *NOVO*
46. Nuance Mix
47. OpenVR
48. PlatformIO *NOVO*
49. Tango *NOVO*
50. Plataformas conversacionais *NOVO*
51. WebVR *NOVO*
52. wit.ai

EVITE
53. CMS como plataforma
54. API gateways excessivamente ambiciosos



O Princípio do Menor Privilégio nos encoraja a restringir os componentes de software para acessarem apenas os recursos de que precisam. Por padrão, no entanto, um processo Linux pode fazer qualquer coisa que seu usuário em execução pode fazer—desde a conexão com portas arbitrárias até a criação de novos shells. O framework **MÓDULOS DE SEGURANÇA DO LINUX** (LSM), que permite que extensões de segurança sejam conectadas ao kernel, tem sido usado para implementar controle de acesso obrigatório no Linux. SELinux e AppArmor são as implementações predominantes e mais conhecidas compatíveis com LSM que acompanham o kernel. Recomendamos que os times aprendam a usar um desses frameworks de segurança (e é por isso que o colocamos em Adote). Eles ajudam os times a avaliar questões sobre quem tem acesso a quais recursos em servidores compartilhados, incluindo serviços internos. Essa abordagem conservadora para o gerenciamento de acesso ajudará os times a construir segurança em seus processos no ciclo de desenvolvimento de software.

AMAZON API GATEWAY permite que as pessoas desenvolvedoras exponham serviços de API a clientes da Internet. Ele oferece os recursos habituais de gateway de API, incluindo gerenciamento de tráfego, monitoramento, autenticação e autorização. Nossos times têm usado esse serviço como frente para outros serviços da AWS tais como AWS Lambda como parte de arquiteturas sem servidor. Por outro lado, temos enfrentado mais desafios usando-o como um gateway de propósito mais geral, como frente para endpoints HTTP/HTTPS executando em EC2—onde nos frustramos por uma falta da interoperabilidade com VPCs e dificuldade em estabelecer a autenticação por certificado do cliente com o Gateway. Devido a essa experiência mista, gostaríamos de aconselhar os times a experimentar a utilização do AWS API Gateway com Lambda, mas avaliar a adequação ao usá-lo em uma configuração mais geral.

O grande número de dispositivos móveis torna quase impossível para empresas testarem seus aplicativos móveis em todos eles. É aí que entra o **AWS DEVICE FARM**, um serviço de testes de aplicativos que permite que você execute e interaja com seus aplicativos Android, iOS e web em uma grande variedade de dispositivos físicos hospedados simultaneamente na nuvem. Registros detalhados, gráficos de desempenho e capturas de tela são gerados durante cada execução para fornecer feedback geral e específico do dispositivo. O serviço oferece muita flexibilidade ao permitir que o estado e a configuração de cada dispositivo sejam alterados para reproduzir cenários de teste muito específicos. Nossas equipes estão usando o AWS Device Farm para executar testes de ponta a ponta em dispositivos com a maior base de instalação para seus aplicativos.

Conforme as aplicações monolíticas estão sendo substituídas por ecossistemas mais complexos de (micro) serviços, rastrear requisições que percorrem vários serviços estão se tornando regra.

— OpenTracing

Conforme as aplicações monolíticas estão sendo substituídas por ecossistemas mais complexos de (micro) serviços, rastrear requisições que percorrem vários serviços estão se tornando regra. Felizmente, **OPENTRACING** está rapidamente se tornando o padrão para o rastreamento distribuído. Desenvolvido por Uber, Apple, Yelp e vários outras grandes empresas, ele suporta múltiplos rastreadores como Zipkin, Instana e Jaeger. Atualmente, o OpenTracing fornece implementações independentes de plataforma em seis linguagens: Go, JavaScript, Java, Python, Objective-C e C ++.

Paralelamente ao aumento recente de chatbots e plataformas de voz, vimos uma proliferação de ferramentas e plataformas como **API.AI**, que fornece um serviço para extração de intenção de texto e gerenciamento de fluxo de conversação ao qual você pode se conectar. Recentemente adquirida pelo Google, esta oferta de "compreensão de linguagem natural como um serviço" compete com outras plataformas nessa área, tais como wit.ai e Lex da Amazon.

Compreensão de imagem costumava ser uma arte obscura e exigia uma equipe de cientistas de dados no local. Nos últimos anos, no entanto, estamos mais perto de resolver problemas como classificação/categorização facial e de imagem, comparações faciais, identificação de marcos faciais e reconhecimento facial. A **COMPREENSÃO DE IMAGEM BASEADA NA NUVEM** fornece acesso a capacidades de aprendizagem de máquina por meio de serviços como Amazon Rekognition, Microsoft Computer Vision API e a Google Cloud Vision API, que podem complementar os aplicativos de realidade aumentada (RA) e qualquer coisa que envolva rotular e classificar fotos.

Tivemos alguns sucessos iniciais com o **DATASTAX ENTERPRISE GRAPH** (DSE Graph) para a manipulação de grandes bancos de dados orientados a grafos. Construído em cima do Cassandra, o DSE Graph tem como alvo os grandes conjuntos de dados onde o nosso favorito de longa data, Neo4j, começa a mostrar algumas limitações. Essa escala tem suas compensações; por exemplo, você perde as transações ACID e a natureza livre de esquemas em tempo de execução do Neo4j, mas o acesso às tabelas estruturais do Cassandra, a integração do Spark para cargas de trabalho analíticas e a poderosa linguagem de consulta TinkerPop/Gremlin tornam essa uma opção que vale a pena considerar.

A excitação com blockchain e criptomoedas parece ter atingido seu pico, já que os anúncios de grande impacto nessa área têm sido cada vez mais raros, e é esperado que alguns dos esforços mais especulativos se extingam com o tempo. Um dos blockchains, **ETHEREUM** apesar de não ser popular entre fãs de blockchain, tem aparecido em cada vez mais iniciativas. Ethereum é um blockchain público com uma linguagem de programação embutida que permite que "contratos inteligentes" sejam criados. Estes são movimentos algorítmicos do "ether" (criptomoeda do Ethereum) em resposta à atividade ocorrendo no blockchain. R3CEV, o consórcio de tecnologia de blockchain para bancos, construiu suas primeiras provas de conceito no Ethereum. O Ethereum foi usado para construir uma Organização Autônoma Distribuída (Distributed Autonomous Organization ou DAO)—uma das primeiras

"corporações algorítmicas"—, embora um assalto recente de 150 milhões de dólares em Ether demonstre que blockchain e criptomoedas ainda são o Velho Oeste do mundo da tecnologia.

HYPERLEDGER é uma plataforma construída em torno de tecnologias blockchain. Consiste em uma implementação blockchain chamada Fabric e outras ferramentas associadas. Desconsiderando a excitacão envolvendo blockchain, nossas equipes acharam fácil começar a trabalhar com essas ferramentas. O fato de ser uma plataforma de código aberto apoiada pela Fundação Linux também contribui com a nossa alta expectativa em relação à Hyperledger.

KAFKA STREAMS é uma biblioteca leve para a construção de aplicações de streaming. Ela foi projetada com o objetivo de simplificar o processamento de streams o suficiente para torná-lo facilmente acessível como um modelo de programação predominante de aplicativos para serviços assíncronos. Ela pode ser uma boa alternativa para cenários em que você deseja aplicar um modelo de processamento de stream a seu problema sem incluir a complexidade de executar um cluster (geralmente introduzido por estruturas de processamento de stream completos).

Em uma arquitetura de microserviços ou qualquer outra arquitetura distribuída, uma das necessidades mais comuns é proteger os serviços ou APIs através de recursos de autenticação e autorização. É aí que a Keycloak entra. **KEYCLOAK** é uma solução de gerenciamento de acesso e identidade de código aberto que facilita a segurança de aplicativos ou microserviços com pouco ou nenhum código. Ela já vem pronta para uso com suporte para logon único, login social e protocolos padrão, como OpenID Connect, OAuth2 e SAML.

MESOSPHERE DCOS é uma plataforma baseada em Mesos que abstrai sua infraestrutura subjacente para aplicações conteinerizadas, bem como para aplicações que você não quer executar dentro do Docker. Pode ser um exagero para implantações mais modestas, mas temos começado a ver sucesso tanto com a versão comercial quanto com a versão open source. Nós particularmente gostamos do fato de que ela facilita portabilidade entre diferentes provedores de nuvem, bem como hardware dedicado, e que para tarefas conteinerizadas você não fica dependente de um framework de orquestração de contêiner. Embora upgrades possam ser um pouco mais complexos do que gostaríamos, a plataforma de maneira geral está se estabilizando.

Em nossa experiência—para soluções de Internet das Coisas (IoT) em que muitos dispositivos se comunicam entre si e/ou com um hub de dados central—o protocolo de conectividade MQTT provou-se eficaz. Nós também gostamos do agente **MOSQUITTO** MQTT. Pode não satisfazer todas as necessidades, especialmente no que diz respeito à escalabilidade, mas a sua natureza compacta e fácil configuração o tornam ideal para propósitos de desenvolvimento e testes.

O **PLATFORMIO** fornece um rico ecossistema para o desenvolvimento de aplicações relacionadas a internet das coisas, contando com compilação multiplataforma, gerenciamento de biblioteca e boa integração com IDEs existentes. O autocompletar inteligente de código e o linter de código inteligente com o terminal interno e o monitor de porta serial aumentam consideravelmente a experiência de quem desenvolve. Ele também organiza e mantém milhares de bibliotecas e fornece um gerenciador de dependência limpo com versões semânticas para facilitar o desenvolvimento de IoT. Começamos a usar o PlatformIO em alguns projetos de IoT e gostamos bastante por sua simplicidade e amplo suporte a plataformas e painéis.

Assim como realidade virtual (RV), que tem uma barreira relativamente alta de entrada devido aos requisitos de hardware e ao esforço para criar mundos virtuais, realidade alternativa (RA) e realidade mista (RM), também passaram a predominar no ano passado. O jogo Pokémon Go forneceu evidências de que os smartphones normais são suficientes para criar experiências convincentes de RA/RM. **TANGO** é uma nova tecnologia de sensor de hardware para telefones celulares que aumenta ainda mais as possibilidades

de RA/RM em telefones. Ele permite que os aplicativos adquiram medições detalhadas em 3D do ambiente em volta do usuário, para que os objetos virtuais possam ser colocados e renderizados de forma mais convincente no feed da câmera. Os primeiros telefones com a tecnologia Tango já estão disponíveis.

PLATAFORMAS CONVERSACIONAIS como Amazon Alexa e Google Home estão em alta; alguns até prenunciam a onipresença da interface de voz conversacional. Já estamos integrando interfaces de conversação em produtos e vendo como essa nova interação impacta a concepção de interfaces de usuário. A Alexa foi especificamente construída sem uma interface gráfica desde o início e trata a interface conversacional com o usuário como conceito de primeira classe. Mas ainda é muito cedo para acreditar na onda, e esperamos que mais grandes empresas entrem no jogo.

Já estamos integrando interfaces de conversação em produtos e vendo como essa nova interação impacta a concepção de interfaces de usuário.

— Plataformas conversacionais

WEBVR é uma API experimental em JavaScript que permite o acesso a dispositivos de realidade virtual (RV) através do seu navegador. Ela vem obtendo suporte da comunidade e está disponível por meio de compilações noturnas, bem como em algumas versões de lançamento. Se você está procurando construir experiências RV em seu navegador, esse é um ótimo lugar para começar. Essa tecnologia, juntamente com ferramentas como Three.js, A-Frame, ReactVR, Argon.js e Awe.js traz experiências de RA para o navegador. A quantidade de ferramentas nessa área, juntamente com comissões de padrões da Internet, poderá ajudar a promover uma adoção mais forte de RA e RV.

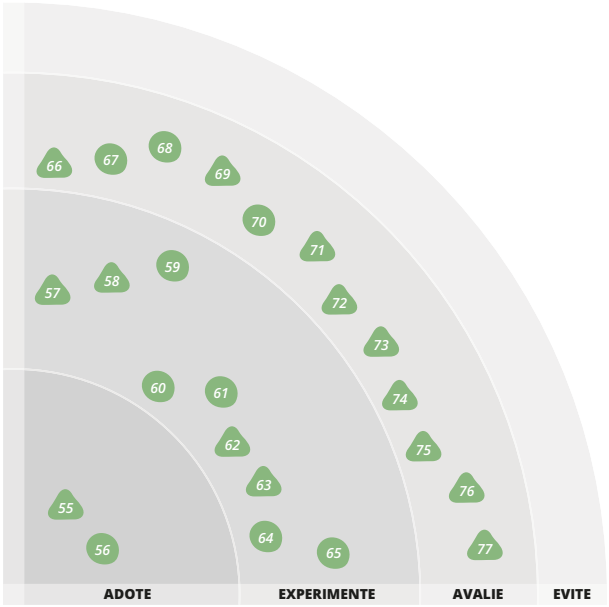
FERRAMENTAS

ADOTE
55. fastlane
56. Grafana

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

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

EVITE



Quem desenvolve aplicações web tem facilidade para simplificar e automatizar fluxos de trabalho diversos. Essas pessoas podem escolher dentre uma variedade de soluções para ajudar a automatizar os processos de lançamento. No entanto, ao desenvolver para dispositivos móveis, estamos lidando com dois sistemas operacionais que possuem maneiras diferentes de construir, testar, distribuir, gerar capturas de tela, assinar e distribuir aplicativos. Para ajudar a aliviar a dificuldade, nossos times adotaram o **FASTLANE** como a ferramenta de escolha para automatizar o processo de lançamento de aplicações iOS e Android. Usando configurações simples e pipelines múltiplos, eles podem alcançar entrega contínua para desenvolvimento móvel.

AIRFLOW é uma ferramenta para criar programaticamente, agendar e monitorar pipelines de dados. Ao tratar Gráficos Acíclicos Dirigidos (GADs) como código, ele incentiva pipelines de dados que podem ser mantidos, versáteis e testáveis. Nós alavancamos essa configuração em nossos projetos para poder criar pipelines dinâmicos que resultaram em fluxos de trabalho de dados enxutos e explícitos. Airflow facilita a definição dos seus operadores e executores e amplia a biblioteca para que ela se ajuste ao nível de abstração adequado ao seu ambiente.

Nossos times adotaram o fastlane como a ferramenta de escolha para automatizar o processo de lançamento de aplicações iOS e Android.

— fastlane

Desde a sua introdução em 2005, o MSBuild tem sido o principal sistema de compilação no ecossistema .NET. No entanto, ele sofre de muitas das mesmas deficiências que já apontamos no Maven. A comunidade .NET começou a desenvolver alternativas ao MSBuild que são mais flexíveis, fáceis de manter e evoluem de

forma mais fluida à medida que um projeto cresce. Duas destas alternativas são **CAKE E FAKE**. Cake usa uma DSL desenvolvida em C#, enquanto Fake usa F#. Cada um desses sistemas observou um crescimento significativo ao longo do último ano e provaram ser uma alternativa viável ao MSBuild para orquestrar tarefas habituais de compilação de projetos .NET.

A comunidade .NET começou a desenvolver alternativas ao MSBuild que são mais flexíveis, fáceis de manter e evoluem de forma mais fluida à medida que um projeto cresce.

— Cake e Fake

SCIKIT-LEARN não é uma ferramenta nova (está se aproximando do décimo aniversário). A novidade é a taxa de adoção de ferramentas e técnicas de aprendizagem de máquina fora da academia e de grandes empresas de tecnologia. Oferecendo um conjunto robusto de modelos e um rico conjunto de funcionalidades, o Scikit-learn desempenha um papel importante no sentido de tornar os conceitos e capacidades de aprendizagem de máquina mais acessíveis a um público mais amplo (e muitas vezes não-especialista).

O popular **SERVERLESS FRAMEWORK** fornece ferramentas para inicialização e implantação de aplicações sem servidor, usando primariamente o AWS Lambda e outras ofertas da AWS. O Serverless Framework fornece suporte a templates em JavaScript, Python, Java e C#, e tem uma comunidade ativa que contribui com plug-ins que estendem o framework. Como uma alternativa à AWS Lambda, o framework também suporta o projeto OpenWhisk da incubadora da Apache.

AMAZON REKOGNITION é uma das ferramentas de compreensão de imagens baseadas em nuvem que mencionamos em outra parte deste Radar. O que nós gostamos sobre ela é que a Amazon tomou uma abordagem um tanto quanto original para tornar os rostos anônimos para a AWS (usando GUIDs), amenizando algumas das preocupações de privacidade que surgem com o reconhecimento facial.

A combinação do AWS Lambda com o Amazon API Gateway teve um grande impacto sobre como implantamos serviços e APIs. No entanto, mesmo nessa configuração sem servidor, a quantidade de configuração necessária para conectar coisas em conjunto não é trivial. **CLAUDIA** é uma ferramenta que automatiza a implantação de funções AWS Lambda escritas em JavaScript com configurações de gateway API associadas. Ela fornece configurações pré-determinadas razoáveis, e nossos times descobriram que ela permite que eles comecem a trabalhar rapidamente com microserviços baseados em Lambda.

Como um negócio lida com a autonomia de times de entrega enquanto se certifica de que suas soluções implantadas são seguras e compatíveis? Como garantir que os servidores, uma vez implantados, permaneçam seguros e compatíveis ao longo de sua vida operacional? Esses são os problemas que o InSpec tenta resolver. **INSPEC** é uma ferramenta de testes de infraestrutura inspirada no Serverspec, mas com modificações que tornam a ferramenta mais útil para profissionais de segurança que precisam garantir a compatibilidade entre milhares de servidores. Testes individuais podem ser combinados em perfis de segurança completos e executados remotamente a partir de uma linha de comando. O InSpec é útil para desenvolvimento, mas se estende a testes em infraestruturas de produção implantadas continuamente, avançando para o QA na produção.

MOLECULE é projetado para auxiliar no desenvolvimento e em testes dos papéis do Ansible. Ao construir uma estrutura base para executar testes dos papéis do Ansible em uma máquina virtual ou contêiner de sua escolha, não precisamos configurar nosso ambiente de teste manualmente. Molecule potencializa Vagrant, Docker e OpenStack para gerenciar máquinas virtuais/contêineres e suporta Serverspec, Testinfra ou Goss para executar testes. As etapas padrões no modelo da sequência de instalação incluem: gerenciamento de máquina virtual, Ansible linting, teste de idempotência e teste de convergência. Embora seja um projeto bastante novo, vemos um grande potencial para seu uso.

Como qualquer fã de Emacs irá dizer, Emacs é mais do que um editor de texto; é uma plataforma para aplicações de mapeamento de caracteres. Ao longo dos últimos anos, tem havido vários avanços nessa plataforma, mas achamos que o **SPACEMACS** merece atenção especial. O Spacemacs fornece uma introdução à plataforma Emacs, com uma nova interface de usuário, camadas de personalização simplificadas e uma triagem cuidadosa de pacotes Emacs. Um dos objetivos do projeto é ser o melhor dos dois mundos, combinando a interface do usuário do Vim com a reprogramação interna do Emacs. Consideramos que as ferramentas de produtividade são uma parte vital do desenvolvimento de software eficaz e, se você não considerou Emacs por um tempo, sugerimos que você dê uma olhada em como a Spacemacs repensa essa clássica plataforma de desenvolvimento.

SPACY é uma biblioteca de processamento de linguagem natural (PNL) escrita em Python. É uma biblioteca de alto desempenho, destinada ao uso em produção, que aplica modelos de PNL adequados para o processamento de texto que muitas vezes se confunde em emoticons e pontuações inconsistentes. Ao contrário de outros frameworks PNL, spaCy é uma biblioteca plugável e não uma plataforma. Destina-se a aplicações em produção ao invés de treinamento de modelo para pesquisa. Ela interage bem com TensorFlow e com o resto do ecossistema Python AI. Utilizamos spaCy no contexto corporativo para construir um mecanismo de pesquisa que usa consultas em linguagem humana e ajuda os usuários a tomar decisões de negócios.

A Netflix abriu o código da **SPINNAKER**, sua plataforma de entrega contínua de microserviços. Comparada com outras plataformas de CI/CD, a Spinnaker implementa o gerenciamento de clusters e a implantação de imagens pré-criadas na nuvem como recursos de primeira classe. Ela suporta a implantação e o gerenciamento de cluster prontos para uso para vários provedores de nuvem, como Google Cloud Platform, AWS e Pivotal Cloud Foundry. Você pode integrar Spinnaker com Jenkins para executar um trabalho de compilação no Jenkins. Nós gostamos da abordagem opinativa de Spinnaker para implantar microserviços na nuvem—com exceção do fato de que os pipelines de Spinnaker são criados por meio de uma interface de usuário (UI) e não podem ser configurados como código.

Dado o amplo uso de ferramentas de infraestrutura nos dias de hoje, não é surpresa o aumento da adoção de infraestrutura como código pelos projetos atuais. Com essa tendência, vem a necessidade de testar esse código. Com **TESTINFRA** você pode testar o estado real de seus servidores configurados manualmente ou por ferramentas como Ansible, Puppet e Docker. Testinfra pretende ser um equivalente a Serverspec em Python e é escrito como um plug-in para o mecanismo de teste Pytest.

Dado o amplo uso de ferramentas de infraestrutura nos dias de hoje, não é surpresa o aumento da adoção de infraestrutura como código pelos projetos atuais.

— Testinfra

YARN é um novo gerenciador de pacotes que substitui o fluxo de trabalho existente pelo cliente npm, permanecendo compatível com o registro npm. Com o cliente npm, podemos terminar com uma estrutura de árvore diferente no node_modules com base na ordem em que as dependências são instaladas. Essa natureza não-determinística pode causar problemas de "funciona na minha máquina". Ao quebrar as etapas de instalação em resolução, busca e vinculação, Yarn evita esses problemas usando algoritmos determinísticos e arquivos de lock, garantindo instalações consistentes. Também vimos construções significativamente mais rápidas em nosso ambiente de integração contínua pelo fato do Yarn colocar em cache todos os pacotes que baixa.

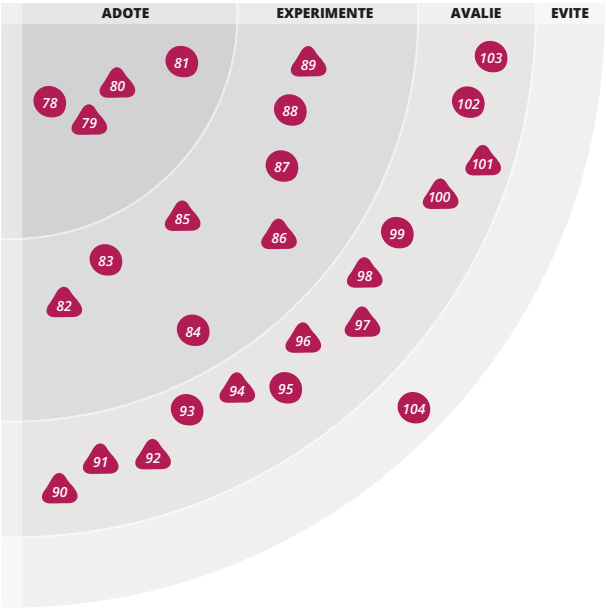
LINGUAGENS & FRAMEWORKS

ADOTE
78. Ember.js
79. Python 3
80. ReactiveX
81. Redux

EXPERIMENTE
82. Avro ***NOVO***
83. Elixir
84. Enzyme
85. Hangfire ***NOVO***
86. Nightwatch ***NOVO***
87. Phoenix
88. Quick e Nimble
89. Vue.js

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

EVITE
104. AngularJS



PYTHON 3 introduziu muitos recursos úteis que não são compatíveis com Python 2.x. Também removeu vários recursos do Python 2.x que foram mantidos para compatibilidade com versões anteriores, tornando Python 3 mais fácil de aprender e usar e mais consistente com o resto da linguagem. Nossa experiência em usar Python 3 em domínios como aprendizagem de máquina e desenvolvimento de aplicativos web demonstra que tanto a linguagem quanto a maioria de suas bibliotecas de suporte amadureceram o suficiente para adoção. Conseguimos fazer forks e corrigir problemas menores de bibliotecas existentes ou evitamos usar bibliotecas do Python 2.x incompatíveis que tinham sido abandonadas. Se você está desenvolvendo em Python, recomendamos fortemente que você use o Python 3.

Sistemas distribuídos frequentemente utilizam comunicação multithreading baseada em eventos e I/O não-bloqueadoras para melhorar a eficiência geral do sistema. Essas técnicas de programação impõem desafios como threading de baixo nível, sincronização, segurança de thread, estruturas de dados simultâneas e I/O não bloqueadoras. A biblioteca de código aberto **REACTIVEX** abrange com maestria essas preocupações, fornece a estrutura da aplicação requerida e estende o padrão observável em fluxos de eventos assíncronos. ReactiveX também tem uma comunidade ativa e suporta uma crescente lista de linguagens, sendo a mais recente adição RxSwift. Ela também implementa ligação a plataformas móveis e desktop.

A biblioteca de código aberto REACTIVEX abrange com maestria essas preocupações—threading de baixo nível, sincronização, segurança de thread, estruturas de dados simultâneas e I/O não bloqueadoras—, fornece a estrutura da aplicação requerida e estende o padrão observável em fluxos de eventos assíncronos.

— ReactiveX

AVRO é um framework para serialização de dados. Ao armazenar o esquema junto com o conteúdo da mensagem, ele incentiva a evolução deste esquema. Produtores podem editar nomes de campos, adicionar novos campos ou excluir campos existentes, e o Avro garante que os clientes continuarão a consumir as mensagens. Ter um esquema permite que cada dado seja escrito sem sobrecarga resultando em dados compactos e processamento mais rápido. Embora a troca de mensagens sem estrutura entre produtor e consumidor seja flexível, vimos times enfrentando problemas com mensagens incompatíveis não-processadas na fila durante implantações. Usamos o Avro em vários projetos e recomendamos o uso apenas no envio mensagens não estruturadas.

O Hangfire é fácil de usar e flexível, e utiliza um estilo funcional. Particularmente interessante é a sua capacidade de salvar o estado de uma tarefa para que ele possa ser retomado quando um aplicativo é reiniciado após um bloqueio ou desligamento.

— Hangfire

Um problema comum no desenvolvimento de aplicativos é como programar tarefas que são executadas periodicamente fora do processo principal ou quando determinadas condições são atingidas. O problema fica mais complicado quando eventos inesperados, como desligamentos de aplicativos, ocorrem. A framework **HANGFIRE**, como nossos times descobriram, pode fazer isso e muito mais no ambiente .NET. O Hangfire é fácil de usar e flexível, e utiliza um estilo funcional. Particularmente interessante é a sua capacidade de salvar o estado de uma tarefa para que ele possa ser retomado quando um aplicativo é reiniciado após um bloqueio ou desligamento.

NIGHTWATCH é uma framework que permite que testes de aceitação automatizados para aplicativos baseados em navegador sejam criados em JavaScript e executados em Node.js. O Nightwatch permite que os testes sejam definidos usando uma API fluente, que pode então ser executada contra um servidor Selenium/WebDriver. No caso de aplicativos de página única ou outras páginas pesadas em JavaScript, isso permite que os testes automatizados sejam criados e executados na mesma linguagem e no mesmo ambiente que a maioria do código.

No mundo em constante mudança de frameworks de front-end JavaScript, um dos favoritos emergentes parece ser **VUE.JS**. Vue.js é uma alternativa mais leve para AngularJS. É projetada para ser uma biblioteca muito flexível—e menos opinativa—que oferece um conjunto de ferramentas para a construção de interfaces web interativas em torno de conceitos como modularidade, componentes e fluxo de dados reativos. Possui uma curva de aprendizagem baixa, o que a torna interessante para quem está aprendendo a desenvolver. Observe, porém, que Vue.js não é uma framework completa; tem como foco apenas a camada de visualização e, portanto, é fácil de integrar com outras bibliotecas ou projetos existentes.

No Radar anterior, movemos AngularJS para o anel Evite (onde permanece nesta edição). Quando se trata de **ANGULAR 2**, vemos mensagens distintas. Ao longo do ano passado, algumas equipes da ThoughtWorks usaram o Angular 2 com sucesso e o consideram uma escolha sólida. No entanto, Angular 2 é uma reescrita, não uma evolução de AngularJS, e mudar de AngularJS para Angular 2 não é muito diferente de mudar de AngularJS para outro framework. Visto que, na nossa experiência, há concorrentes superiores como React.js, Ember.js e Vue.js, ainda estamos hesitantes em dar a Angular 2 uma forte recomendação. Queremos ressaltar, no entanto, que não se trata de uma escolha ruim, especialmente se você adotou TypeScript.

O **CAFFE** é uma biblioteca de código aberto para aprendizagem profunda criada pelo Berkeley Vision and Learning Center. Concentra-se principalmente em redes convolucionais para aplicações de visão computacional. Caffe é uma escolha sólida e popular para tarefas relacionadas à visão computacional e você pode baixar muitos modelos bem-sucedidos feitos por usuários do Caffe Model Zoo para produtividade imediata. Assim como Keras, Caffe é uma API com base

Python. Em Keras, no entanto, modelos e componentes são objetos criados diretamente em código Python, enquanto modelos Caffe são descritos por arquivos de configuração Protobuf. As duas abordagens tem seus prós e contras, e a conversão entre as duas também é possível.

DEEPLEARNING.SCALA é um conjunto de ferramentas de aprendizagem profunda de código aberto em Scala criado por colegas na ThoughtWorks. Esse projeto tem nos animado porque ele usa programação funcional diferenciável para criar e compor redes neurais; uma pessoa desenvolvedora simplesmente escreve código em Scala com tipos estáticos. O DeepLearning.scala atualmente suporta tipos básicos como float, double, matrizes N-dimensionais aceleradas por GPU, bem como tipos de dados algébricos. Esperamos com ansiedade futuras versões do conjunto de ferramentas que promete suportar funções de ordem superior e treinamento distribuído no Spark.

INSTANA é mais um estreante no povoado espaço de gerenciamento de desempenho de aplicativos. O fato de que ele foi construído do zero para as arquiteturas nativas de nuvem diferencia Instana de muitos de seus concorrentes. Os recursos incluem descoberta dinâmica, rastreamento distribuído e saúde do serviço, além da capacidade de "mudar o tempo" de visualização da sua infraestrutura para o momento em que ocorreu um incidente. Resta saber se este produto pode ganhar junto à variedade de projetos de código aberto—como Consul, Prometheus e as implementações do OpenTracing–que fazem a mesma coisa; no entanto, vale a pena analisar se você precisa de uma solução fora da caixa.

KERAS é uma interface de alto nível em Python para a construção de redes neurais. Criada por um engenheiro do Google, Keras possui código aberto e funciona tanto com TensorFlow quanto com Theano. Ela fornece uma interface surpreendentemente simples para criar poderosos algoritmos de aprendizagem profunda para treinar em CPUs ou GPUs. Keras é bem projetada tendo modularidade, simplicidade e extensibilidade em mente. Ao contrário de bibliotecas como Caffe, Keras suporta arquiteturas de rede mais gerais, tais como redes recorrentes, tornando-a de forma geral mais útil para análise de texto, PLN e aprendizagem de máquinas em geral. Se a visão computacional, ou qualquer outro ramo especializado de aprendizagem de máquina, é a sua principal preocupação, Caffe pode ser uma escolha acertada. No entanto, se você procura aprender uma framework simples mas poderosa, Keras deve ser a sua escolha.

KNET.JL é uma framework de aprendizagem profunda da Universidade Koç implementada em Julia por Deniz Yuret e colaboradores. Ao contrário dos compiladores de geração de gradientes, como Theano e TensorFlow, que forçam os usuários a entrar em uma mini-linguagem restrita, o Knet permite a definição e o treinamento de modelos de aprendizado de máquina usando a plena potência e expressividade de Julia. O Knet utiliza grafos computacionais dinâmicos gerados em tempo de execução para a diferenciação automática de praticamente qualquer código Julia. Nós realmente gostamos do suporte de operações de GPU por meio do tipo KnetArray, e caso você não tenha acesso a uma GPU, a equipe por trás do Knet também mantém uma Amazon Machine Image (AMI) pré-configurada para que você possa avaliá-lo na nuvem.

A linguagem de programação **KOTLIN** está em muitas das nossas listas de coisas para avaliar esse ano, e algumas pessoas já a usaram com sucesso em produção. É uma linguagem JVM de código aberto da JetBrains. Quem desenvolve em Swift para dispositivos móveis gosta, visto que é sintaticamente mais próxima de Swift e igualmente concisa. As pessoas que desenvolvem em Java têm apreciado sua interoperabilidade perfeita com a linguagem e com as ferramentas Java e achado mais fácil de aprender do que Scala. Kotlin suporta conceitos de programação funcional, mas com menos recursos do que Scala. Algumas pessoas que gostam de tipos estáticos com o compilador capturando defeitos de ponteiros nulos perceberam que estavam escrevendo menos testes para estes casos.

Times construindo sistemas compostos por microsserviços precisam pensar sobre técnicas de coordenação tais como descoberta de serviços, balaceamento de carga, circuit breakers e health check.

— Spring Cloud

POSTCSS é uma framework JavaScript baseada em Node.js para operar em uma representação abstrata baseada em árvore de sintaxe de documentos CSS com um rico ecossistema de plug-ins. Muitas vezes considerado incorretamente como um pré-processador (como SaaS ou Less), descobrimos que o verdadeiro poder do PostCSS vem do número de coisas que podem ser feitas com o rico conjunto de plug-ins, que inclui verificação de estilo (o plug-in stylelint), compilação cruzada (o plug-in sugarss), name-mangling para evitar a colisão de seletor (o plug-in modules), geração de código CSS padrão (o autoprefixer plug-in), minificação e muitos outros. Apesar dos diferentes níveis de maturidade dos plug-ins, o PostCSS permanece uma framework simples e poderosa para o tratamento de CSS como uma linguagem completa para o desenvolvimento de front-end.

Times construindo sistemas compostos por microsserviços precisam pensar sobre técnicas de coordenação tais como descoberta de serviços, balaceamento de carga, circuit breakers e health check. Muitas dessas técnicas exigem que os times configurem ferramentas, o que nem sempre é trivial. O projeto **SPRING CLOUD** provê ferramentas para que pessoas desenvolvedoras utilizem essas técnicas no ambiente Spring. Essas ferramentas suportam Consul, ZooKeeper e a stack Netflix OSS, todas ferramentas que gostamos. De maneira simples, é mais fácil fazer a coisa certa com esse conjunto de ferramentas. Embora ainda tenhamos nossa preocupação usual com Spring, visto que muito da complexidade é escondida, você pode considerar Spring Cloud se já estiver no ecosistema e precisar resolver esses tipos de problemas.

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

INSCREVA-SE AGORA

thght.works/Sub-PT

ThoughtWorks®

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

Fundada há mais de 20 anos, a ThoughtWorks se tornou uma empresa com mais de 4000 pessoas, incluindo uma área de produtos que desenvolve ferramentas pioneiras para times de software. A ThoughtWorks tem 40 escritórios em 14 países: África do Sul, Alemanha, Austrália, Brasil, Canadá, China, Equador, Espanha, Estados Unidos, Índia, Itália, Reino Unido, Singapura e Turquia.

thoughtworks.com