ThoughtWorks®

TECHNOLOGY RADAR *ABRIL '16*

Nossas ideias sobre tecnologias e tendências que estão moldando o futuro

thoughtworks.com/radar

O QUE HÁ DE NOVO?

Aqui estão os assuntos em destaque nessa edição:

CÓDIGO ABERTO COMO UM SUBPRODUTO VIRTUOSO

Alguns dos softwares mais influentes que aparecem em nosso radar vêm de empresas cuja missão primordial não é criar ferramentas de software. Várias das nossas notas no radar vêm do Facebook, que não é considerada uma empresa tradicional de desenvolvimento de ferramentas de software. Diferentemente do que acontecia antes, hoje muitas empresas abrem o código de importantes componentes de software — para atrair as pessoas e para credenciar elas próprias. Isso cria um ciclo virtuoso: ferramentas inovadoras de código aberto atraem pessoas talentosas, que por sua vez são mais propícias a inovar. Como um efeito colateral, os frameworks e bibliotecas dessas empresas estão entre os mais influentes da indústria. Isso representa uma grande mudança no ecossistema de desenvolvimento de software e é mais uma evidência da eficácia do software de código aberto — no contexto adequado (nossa recomendação sobre inveja de escalabilidade web ainda se aplica).

DECIFRANDO O ENIGMA DE PAAS

Muitas grandes organizações vêem a Nuvem e Plataforma como Serviço (Platform as a Service ou PaaS) como um caminho óbvio para padronizar infraestruturas, facilitar implantações e operações e tornar pessoas desenvolvedoras mais produtivas. Mas ainda é cedo, a definição de PaaS continua sendo nebulosa e muitas abordagens de PaaS são incompletas ou prejudicadas pela imaturidade de frameworks e ferramentas de suporte. Algumas soluções de PaaS dificultam a execução de tarefas que são mais facilmente executadas com uma solução simples de Infraestrutura como Serviço (Infrastructure as a Service ou IaaS), como usar um Service Locator personalizado ou uma topologia de rede complexa, e ainda não está claro se a abordagem de "Contêineres como serviço" será capaz de proporcionar valor semelhante com mais flexibilidade. Vemos muitas empresas implementando PaaS de prateleira ou gradualmente construindo suas próprias soluções, com níveis de sucesso variáveis. Suspeitamos que qualquer PaaS construída hoje não será um produto final, mas parte de um caminho evolutivo. A migração corporativa para a Nuvem e PaaS, embora traga muitos benefícios, traz dificuldades e desafios, principalmente no que se refere a design de pipeline e ferramentas em geral. Quem consome essas tecnologias deve buscar pelo momento decisivo em que a maturidade é atingida no seu contexto, e evitar um acoplamento muito alto aos detalhes de implementação da sua PaaS.

DOCKER, DOCKER, DOCKER!

A conteinerização, Docker em particular, provou ser extremamente benéfica como técnica de gerenciamento de aplicações, racionalizando a implantação entre ambientes e simplificando problemas do tipo "funciona aqui mas aí não". Vemos uma quantidade significativa de energia concentrada no uso de Docker — e, particularmente, no ecossistema ao seu redor — além das etapas de desenvolvimento/teste, indo até produção. Contêineres Docker são usados como "unidades de escalabilidade" para muitas PaaS e plataformas "data center OS", dando ainda mais força à ferramenta. À medida que amadurece como ambiente tanto de desenvolvimento quanto de produção, as pessoas passam a dar mais atenção à conteinerização, seus efeitos colaterais e suas implicações.

REATIVA DEMAIS?

A programação reativa — na qual componentes reagem a mudanças nos dados que são transmitidos a eles ao invés de usar formas imperativas — tornou-se extremamente popular, com extensões reativas disponíveis em quase todas as linguagens de programação. Interfaces de usuário, em particular, são geralmente escritas em estilo reativo, e muitos ecossistemas estão se definindo nesse paradigma. Apesar de gostarmos do padrão, o uso exagerado de sistemas baseados em eventos complica a lógica de programação, deixando-a difícil de entender. Esse estilo de programação deve ser usado de maneira sensata. Certamente é popular: incluímos um número significativo de frameworks reativos e ferramentas de suporte nessa edição do radar.

COLABORADORES

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

Rebecca Parsons *(CTO)*
Martin Fowler *(Cientista-Chefe)*
Anne J Simmons
Badri Janakiraman
Brain Leke
Dave Elliman
Erik Doernenburg
Evan Bottcher
Fausto de la Torre
Hao Xu
Ian Cartwright
James Lewis
Jonny LeRoy
Mike Mason
Neal Ford
Rachel Laycock
Sam Newman
Scott Shaw
Srihari Srinivasan
Thiyagu Palanisamy

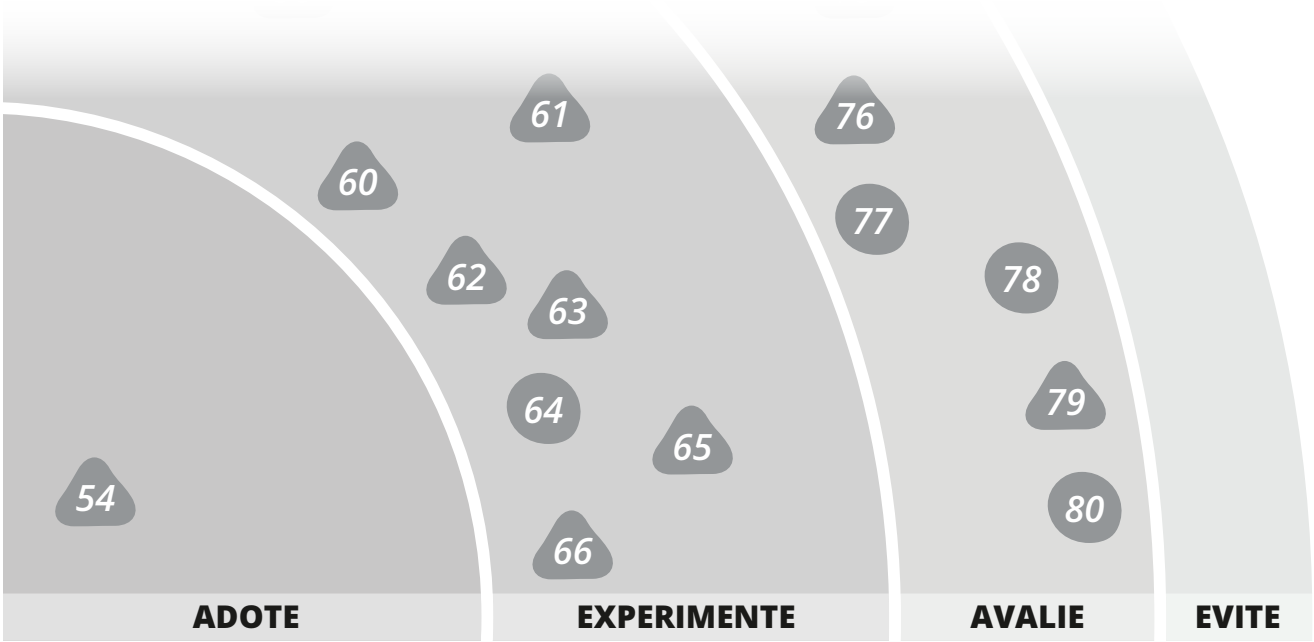
TRADUÇÃO E REVISÃO

Alexandre Barbosa, Alexey Boas, Gregório Melo, Guilherme Froes, Paula Ribas e Ricardo Cavalcanti.

SOBRE O TECHNOLOGY 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 CIOs 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 dentro deles:



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

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.

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

Prossiga com cautela.

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. Os gráficos detalhados de cada quadrante mostram o movimento tomado pelos itens. Temos interesse em muito mais itens do que seria razoável adicionar em um documento deste tamanho, por isso removemos muitos itens do último radar para abrir espaço para novos itens. Quando apagamos um item não significa que deixamos de nos preocupar com ele.

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

O RADAR

TÉCNICAS

ADOTE
1. Desacoplando deployment e release
2. Produtos acima de projetos
3. Modelagem de ameaças

EXPERIMENTE
4. Back-end para front-ends (BFF)
5. Recompensas por bugs
6. Lago de Dados
7. Event Storming
8. Flux
9. Filtro de idempotência
10. iFrames para sandboxing
11. NPM para tudo
12. Ambientes Fênix
13. QA em produção
14. Arquiteturas reativas

AVALIE
15. Políticas de Segurança de Conteúdo novo
16. IDE's hospedados
17. Hospedando dados PII na União Europeia novo
18. Monitoramento de invariantes
19. ASVS do OWASP novo
20. Arquitetura sem servidor novo
21. Unikernels novo
22. Realidade virtual além dos jogos novo

EVITE
23. Instância única de CI para todos os times novo
24. Inveja de Big Data novo
25. Gitflow
26. Inveja da alta performance/inveja da escalabilidade web
27. SAFe™

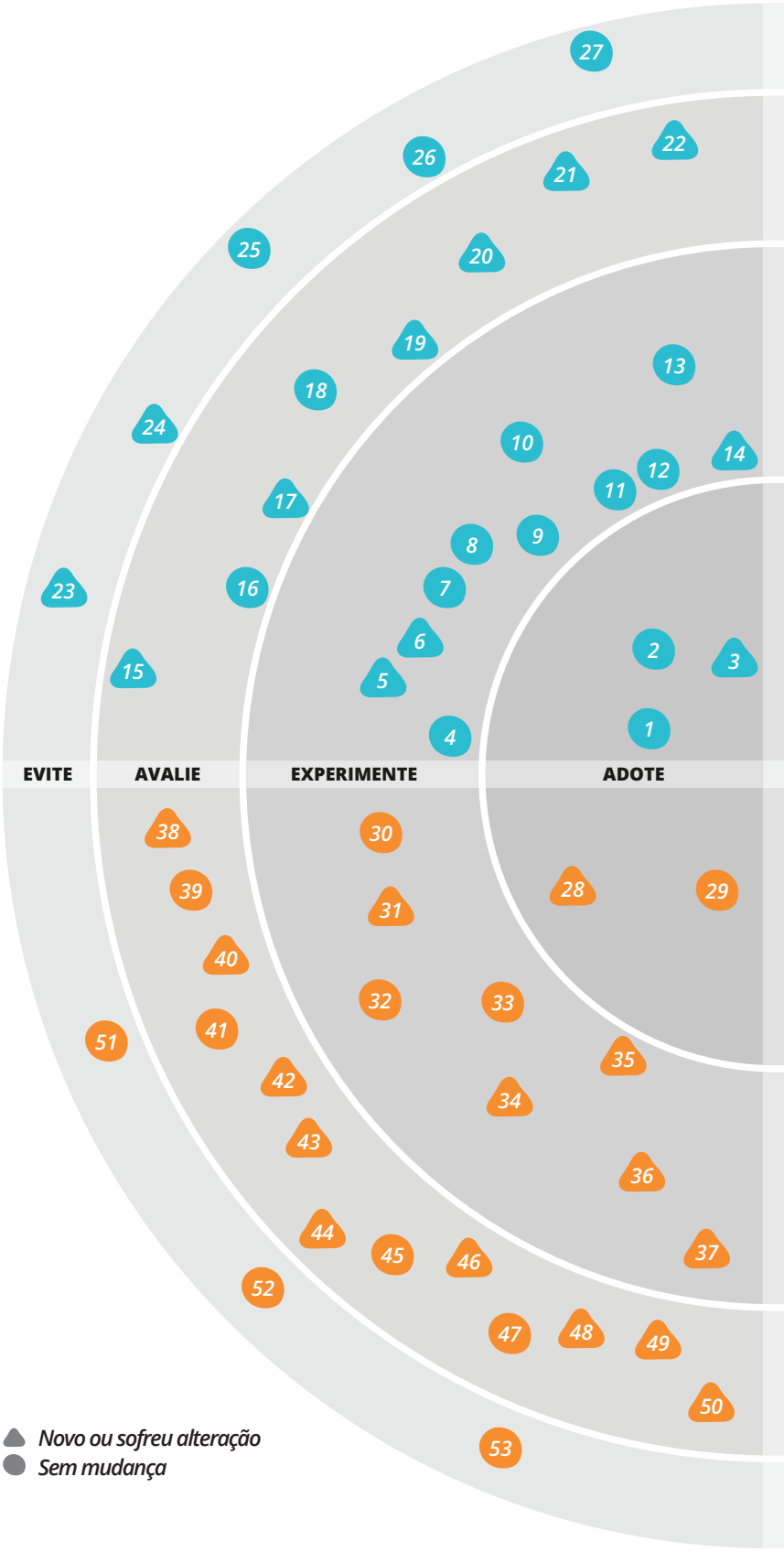
PLATAFORMAS

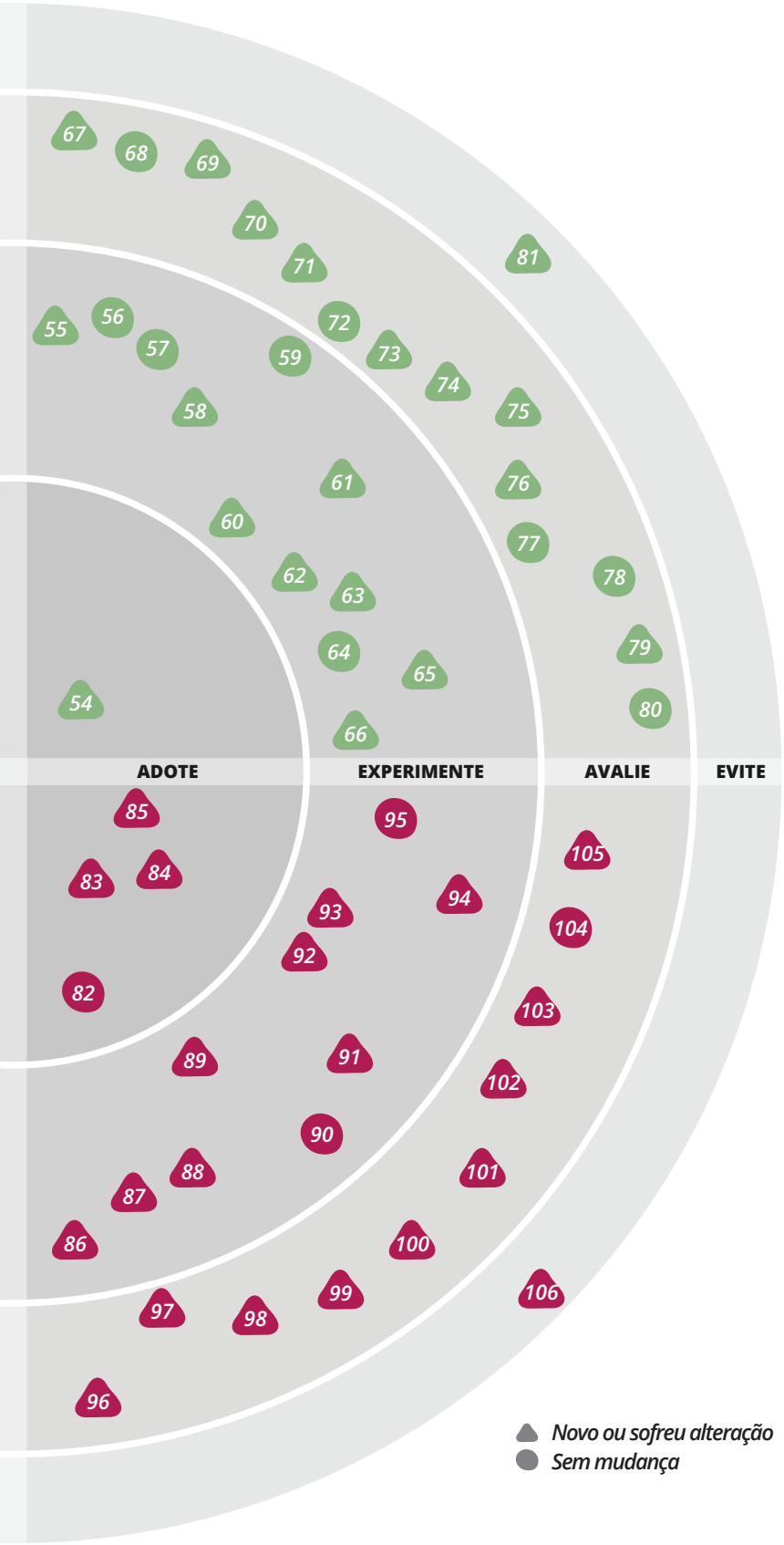
ADOTE
28. Docker
29. TOTP Autenticação de duas etapas

EXPERIMENTE
30. Apache Mesos
31. AWS Lambda
32. H2O
33. HSTS
34. Kubernetes
35. Módulos de segurança Linux
36. Pivotal Cloud Foundry novo
37. Rancher

AVALIE
38. Amazon API Gateway novo
39. AWS ECS
40. Bluetooth Mesh novo
41. Ceph
42. Deflect novo
43. ESP8266 novo
44. MemSQL novo
45. Mesosphere DCOS
46. Nomad novo
47. Presto
48. Realm novo
49. Sandstorm novo
50. TensorFlow novo

EVITE
51. Servidores de aplicação
52. API Gateways excessivamente ambiciosos
53. Nuvem privada superficial





O RADAR

FERRAMENTAS

ADOTE
54. Consul

EXPERIMENTE
55. Apache Kafka
56. Browsersync
57. Carthage
58. Gauge
59. GitUp
60. Let's Encrypt
61. Load Impact novo
62. OWASP Dependency-Check novo
63. Serverspec novo
64. SysDig
65. Webpack novo
66. Zipkin

AVALIE
67. Apache Flink novo
68. Concourse CI
69. Gitrob
70. Grasp novo
71. HashiCorp Vault novo
72. ievms
73. Jepsen novo
74. LambdaCD novo
75. Pinpoint novo
76. Pitest novo
77. Prometheus
78. RAML
79. Repsheet novo
80. Sleepy Puppy

EVITE
81. Jenkins como uma pipeline de implantação novo

LINGUAGENS E FRAMEWORKS

ADOTE
82. ES6
83. React.js
84. Spring Boot
85. Swift

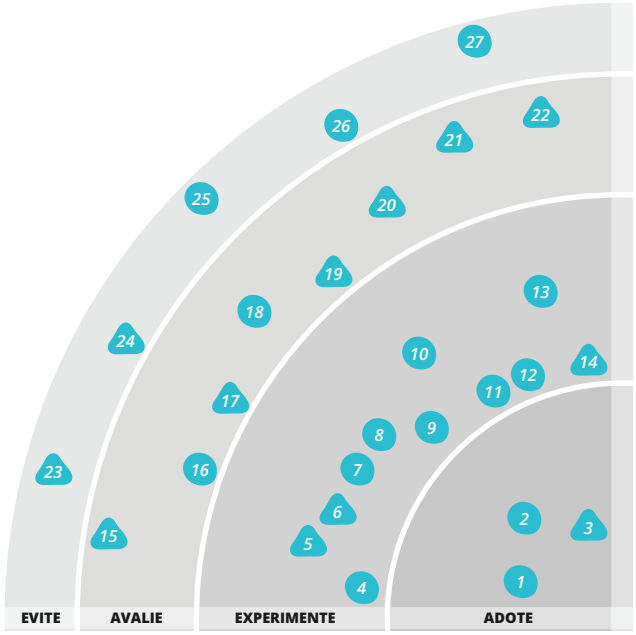
EXPERIMENTE
86. Butterknife novo
87. Dagger novo
88. Dapper novo
89. Ember.js
90. Enlive
91. Fetch novo
92. React Native
93. Redux novo
94. Robolectric novo
95. SignalR

AVALIE
96. Alamofire novo
97. AngularJS
98. Aurelia novo
99. Cylon.js novo
100. Elixir
101. Elm
102. GraphQL novo
103. Immutable.js novo
104. OkHttp
105. Recharts novo

EVITE
106. JSPatch novo

TÉCNICAS

Com o número de violações graves de segurança nos últimos meses, os times de desenvolvimento de software já estão convencidos da importância de escrever software seguro e lidar com dados de usuários de forma responsável. No entanto, os times enfrentam uma curva de aprendizagem acentuada, e o vasto número de potenciais ameaças — que variam de crime organizado e espionagem do governo a adolescentes que atacam sistemas por diversão — pode ser avassalador. **A modelagem de ameaças** fornece uma série de técnicas que te ajudam a identificar e a classificar potenciais ameaças no início do processo de desenvolvimento. É importante entender que modelagem de ameaças é apenas parte de uma estratégia para se antecipar às ameaças. Quando usada em conjunto com técnicas como o estabelecimento de requisitos de segurança multifuncionais para tratar de riscos comuns nas tecnologias usadas em um projeto e o uso de scanners de segurança automatizados, a modelagem de ameaças pode ser um trunfo poderoso.



A popularidade do uso de **recompensas por bugs** continua a crescer entre muitas organizações, incluindo empresas e órgãos governamentais de destaque. Um programa de recompensas por bugs incentiva participantes a identificar potenciais vulnerabilidades em troca de prêmios ou de reconhecimento. Empresas como HackerOne e Bugcrowd oferecem serviços para ajudar organizações a gerenciar esse processo com mais facilidade, e estamos vendo esse tipo de serviço ganhar adoção.

Um **Lago de Dados** é um armazenamento de dados imutáveis, em grande parte brutos e não-processados, que funciona como uma fonte para análise de dados. Embora a técnica possa ser mal utilizada, temos a usado com sucesso em clientes, o que justifica sua mudança para Experimente. Continuamos recomendando outras abordagens para colaborações operacionais, limitando o uso do lago de dados a produção de relatórios, análise e alimentação de repositórios de dados.

Observamos a adoção contínua e o sucesso das **arquiteturas reativas**, com extensões de linguagem reativas e frameworks reativos ganhando muita popularidade (incluímos diversos blips relacionados nessa edição do radar). Interfaces de usuário, em particular, beneficiam-se muito de um estilo reativo de programação. Nossas ressalvas anteriores ainda se aplicam: arquiteturas baseadas em passagem de mensagem assíncrona introduzem complexidade e tornam o sistema como um todo mais difícil de entender — não sendo mais possível simplesmente ler o código do programa e entender o que o sistema faz. Recomendamos avaliar as necessidades de performance e escalabilidade do seu sistema antes de se comprometer a esse estilo de arquitetura.

Temos considerado as **Políticas de Segurança de Conteúdo** adições úteis a nosso conjunto de ferramentas de segurança quando lidamos com sites que extraem assets de diversos contextos. A política define um conjunto de regras que estabelecem de onde assets

ADOTE
1. Desacoplando deployment e release
2. Produtos acima de projetos
3. Modelagem de ameaças

EXPERIMENTE
4. Back-end para front-ends (BFF)
5. Recompensas por bugs
6. Lago de Dados
7. Event Storming
8. Flux
9. Filtro de idempotência
10. iFrames para sandboxing
11. NPM para tudo
12. Ambientes Fênix
13. QA em produção
14. Arquiteturas reativas

AVALIE
15. Políticas de Segurança de Conteúdo
16. IDE's hospedados
17. Hospedando dados PII na União Europeia
18. Monitoramento de invariantes
19. ASVS do OWASP
20. Arquitetura sem servidor
21. Unikernels
22. Realidade virtual além dos jogos

EVITE
23. Instância única de CI para todos os times
24. Inveja de Big Data
25. Gitflow
26. Inveja da alta performance/inveja da escalabilidade web
27. SAFe™

TÉCNICAS *continuação*

podem ser extraídos (e permitem ou não tags de script embutidas). O navegador então se recusa a carregar ou executar JavaScript, CSS ou imagens que violam essas regras. Quando usadas em conjunto com boas práticas como codificação de saída, elas minimizam ataques XSS satisfatoriamente. Curiosamente, foi por meio do endpoint opcional para publicar relatórios JSON sobre violações que o Twitter descobriu que ISPs estavam injetando HTML ou JavaScrpit em suas páginas.

Em vários países do mundo, vemos agências governamentais buscando amplo acesso a informações de identificação pessoal (*personally identifiable information* ou PII) privadas. Na União Europeia, a corte suprema invalidou o framework Safe Harbor, e o Privacy Shield, seu sucessor, deve ser contestado também. Ao mesmo tempo, o uso da computação em nuvem está crescendo, e todos os principais provedores — Amazon, Google e Microsoft — oferecem múltiplos centros de processamento de dados e regiões na União Europeia. Portanto, recomendamos que as empresas, principalmente as que têm bases de usuários globais, avaliem a viabilidade de um refúgio seguro para os dados de seus usuários, protegido pelas leis de privacidade mais progressistas, **hospedando dados PII na União Europeia**.

À medida que os times de desenvolvimento incorporam a segurança mais cedo no ciclo de desenvolvimento, determinar requisitos para limitar riscos de segurança pode parecer uma tarefa complicada. Poucas pessoas têm o extenso conhecimento técnico necessário para identificar todos os riscos que uma aplicação pode enfrentar, e os times podem ter dificuldades para decidir por onde começar. Apoiar-se em frameworks como o **ASVS** — *Application Security Verification Standard* (padrão de verificação de segurança de aplicações) do OWASP pode facilitar esse processo. Embora seja um pouco extenso, ele contém uma lista completa de requisitos categorizados por funções como autenticação, controle de acesso e tratamento e registro de erros, o que pode ser revisto conforme a necessidade. Também é útil como um recurso para analistas de testes quando chega o momento de verificar software.

A **arquitetura sem servidor** substitui máquinas virtuais de longa duração por poder computacional efêmero, que passa a existir sob solicitação e desaparece imediatamente apos o uso. Firebase e AWS Lambda são alguns exemplos. O uso dessa arquitetura pode minimizar algumas preocupações com segurança, como atualizações de segurança e controle de acesso SSH, e pode tornar muito mais eficiente o uso de recursos computacionais. Esses sistemas têm custos de operação muito baixos e podem ter funcionalidades de escalabilidade integradas (isso vale principalmente para AWS Lambda). Um exemplo de arquitetura poderia ser um aplicativo JavaScript com assets estáticos servido por uma CDN ou um S3 acoplado com chamadas AJAX servidas pelo API Gateway e Lambda. Embora arquiteturas sem servidor tenham benefícios significativos, há duas desvantagens também: implantar, gerenciar e compartilhar código entre serviços é mais complexo, e testes locais ou offline são mais difíceis, se não impossíveis.

Com a contínua ascenção do modelo de contêiner liderada pela adoção a Docker, acreditamos que seja válido chamar atenção para o rápido e contínuo desenvolvimento na área de Unikernel. **Unikernels** são sistemas operacionais biblioteca de propósito único, que podem ser compilados a partir de linguagens de alto nível para serem executados diretamente nos hipervisores usados por plataformas commodity de nuvem. Eles prometem diversas vantagens em relação aos contêineres, além de seu tempo extremamente rápido de inicialização e sua reduzida exposição a ataques. Muitos ainda estão na fase de pesquisa/projeto — Drawbridge da Microsoft Research, MirageOS e HaLVM, entre outros — mas achamos que as ideias são muito interessantes e combinam bem com a técnica de arquitetura sem servidor.

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 estamos chegando a um momento decisivo agora. Placas de vídeo modernas fornecem poder computacional

TÉCNICAS *continuação*

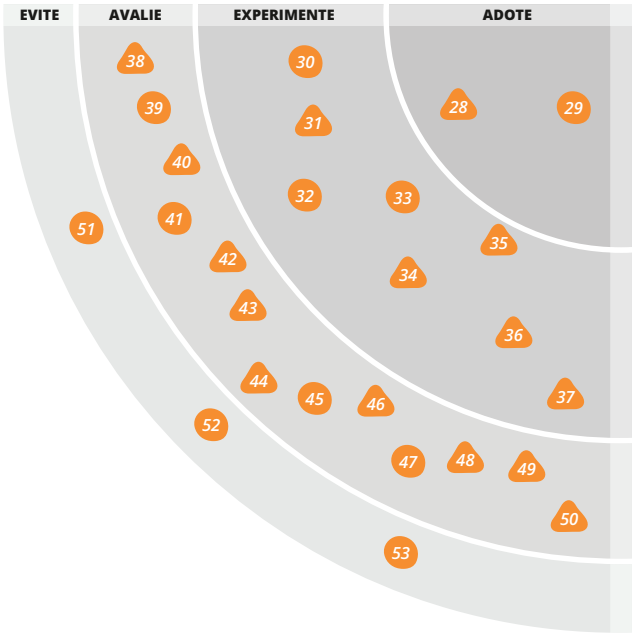
suficiente para renderizar cenas detalhadas e realistas em altas resoluções e, ao mesmo tempo, pelo menos dois headsets de realidade virtual orientados ao consumidor (o HTC Vive e o Oculus Rift do Facebook) estão chegando ao mercado. Esses headsets têm preços acessíveis, displays de alta resolução e eliminam o atraso perceptível na captura de movimentos, que vinha provocando problemas como dores de cabeça e náuseas. 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**, principalmente à medida que as abordagens low-fi, como Google Cardboard, estão contribuindo para um maior conhecimento em torno do tema.

Pode haver a impressão de que é mais fácil gerenciar uma **instância única de Integração Contínua (Continuous Integration ou CI) para todos os times**, porque ela dá a eles um ponto único de configuração e monitoramento. Mas uma instância sobrecarregada compartilhada por todos os times de uma organização pode causar grande prejuízo. Encontramos problemas como build timeouts, conflitos de configuração e filas de build extensas aparecendo com maior frequência. Um único ponto de falha pode interromper o trabalho de vários times. Considere com cuidado as implicações desse tipo de armadilha e de ter um único ponto de configuração. Em organizações com múltiplos times, recomendamos instâncias de CI distribuídas por times, com decisões corporativas baseadas não em uma instalação única de CI, mas na definição de orientações sobre seleção e configuração de instâncias.

Embora tenhamos compreendido bem o valor de Big Data para melhor entendermos como as pessoas interagem conosco, notamos uma tendência alarmante de **inveja de Big Data**: organizações usando ferramentas complexas para lidar com volumes "não tão grandes" de dados. Algoritmos MapReduce distribuídos são uma técnica eficiente para grandes conjuntos de dados, mas muitos conjuntos que observamos poderiam facilmente se encaixar em um banco de dados relacional ou gráfico com um nó único. Mesmo que você tenha uma quantidade maior de dados, em geral o melhor a se fazer é primeiro escolher os dados que você precisa, que frequentemente podem ser processados nesse nó único. Por isso, insistimos que antes de iniciar seus clusters você faça uma avaliação realista daquilo que você precisa processar e, se for possível — talvez na memória RAM —, use a opção simples.

PLATAFORMAS

Continuamos animados com **Docker**, dada a sua evolução de uma ferramenta para uma plataforma complexa de tecnologias. Times de desenvolvimento adoram Docker, já que seu formato de imagem torna mais fácil alcançar a paridade entre desenvolvimento e produção, contribuindo para implantações confiáveis. Funciona bem em aplicações estilo microsserviços como mecanismo de empacotamento para serviços independentes. Do ponto de vista operacional, o suporte de Docker para ferramentas de monitoramento (Sensu, Prometheus, cAdvisor, etc.), ferramentas de orquestração (Kubernetes, Marathon, etc.) e ferramentas de automação de implantação reflete a crescente maturidade da plataforma e sua prontidão para uso em produção. Um alerta, no entanto: existe uma visão predominante de contêineres Docker e Linux em geral como "virtualização leve", mas não recomendamos usar Docker como um mecanismo de isolamento de processos seguro, embora estejamos prestando atenção à introdução de namespaces de usuário e perfis seccomp na versão 1.10 nesse sentido.



Nossos times continuam gostando de usar **AWS Lambda** e estão começando a usar o serviço para experimenção com arquiteturas sem servidor, combinando Lambda com o API Gateway para produzir sistemas altamente escaláveis com infraestrutura invisível. Encontramos problemas significativos usando Java para funções Lambda, com latências irregulares durando vários segundos quando o contêiner Lambda é iniciado. Recomendamos continuar com JavaScript ou Python por ora.

Kubernetes é a resposta do Google para o problema de implantar contêineres em um cluster de máquinas, o que tem se tornado um cenário comum. Não é a solução usada pelo Google internamente, mas um projeto de código aberto que se originou no Google e tem recebido um bom número de contribuições externas. Desde que mencionamos Kubernetes na última edição do radar, nossas impressões iniciais positivas se confirmaram, e temos visto uso bem sucedido de Kubernetes em produção em nossos clientes.

Em versões anteriores do radar, destacamos o valor dos **módulos de segurança do Linux**, descrevendo como eles habilitam as pessoas a pensar sobre enrijecimento de servidores como parte de seus fluxos de desenvolvimento. Mais recentemente, com contêineres LXC e Docker trazendo suporte a perfis AppArmor, padrão em determinadas distribuições Linux, muitos times foram obrigados a entender como essas ferramentas funcionam. No caso de times que usam imagens de contêineres para executar qualquer processo que não tenha sido criado por eles mesmos, essas ferramentas os ajudam a avaliar questões como,

ADOTE
28. Docker
29. TOTP Autenticação de duas etapas

EXPERIMENTE
30. Apache Mesos
31. AWS Lambda
32. H2O
33. HSTS
34. Kubernetes
35. Módulos de segurança Linux
36. Pivotal Cloud Foundry
37. Rancher

AVALIE
38. Amazon API Gateway
39. AWS ECS
40. Bluetooth Mesh
41. Ceph
42. Deflect
43. ESP8266
44. MemSQL
45. Mesosphere DCOS
46. Nomad
47. Presto
48. Realm
49. Sandstorm
50. TensorFlow

EVITE
51. Servidores de aplicação
52. API Gateways excessivamente ambiciosos
53. Nuvem privada superficial

PLATAFORMAS *continuação*

por exemplo, quem tem acesso a quais recursos no servidor compartilhado e as capacidades que esses serviços contidos possuem, adotando uma postura mais conservadora ao gerenciar níveis de acesso

A área de PaaS passou por muitas mudanças desde a última vez em que mencionamos Cloud Foundry, em 2012. Embora existam várias distribuições do núcleo de código aberto, têm nos impressionado a oferta e o ecossistema montados como **Pivotal Cloud Foundry**. Ainda que esperemos convergência contínua entre a abordagem não-estruturada (Docker, Mesos, Kubernetes, etc.) e o estilo mais estruturado e assertivo de buildpack oferecido por Cloud Foundry e outras plataformas, vemos benefícios reais para organizações dispostas a aceitar as limitações e o ritmo de evolução para adotar uma PaaS. É particularmente interessante a velocidade de desenvolvimento que resulta da simplificação e da padronização da interação entre times de desenvolvimento e operações de plataforma.

A emergente área de Contêineres como Serviço (Containers as a Service ou CaaS) tem passado por muitas mudanças e oferece uma opção útil entre IaaS (Infraestrutura como Serviço) básica e PaaS (Plataforma como Serviço) mais assertiva. Embora **Rancher** chame menos atenção que outros serviços similares, tem nos agradado a simplicidade proporcionada pela ferramenta para executar contêineres Docker> em produção. Pode ser executada de forma independente como uma solução completa ou em conjunto com ferramentas como Kubernetes .

Amazon API Gateway é a oferta da Amazon que permite a pessoas desenvolvedoras disponibilizar serviços de API para clientes da Internet, oferecendo as usuais funcionalidades de API Gateway como gestão de tráfego, monitoramento, autenticação e autorização. Nossos times têm usado esse serviço como frente para outras capacidades da AWS, por exemplo AWS Lambda, como parte de arquiteturas sem servidor. Continuamos monitorando os desafios apresentados por API Gateways excessivamente ambiciosos, mas nesse momento a oferta da Amazon parece ser leve o suficiente para evitar esses problemas.

Embora muitas implantações de dispositivos inteligentes dependam de conectividade Wi-Fi, temos visto sucesso em redes **Bluetooth Mesh** que não precisam de hub ou gateway. Com melhor uso de energia que Wi-Fi e adoção de smartphones superior a ZigBee, Bluetooth LE implantada como uma rede auto-reparadora proporciona novas abordagens interessantes para conectar redes de dispositivos de área locais. Ainda estamos esperando pelo surgimento de uma abordagem formal de Bluetooth SIG, mas já tivemos implantações bem sucedidas. Nós particularmente gostamos da ausência de infraestrutura necessária para erguer uma rede descentralizada, mas ainda mantendo a opção de "melhorar progressivamente" o sistema com a adição de um gateway e de serviços de nuvem.

Deflect é um serviço de código aberto para proteger ONGs, ativistas e empresas de mídia independente de ataques DDoS. Similar a uma CDN comercial, ele usa caching de proxy reverso distribuído, oculta seus endereços IP do servidor e bloqueia o acesso público a URLs de admin. Um esforço particular é dedicado a combater botnets tipicamente usadas para censura extrajudicial de vozes independentes.

Nossos crescentes escalões de hackers de hardware têm se mostrado animados com o microcontrolador Wi-Fi **ESP8266**. Não foi uma inovação tecnológica específica, mas a combinação de preço baixo e SFF que provocou uma mudança de pensamento nas pessoas em relação ao que é possível alcançar agora com dispositivos de hardware personalizados. Suas principais características são: capacidades Wi-Fi (pode funcionar como estação, ponto de acesso ou uma combinação dos dois), baixa energia, hardware aberto, programabilidade Arduino SDK, programabilidade Lua, imenso suporte da comunidade e baixo custo comparado a outros módulos de IoT.

Como prevê a Lei de Moore, continuamos a aumentar a capacidade de sistemas computacionais e a reduzir seu custo e, assim, novas técnicas de processamento que alguns anos atrás pareciam fora de alcance tornam-se possíveis. Uma dessas técnicas é o banco de dados in-memory: ao invés de usar discos lentos ou SSDs relativamente lentos para armazenar dados, podemos mantê-los na memória para obter alta performance. Um desses bancos de dados in-memory, o **MemSQL** está ganhando atenção por ser horizontalmente escalável através de um cluster e por oferecer uma linguagem de consulta familiar baseada em SQL. MemSQL também se conecta a Spark para análise de dados em tempo real, diferentemente de dados obsoletos em um depósito de dados.

PLATAFORMAS *continuação*

A HashiCorp continua a produzir software interessante. O último a chamar nossa atenção é o **Nomad**, que está competindo no cada vez mais disputado campo de agendadores. Entre suas maiores vantagens estão o fato de não se limitar apenas a cargas de trabalho conteinerizadas e operar em centros multidados / implantações multi-regiões.

Realm é um banco de dados projetado para uso em dispositivos móveis, com seu próprio mecanismo de persistência para alcançar alta performance. Realm é vendido como um substituto para SQLite e Core Data, e nossos times têm gostado de usá-lo. Vale notar que as migrações não são tão fáceis quanto a documentação do Realm faz parecerem. Ainda assim, Realm nos animou, e sugerimos que você dê uma olhada.

Para quem quer o benefício das ferramentas colaborativas baseadas em nuvem mas não quer inadvertidamente "se tornar o produto" de um grande provedor de nuvem, **Sandstorm** oferece uma alternativa de código aberto interessante, com potencial para hospedagem local. Particularmente interessante é a abordagem de isolamento, pela qual a conteinerização é aplicada por documento e não por aplicação, e syscall whitelisting é adicionado para dar maior segurança ao sandbox.

TensorFlow do Google é uma plataforma de código aberto para aprendizagem de máquina que pode ser usada para qualquer coisa, de pesquisa a produção, e pode ser executada em hardware desde uma CPU móvel até um grande cluster de GPUs. É uma plataforma importante porque torna a implementação de algoritmos de aprendizagem profunda muito mais acessíveis e convenientes. Apesar do entusiasmo, TensorFlow não é algo realmente novo em termos de algoritmo: todas essas técnicas estão disponíveis no domínio público por meio de acervos acadêmicos há algum tempo. Também é importante compreender que a maioria das empresas ainda não está fazendo análises preditivas nem mesmo básicas, e que saltar para aprendizagem profunda provavelmente não vai ajudar a dar sentido à maioria dos conjuntos de dados. Entretanto, para quem de fato tem o problema e o conjunto de dados certos, TensorFlow é um conjunto de ferramentas útil.

FERRAMENTAS

Movemos **Consul**, a ferramenta de descoberta de serviços que suporta mecanismos de descoberta baseados tanto em DNS quanto em HTTP, para Adote. Ela vai além de outras ferramentas de descoberta ao fornecer verificações personalizadas da saúde dos serviços registrados, garantindo que instâncias não-saudáveis sejam corretamente marcadas. Mais ferramentas surgiram para trabalhar junto com Consul, tornando-o ainda mais poderoso. Consul Template permite que arquivos de configuração sejam preenchidos com informações do Consul, tornando coisas como balanceamento de carga do lado do cliente usando mod_proxy muito mais fáceis. No universo do Docker, o registrador pode automaticamente registrar contêineres Docker ao Consul à medida que aparecem com pouquíssimo esforço, tornando muito mais fácil gerenciar configurações baseadas em contêiner. Você ainda deve pensar muito e com cuidado se você precisa de uma ferramenta como essa ou se algo mais simples vai resolver, mas caso você decida que precisa de descoberta de serviços, você não vai errar escolhendo Consul.

Muitas organizações estão olhando mais de perto agora para novas arquiteturas de dados que capturam informações na forma de sequências imutáveis de eventos em escala. O **Apache Kafka** continua a ganhar força como um framework de mensagens de código aberto que fornece uma solução para publicação de feeds de eventos ordenados para grandes quantidades de consumidores leves e independentes. Configurar o Kafka não é simples, mas nossos times têm relatado experiências positivas com o framework.

Gauge é uma ferramenta de automação de testes multiplataforma. As especificações são escritas em Markdown de formato livre para que casos de teste possam ser escritos na linguagem de negócios, ao invés do formato mais comum, porém restritivo, "dado-quando-então". Suporte a linguagens e a IDEs são implementados como plugins, permitindo que analistas de teste usem as mesmas IDEs que o restante do time, com capacidades poderosas como autocompletar e refatoração. Essa ferramenta, desenvolvida com código aberto pela ThoughtWorks, também traz nativamente a execução paralela para todas as plataformas suportadas.

Let's Encrypt apareceu pela primeira vez na última edição do radar, e desde dezembro de 2015 esse projeto modificou o status de sua versão beta de privado para público, o que significa que usuários não precisam mais de um convite para testá-lo. Let's Encrypt permite acesso a um mecanismo mais simples para obter e gerenciar certificados para um grupo maior de usuários que buscam formas para proteger seus websites. Também promove um grande avanço em termos de segurança e privacidade. Essa tendência já começou a ser adotada dentro da ThoughtWorks, e muitos dos nossos projetos agora têm certificados verificados pelo Let's Encrypt.

ADOTE
54. Consul

EXPERIMENTE
55. Apache Kafka
56. Browsersync
57. Carthage
58. Gauge
59. GitUp
60. Let's Encrypt
61. Load Impact
62. OWASP Dependency-Check
63. Serverspec
64. SysDig
65. Webpack
66. Zipkin

AVALIE
67. Apache Flink
68. Concourse CI
69. Gitrob
70. Grasp
71. HashiCorp Vault
72. ievms
73. Jepsen
74. LambdaCD
75. Pinpoint
76. Pitest
77. Prometheus
78. RAML
79. Repsheet
80. Sleepy Puppy

EVITE
81. Jenkins como uma pipeline de implantação

FERRAMENTAS *continuação*

Load Impact é uma ferramenta SaaS de testes de carga que pode gerar cargas altamente realistas de até 1,2 milhão de usuários simultâneos. Gravação e reprodução de interações web usando um plugin para Chrome simulam conexões de rede para usuários móveis ou desktop e geram carga de até 10 locais diferentes ao redor do mundo. Embora não seja a única ferramenta de testes de carga sob demanda que usamos — também gostamos de BlazeMeter —, nossos times ficaram bastante entusiasmados com Load Impact.

Em um universo repleto de bibliotecas e ferramentas que simplificam a vida de muitas pessoas desenvolvedoras de software, deficiências na segurança se tornaram visíveis e aumentaram a superfície de vulnerabilidade nas aplicações que fazem uso delas. **OWASP Dependency-Check** identifica automaticamente potenciais problemas de segurança no código, verificando se existem quaisquer vulnerabilidades reveladas publicamente, e usando métodos para atualizar constantemente o banco de dados de vulnerabilidades públicas. Dependency-Check possui algumas interfaces e plugins para automatizar essa verificação em Java e .NET (os quais usamos com sucesso), bem como em Ruby, Node.js e Python.

Em edições passadas, incluímos Testes de Provisionamento automatizados como uma técnica recomendada, e nesta edição destacamos **Serverspec** como uma ferramenta popular para implementar esses testes. Embora essa ferramenta não seja nova, temos observado seu uso se tornar mais comum à medida que mais times multifuncionais de entrega assumem a responsabilidade pelo provisionamento de infraestrutura. Serverspec é baseado na biblioteca Ruby RSpec e possui um extenso conjunto de utilitários para garantir que a configuração do servidor esteja correta.

Webpack se solidificou como nosso empacotador de módulos JavaScript de confiança. Com sua sempre crescente lista de carregadores, ele fornece uma única árvore de dependências para todos os seus assets estáticos, permitindo manipulação flexível de JavaScript, CSS, etc., e reduzindo o que precisa ser enviado ao navegador e a frequência. Particularmente relevante é a integração harmoniosa entre módulos AMD, CommonJS e ES6, e como ela habilitou times a trabalhar no ES6 e transpilar com perfeição (usando Babel) para versões anteriores para compatibilidade de navegador. Muitos dos nossos times também apreciam o Browserify, que cumpre um papel semelhante, mas tem como objetivo maior disponibilizar módulos Node.js para uso do lado do cliente.

O desenvolvimento do **Zipkin** continua em ritmo acelerado, e desde a metade de 2015 a ferramenta foi movida para a organização openzipkin/zipkin no GitHub. Existem agora bibliotecas para Python, Go, Java, Ruby, Scala e C#, além de imagens Docker disponíveis para quem quer começar rapidamente. Ainda gostamos dessa ferramenta. Há uma comunidade ativa e crescente em torno de seu uso, e sua implementação tem se tornado mais simples. Se você precisa de uma forma de medir a latência de ponta-a-ponta de muitas solicitações lógicas, Zipkin continua a ser uma escolha sólida.

Apache Flink é uma plataforma de nova geração para processamento distribuído e escalável de lotes e fluxos de dados. Em seu núcleo está um mecanismo de transmissão de fluxo de dados. Também suporta operações tabulares (como SQL), de processamento de grafos e aprendizagem de máquina. Apache Flink se destaca por funcionalidades ricas para processamento de fluxos de dados: horário de evento, operações ricas de janela de transmissão, tolerância a erros e semântica "exatamente uma vez". Embora ainda não esteja em sua versão 1.0, a ferramenta conquistou interesse significativo da comunidade devido a inovações no processamento de fluxos de dados, manipulação de memória, gerenciamento de estado e simplicidade de configuração.

Invasores continuam usando software automatizado para acessar repositórios públicos do GitHub, encontrar credenciais AWS e iniciar instâncias EC2 para minerar Bitcoins ou para outros propósitos nocivos. Embora a adoção de ferramentas como git-crypt e Blackbox para armazenar com segurança segredos como senhas e tokens de acesso em repositórios de código esteja crescendo, ainda é muito comum que segredos sejam armazenados sem segurança. Não é incomum também que informações confidenciais de projetos sejam colocadas em repositórios pessoais. **Gitrob** pode ajudar a minimizar o prejuízo de expor segredos. A ferramenta verifica os repositórios de uma organização no GitHub, sinalizando todos os arquivos que possam conter informação sigilosa que não deveria estar em um repositório. A atual versão tem algumas limitações: pode ser usada apenas para verificar organizações públicas no GitHub e seus membros, não inspeciona o conteúdo de arquivos, não revisa o histórico de commits e faz uma verificação completa toda vez que é executada. Apesar dessas limitações, pode ser uma ferramenta reativa útil para ajudar a alertar times antes que seja tarde demais. Deve ser considerada uma abordagem complementar a uma ferramenta proativa, como Talisman.

FERRAMENTAS *continuação*

Muito nos impressionou uma pequena ferramenta de refatoração de linha de comando JavaScript chamada **Grasp**. Oferecendo um rico conjunto de seletores e atuando contra a árvore sintática abstrata, a ferramenta vai muito além de sed e grep. Uma adição útil ao conjunto de ferramentas na nossa busca contínua para que JavaScript seja tratada como uma linguagem de primeira classe

Ter uma forma de armazenar segredos com segurança tem se tornado uma questão cada vez mais importante para projetos. A velha ideia de ter apenas um arquivo com segredos ou variáveis de ambiente está se tornando difícil de administrar, principalmente em ambientes com múltiplas aplicações como microsserviços ou ambientes de microcontêiner, nos quais as aplicações precisam acessar uma variedade de segredos. **HashiCorp Vault** é uma ferramenta promissora que tenta resolver o problema fornecendo mecanismos para acessar segredos com segurança, por meio de uma interface unificada. Possui algumas funcionalidades que facilitam a vida, como criptografia e geração automática de segredos para ferramentas conhecidas, entre outras.

Com o crescimento do uso de bancos de dados NoSQL e com o aumento da popularidade de abordagens poliglotas para persistência, os times agora têm muitas opções no que se refere ao armazenamento dos seus dados. Embora isso tenha trazido muitas vantagens, o comportamento do produto com redes instáveis pode frequentemente provocar problemas sutis (e não tão sutis) que com frequência não são bem compreendidos, em alguns casos até mesmo por quem desenvolveu o produto. O conjunto de ferramentas **Jepsen** e o blog que o acompanha se tornaram de fato referências para qualquer pessoa que busque entender como diferentes bancos de dados e tecnologias de fila reagem em circustâncias adversas. Crucialmente, a abordagem de testes, que inclui clientes nas transações, evidencia possíveis modos de falha para muitos times construindo microsserviços.

LambdaCD oferece aos times uma forma de definir pipelines de Entrega Contínua em Clojure. Isso leva os benefícios da Infraestrutura como código à configuração de servidores de CD: gerenciamento do controle de versão, testes de unidade, refatoração e reutilização de código. Na área de "pipeline como código", LambdaCD se destaca por ser leve, independente e completamente programável, permitindo que times trabalhem com suas pipelines da mesma forma que trabalham com seu código.

Times usando Servidor Fênix ou técnicas de Ambiente Fênix têm encontrado poucas opções de suporte de ferramentas de Gerenciamento de Performance de Aplicações (*Application Performance Management ou APM*). Seus modelos de licença, baseados em hardware de longa duração e em quantidades limitadas, e sua dificuldade em lidar com hardware efêmero, frequentemente as tornam mais problemas do que soluções. Entretanto, sistemas distribuídos precisam de monitoramento, e em algum ponto muitos times identificam a necessidade de uma ferramenta de APM. Acreditamos que **Pinpoint**, uma ferramenta de código aberto nessa área, mereça ser investigada como uma alterativa a AppDynamics e Dynatrace. Pinpoint é escrita em Java, com plugins disponíveis para muitos servidores, bancos de dados e frameworks. Apesar de acharmos que você pode fazer muito usando uma combinação de outras ferramentas leves de código aberto — Zipkin, por exemplo —, se você está procurando por APM, vale a pena considerar Pinpoint.

FERRAMENTAS *continuação*

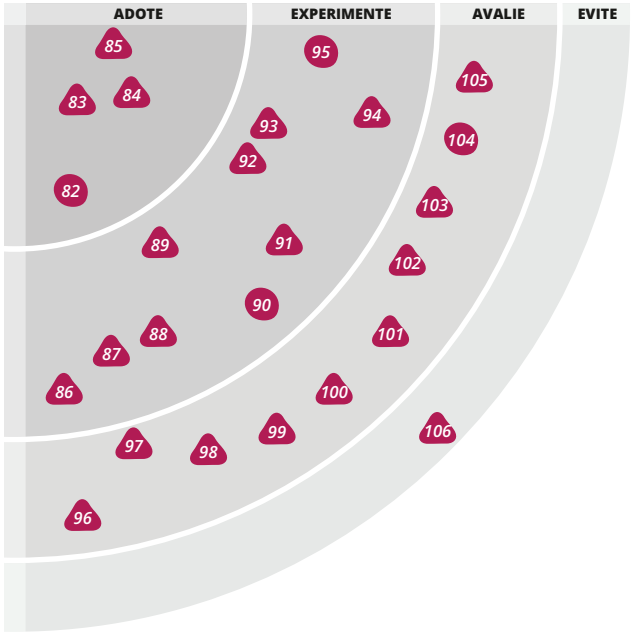
Pitest é uma ferramenta de análise de testes de cobertura para Java que usa uma técnica de teste de mutação. A análise de teste de cobertura tradicional tende a medir o número de linhas executadas por seus testes. É, portanto, capaz apenas de identificar código definitivamente não testado. O teste de mutação, por outro lado, procura testar a qualidade das linhas que são executadas pelo seu código de teste e podem ainda conter erros gerais. Vários problemas podem ser identificados dessa forma, ajudando o time a medir e a desenvolver uma suíte de testes saudável. A maioria das ferramentas tende a ser lenta e difícil de usar, mas Pitest provou ter melhor performance, é fácil de configurar e tem suporte ativo.

Ataques em propriedades web usando bots estão se tornando mais sofisticados. Identificar esses agentes mal-intencionados e seus comportamentos é o objetivo do projeto **Repsheet**. É um plugin para Apache ou NGINX que registra a atividade dos usuários, identifica agentes usando regras definidas previamente e por usuário, permitindo então que ações sejam tomadas, incluindo a possibilidade de bloquear agentes ofensores. Inclui uma utilidade para visualização de agentes atuais, o que permite que membros do time gerenciem ameaças baseadas em bots, aumentando a consciência dos times em relação à segurança. Gostamos disso por ser um exemplo de uma ferramenta simples resolvendo um problema muito real mas frequentemente invisível — ataques realizados por bots.

Sabemos que estamos em um território perigoso aqui, já que desenvolvemos uma ferramenta concorrente, mas achamos que devemos abordar um problema persistente. Ferramentas de Integração Contínua como CruiseControl e Jenkins são valiosas para desenvolvimento de software, mas à medida que seu processo fica mais complexo, ele requer algo além de apenas Integração Contínua: requer uma pipeline de implantação. Frequentemente vemos pessoas tentando usar J**enkins como uma pipeline de implantação** com ajuda de plugins, mas nossa experiência é que isso rapidamente se torna confuso. Jenkins 2.0 introduz "Pipeline como código" mas continua modelando pipelines usando plugins e não altera o produto principal do Jenkins para modelar pipelines diretamente. Em nossa experiência, ferramentas que foram desenvolvidas com uma representação de primeira classe de pipelines de implantação são muitos mais convenientes, e isso foi o que nos levou a substituir CruiseControl por GoCD. Atualmente vemos vários produtos que adotam pipelines de implantação, incluindo ConcourseCI, LambdaCD, Spinnaker, Drone e GoCD.

LINGUAGENS E FRAMEWORKS

Na avalanche de frameworks de front-end JavaScript, **React.js** se destaca devido a seu design em torno de um fluxo de dados reativo. O fato de permitir apenas ligações de dados em sentido único simplifica bastante a lógica de renderização e evita muitos dos problemas que geralmente atingem aplicações escritas com outros frameworks. Estamos observando os benefícios de React.js em um número crescente de projetos, grandes e pequenos. Ao mesmo tempo, continuamos preocupados com o estado e com o futuro de outros frameworks populares como AngularJS. Isso levou React.js a se tornar nossa escolha padrão para frameworks JavaScript.



Bastante esforço foi dedicado ao **Spring Boot** para reduzir complexidade e dependências, o que alivia bastante nossas ressalvas anteriores. Se você se encontra em um ecossistema Spring e está mudando para microsserviços, Spring Boot agora é a escolha óbvia. Para quem não está no mundo do Spring, Dropwizard também vale muito considerar.

Swift é atualmente nossa escolha padrão para desenvolvimento no ecossistema Apple. Com o lançamento da versão Swift 2, a linguagem se aproximou de um nível de maturidade que proporciona a estabilidade e a performance necessárias para a maioria dos projetos. Um número considerável de bibliotecas que suportam desenvolvimento iOS — SwiftyJSON, Quick, etc. — está migrando para Swift, que é o caminho que as demais aplicações devem seguir. Swift teve seu código aberto agora, e temos visto uma comunidade de pessoas desenvolvedoras dedicada a continuamente melhorar o desenvolvimento em iOS.

Butterknife é uma biblioteca de injeção de ligação entre visualização e campo ou método. Ela permite a injeção de objetos arbitrários, visualizações e listeners, garantindo assim código mais limpo com menos código de ligação para desenvolvimento Android. Com Butterknife, múltiplas visualizações podem ser agrupadas em uma lista ou array com ações comuns aplicadas às visualizações simultaneamente, sem grande dependência de configurações XML. Nossos times de projeto têm usado essa biblioteca e se beneficiado de sua simplicidade e facilidade de uso.

Com o aumento da necessidade de aplicações baseadas em Android, **Dagger** oferece um framework de injeção de dependências de tempo de compilação completamente estático. Sua implementação estritamente gerada e não-dependente de soluções baseadas em reflexão lida com muitos dos problemas de performance e desenvolvimento, tornando-a assim adequada para desenvolvimento Android. Com Dagger, há uma rastreabilidade completa com fácil depuração, já que toda a pilha de chamadas para provisionamento e criação é disponibilizada.

Dapper é um ORM minimalista e leve para .NET. Ao invés de tentar escrever consultas SQL para você, Dapper mapeia consultas SQL para objetos dinâmicos. Embora não seja tão recente, Dapper tem consistentemente conquistado a aceitação de

ADOTE
82. ES6
83. React.js
84. Spring Boot
85. Swift

EXPERIMENTE
86. Butterknife
87. Dagger
88. Dapper
89. Ember.js
90. Enlive
91. Fetch
92. React Native
93. Redux
94. Robolectric
95. SignalR

AVALIE
96. Alamofire
97. AngularJS
98. Aurelia
99. Cylon.js
100. Elixir
101. Elm
102. GraphQL
103. Immutable.js
104. OkHttp
105. Recharts

EVITE
106. JSPatch

LINGUAGENS E FRAMEWORKS *continuação*

times da ThoughtWorks trabalhando em .NET. Para quem programa em C#, Dapper elimina uma parte do trabalho massante de mapear consultas relacionais para objetos, ao mesmo tempo possibilitando controle completo sobre o SQL ou stored procedures.

Ember.js desenvolveu um suporte adicional baseado em experiências de projetos e é claramente um concorrente forte no campo de frameworks de aplicações JavaScript. Ember tem recebido elogios pela experiência proporcionada a quem desenvolve, com muito menos surpresas que outros frameworks, como AngularJS. O conjunto de ferramentas de compilação CLI Ember, a abordagem "convenção acima de configuração" e o suporte para ES6 também têm recebido retorno positivo.

Nossos times estão se afastando de JQuery e XHR bruto para ligações JavaScript remotas, e usando em seu lugar a nova API Fetch, e **Fetch** polyfill em particular. A semântica permanece similar, mas com suporte mais limpo a promises e CORS. Estamos vendo isso como a nova abordagem de fato.

Temos visto sucesso contínuo com **React Native** para desenvolvimento multiplataforma acelerado. Apesar de perder alguma adesão à medida que é submetido a desenvolvimento contínuo, as vantagens da integração trivial entre códigos e visualizações nativos e não-nativos, o acelerado ciclo de desenvolvimento (recarregamento instantâneo, depuração em chrome, layout Flexbox) e o crescimento geral do estilo reativo têm nos conquistado. Como acontece com muitos frameworks, é preciso tomar cuidado para manter seu código bem estruturado, mas o uso assíduo de uma ferramenta como Redux ajuda nesse sentido.

Redux é uma ferramenta ótima e madura que tem ajudado muitos dos nossos times a reformular a forma que pensam o gerenciamento de estado em apps lado do cliente. Usando uma abordagem estilo Flux, proporciona uma arquitetura de máquina de estados com acoplamento fraco que facilita o raciocínio. Consideramos uma boa companhia para nossos frameworks JavaScript favoritos, como Ember e React.

No universo de desenvolvimento de aplicações Android, **Roboeletric** é um framework de testes de unidade que tem sido usado por diversos times em nossa comunidade técnica. É a melhor opção entre as disponíveis para escrever testes de unidade reais que se estendem ou interagem diretamente com componentes Android, e suporta testes JUnit. Alertamos, entretanto, que pelo fato de ser uma implementação do Android SDK, pode haver problemas específicos de dispositivos para alguns testes que passam no Roboeletric. Simular manualmente todas as dependências do Android, garantindo apenas testes do sistema-em-teste requer muito código complexo, e esse framework lida com isso de forma eficaz.

Trabalhar com redes e decodificação em aplicações iOS tem sido um grande esforço para muitos times. Foram muitas as bibliotecas e as tentativas de solucionar esse problema constante. Parece que **Alamofire** é a biblioteca mais robusta e acessível para lidar com decodificação JSON. Foi escrita pela mesma criadora da contrapartida Objective-C (AFNetworking), que foi bastante utilizada durante a época de Objective-C.

Embora tenhamos entregado muitos projetos bem sucedidos usando **AngularJS** e estejamos vendo uma aceleração na adoção de configurações corporativas, decidimos mover Angular de volta a Avalie nessa edição do radar. Essa mudança tem como objetivo ser um alerta: React.js e Ember são alternativas sólidas, o caminho para migração da primeira versão de Angular para a segunda tem provocado incerteza e temos visto algumas organizações adotando o framework sem realmente avaliar se uma aplicação de página única atende suas necessidades. Temos debates internos intensos sobre esse tópico, mas certamente temos visto bases de código se tornarem excessivamente complexas devido a uma combinação entre ligações de duas vias e padrões inconsistentes de gerenciamento de estado. Acreditamos que ao invés de exigir que um framework sólido seja abandonado, esses problemas podem ser resolvidos com design cuidadoso e uso de Redux ou Flux desde o princípio.

Aurelia é considerado o framework JavaScript de cliente da próxima geração, e foi escrito usando uma versão moderna de JavaScript: ECMAScript 2016. Aurelia foi criado por Rob Eisenberg, o criador do Durandal. Ele deixou o time principal do Angular 2.0 para se dedicar a esse projeto. O grande diferencial do Aurelia é que ele é altamente modular, contém bibliotecas simples e pequenas e é projetado para ser personalizado com facilidade. Aurelia segue o padrão convenção em vez de configuração, que permite produção e consumo de módulos com mais facilidade, mas sem convenções sólidas que você tenha que aderir. Aurelia possui uma grande comunidade, e pelo site do projeto você pode aprender mais usando os tutoriais.

LINGUAGENS E FRAMEWORKS *continuação*

A interceção entre dispositivos IoT e o ecossistema JavaScript oferece possibilidades interessantes. **Cylon.js** é uma biblioteca JavaScript para construir interfaces para robótica e Internet das Coisas que entusiasmou nossa comunidade técnica. Ela oferece suporte para mais de 50 plataformas de dispositivos, bem como suporte para entrada e saída em geral, com um conjunto compartilhado de drivers fornecidos pelo módulo cylon-gpio. O controle dos dispositivos é possibilitado dessa forma pela interface de um navegador web.

Continuamos a ver bastante entusiasmo entre pessoas usando a linguagem de programação **Elixir**. Elixir, que é desenvolvida com base na máquina virtual Erlang, tem mostrado potencial na criação de sistemas altamente concorrentes e tolerantes a falhas. Elixir possui funcionalidades características como o operador Pipe, que permite construir uma pipeline de funções como no shell do UNIX. O bytecode compartilhado permite ao Elixir interoperar com Erlang e alavancar bibliotecas existentes, ao mesmo tempo suportando ferramentas como Mix, para builds, a shell interativa Iex e o framework de testes de unidade ExUnit.

Tivemos que reconsiderar **Elm** devido à rápida adoção do framework Redux. Elm — a inspiração original para Redux — permite a criação de componentes de visão e a reatividade do React.js, somadas ao estado previsível do Redux em uma linguagem funcional, compilada e fortemente tipada. Elm é escrita em Haskell e possui uma sintaxe similar a Haskell, mas compila para HTML, CSS e JavaScript para o navegador. Pessoas que programam em JavaScript ansiosas em adotar React.js e Redux podem também considerar Elm como uma alternativa de tipagem forte para algumas aplicações.

Quando observamos implementações REST no mundo real, frequentemente vemos REST utilizado incorretamente para, de forma ingênua, recuperar grafos de objeto por meio de uma série de interações entre cliente e servidor. **GraphQL** do Facebook é uma alternativa interessante a REST, que pode ser uma melhor abordagem para esse caso de uso muito comum. Como um protocolo para recuperar grafos de objetos remotamente, GraphQL tem recebido muita atenção ultimamente. Uma de suas funcionalidades mais interessantes é sua natureza orientada ao consumidor: a estrutura de uma resposta é inteiramente definida pelo cliente, não pelo servidor. Implementações de cliente estão disponíveis atualmente em muitas linguagens de programação, mas temos visto uma onda de interesse no Relay do Facebook, um framework JavaScript desenhado para suportar o modelo de componente sem estado do React.js.

A imutabilidade é frequentemente enfatizada no paradigma de programação funcional, e a maioria das linguagens é capaz de criar objetos imutáveis, que não podem ser alterados depois de criados. **Immutable.js** é uma biblioteca para JavaScript que fornece muitas estruturas de dados imutáveis persistentes, altamente eficientes em máquinas virtuais JavaScript modernas. Os objetos de Immutable.js, entretanto, não são objetos JavaScript normais, portanto, referências a objetos JavaScript a partir de objetos imutáveis devem ser evitadas. Nossos times tiveram bons resultados usando essa biblioteca para monitorar mutações e manter estados, e é uma biblioteca que incentivamos pessoas desenvolvedoras a investigar, principalmente quando combinada com o restante do pacote do Facebook.

Temos gostado de como **Recharts** integra gráficos D3 em React.js de maneira limpa e declarativa.

Muitas pessoas que desenvolvem em iOS estão usando **JSPatch** para corrigir dinamicamente seus apps. Quando um app JSPatch é executado, ele carrega um pedaço de JavaScript (potencialmente por meio de uma conexão HTTP insegura) e então se conecta ao código Objective-C da aplicação para alterar comportamento, reparar erros, e assim por diante. Embora seja conveniente, achamos que monkey-patching é uma ideia ruim e deve ser evitada. Ao fazer reparos incrementais de qualquer tamanho, é muito importante que seu processo de testes corresponda à experiência que usuários finais terão, para assim validar corretamente a funcionalidade. Uma abordagem alternativa é usar React Native para o app e AppHub e CodePush para pequenas atualizações e novas funcionalidades.

A ThoughtWorks é uma empresa de software e uma comunidade de pessoas apaixonadas e guiadas por propósitos, especialistas em consultoria, entrega e produtos de software. Pensamos de forma disruptiva para entregar tecnologia que atenda aos maiores desafios de clientes, ao mesmo tempo que buscamos revolucionar a indústria de TI e promover mudanças sociais positivas. Criamos ferramentas pioneiras para times de desenvolvimento que aspiram a ser grandiosos.

Nossos produtos ajudam organizações a melhorar constantemente e a entregar software de qualidade para suas necessidades mais críticas. Fundada há mais de 20 anos, a ThoughtWorks cresceu de um pequeno grupo em Chicago para uma empresa de mais de 3500 pessoas, espalhadas em 35 escritórios e em 12 países: Austrália, Brasil, Canadá, China, Equador, Alemanha, Índia, Singapura, África do Sul, Turquia, Reino Unido e Estados Unidos.

ThoughtWorks®