ThoughtWorks®

TECHNOLOGY RADAR *NOV '15*

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:

DOCKER INCITA EXPLOSÃO DE ECOSSISTEMA DE CONTÊINERES

A conteinerização, exemplificada pelo Docker, tem se tornado extremamente popular em um número crescente de empresas. O interesse varia muito de empresa para empresa e dentro das próprias empresas. Nossas recomendações se estendem do Avalie até o Adote. O ecossistema (ferramentas, plataformas e técnicas) está crescendo e amadurecendo, aumentando ainda mais o interesse. Leitores atentos vão notar tópicos relacionados em todo o radar, abrangendo desde o Docker como uma ferramenta de desenvolvimento para gerenciamento de dependências até grandes plataformas de nuvem como Mesos e AWS ECS, que usam contêineres como sua "unidade de escalabilidade".

MICROSSERVIÇOS E FERRAMENTAS RELACIONADAS GANHAM POPULARIDADE

O interesse em relação a esse estilo de arquitetura não tem diminuído e, consequentemente, vem impulsionando o interesse em ferramentas de suporte e técnicas, por exemplo, práticas de DevOps como a conteinerização, lições aprendidas como os riscos de programar em sua ferramenta de CI/CD, a maturidade das ferramentas de descoberta de serviços, e assim por diante. Esperamos ver ainda mais crescimento e maturidade nessa área em um futuro próximo.

FERRAMENTAS DE JAVASCRIPT CONVERGEM PARA MERAMENTE CAÓTICAS

Já destacamos a agitação no campo de ferramentas de JavaScript antes, mas a comunidade está gradualmente se acalmando e aderindo a algumas práticas comuns. Times estão descobrindo as melhores combinações (incluindo nenhuma) para ferramentas de construção e gerenciamento de pacotes, e temos percebido menos divergências em relação a práticas efetivas.

SEGURANÇA É UM PROBLEMA DE TODOS

Segurança é uma questão que afeta de forma única todos os papéis dentro do ciclo de desenvolvimento de software. Destacamos melhorias na área de segurança no último radar e estamos satisfeitos de ver times cultivando práticas de segurança em seus ciclos de desenvolvimento. Nessa edição também destacamos abordagens inovadoras como recompensas por bugs, modelagem de ameaças, HSTS, TOTP e Let's Encrypt.

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
Claudia Melo
Dave Elliman
Erik Doernenburg
Evan Bottcher
Hao Xu
Ian Cartwright
James Lewis
Jonny LeRoy
Mike Mason
Neal Ford
Rachel Laycock
Sam Newman
Scott Shaw
Srihari Srinivasan
Thiyagu Palanisamy

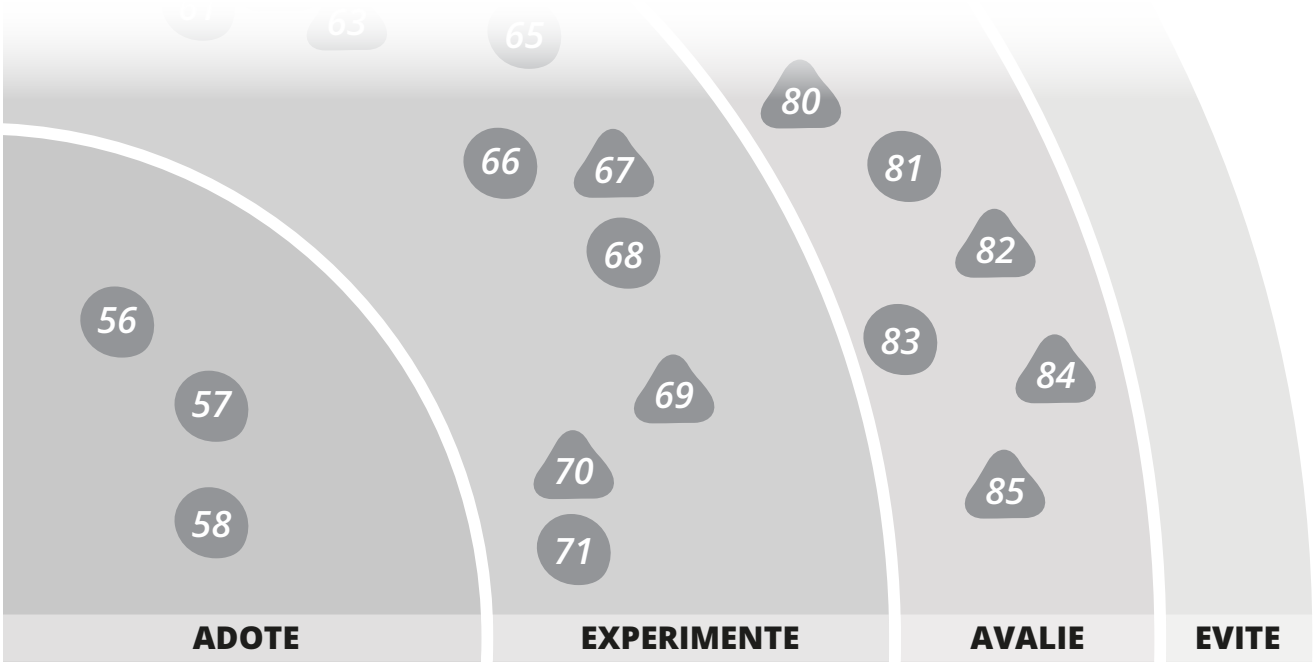
TRADUÇÃO E REVISÃO

Alexandre Barbosa, Alexey Boas, Gregório Melo, Marco Valtas, Paula Ribas, Ricardo Cavalcanti e Ricardo Wendell.

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. Testes de contrato guiados pelo consumidor
2. Desacoplando deployment e release novo
3. Diagrama automatizado de infraestrutura
4. NoPSD
5. Produtos acima de projetos
6. Modelagem de ameaças

EXPERIMENTE
7. BEM novo
8. Back-end para front-ends (BFF) novo
9. Docker para builds novo
10. Event Storming novo
11. Flux
12. Filtro de idempotência novo
13. iFrames para sandboxing novo
14. NPM para tudo novo
15. Aplicações web "offline first"
16. Ambientes Fênix
17. QA em produção novo

AVALIE
18. Dados exclusivamente acumulativos
19. Recompensas por bugs novo
20. Lago de Dados
21. IDE's hospedados novo
22. Monitoramento de invariantes novo
23. Arquiteturas reativas

EVITE
24. Gitflow novo
25. Inveja da alta performance/inveja da escalabilidade web novo
26. Inveja de microsserviços
27. Estratégia de desenvolvimento em camadas
28. Programar em sua ferramenta de integração contínua/Entrega Contínua
29. SAFe™
30. DevOps como um time

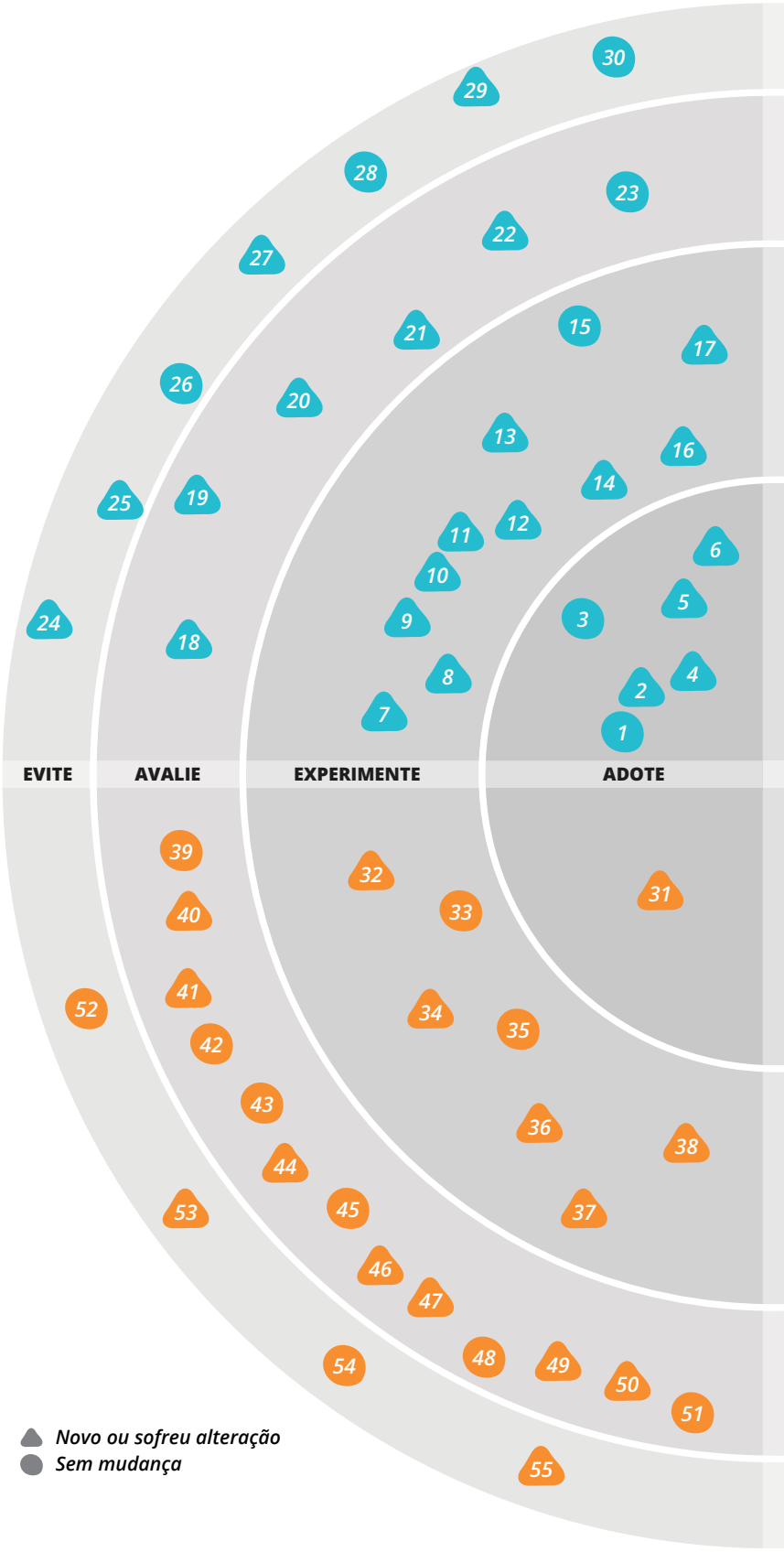
PLATAFORMAS

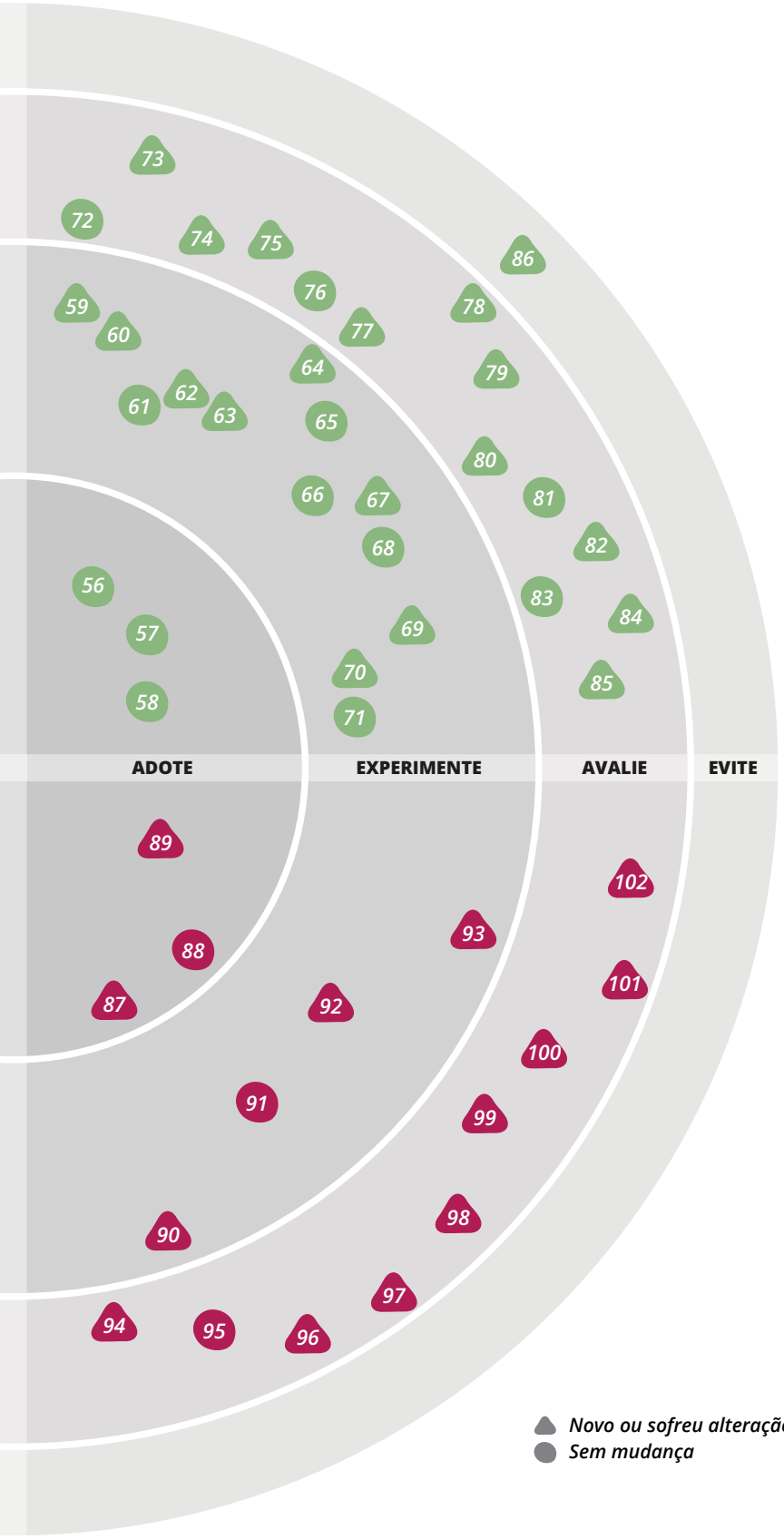
ADOTE
31. TOTP Autenticação de duas etapas

EXPERIMENTE
32. Apache Mesos
33. Apache Spark
34. AWS Lambda novo
35. Cloudera Impala
36. Fastly novo
37. H2O
38. HSTS novo

AVALIE
39. Apache Kylin
40. AWS ECS novo
41. Ceph novo
42. CoreCLR e CoreFX
43. Deis
44. Kubernetes novo
45. Módulos de segurança do Linux
46. Mesosphere DCOS novo
47. Microsoft Nano Server novo
48. Particle Photon/Particle Electron
49. Presto novo
50. Rancher novo
51. Banco de dados de séries temporais

EVITE
52. Servidores de aplicação
53. API Gateways excessivamente ambiciosos novo
54. SPDY
55. Nuvem privada superficial novo





O RADAR

FERRAMENTAS

ADOTE
56. Composer
57. Mountebank
58. Postman

EXPERIMENTE
59. Browsersync novo
60. Carthage novo
61. Consul
62. Docker Toolbox novo
63. Gitrob novo
64. GitUp novo
65. Hamms
66. IndexedDB
67. Polly
68. REST-assured
69. Sensu
70. SysDig novo
71. ZAP

AVALIE
72. Apache Kafka
73. Concourse CI novo
74. Espresso novo
75. Gauge novo
76. Gor
77. ievms novo
78. Let's Encrypt novo
79. Pageify novo
80. Prometheus
81. Quick
82. RAML novo
83. Security Monkey
84. Sleepy Puppy novo
85. Visual Studio Code novo

EVITE
86. Citrix para desenvolvimento

LINGUAGENS & FRAMEWORKS

ADOTE
87. ECMAScript 6 novo
88. Nancy
89. Swift

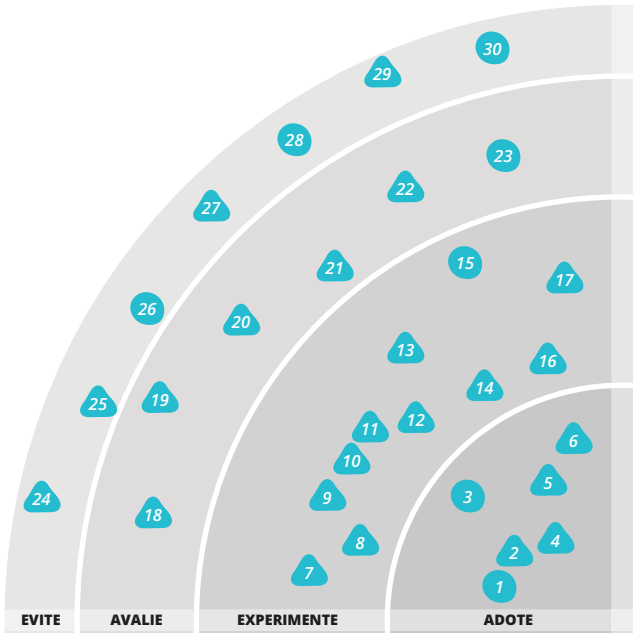
EXPERIMENTE
90. Enlive novo
91. React.js
92. SignalR novo
93. Spring Boot

AVALIE
94. Axon novo
95. Ember.js
96. Frege novo
97. HyperResource novo
98. Material UI novo
99. OkHttp novo
100. React Native novo
101. TLA+ novo
102. Traveling Ruby novo

EVITE

TÉCNICAS

Implementar a Entrega Contínua permanece um desafio para muitas empresas, e continua sendo importante destacar técnicas úteis como o **desacoplamento entre deployment (implantação) e release (liberação)**. Recomendamos usar o termo "deployment" estritamente para se referir ao ato de implantar uma alteração aos componentes ou à infraestrutura de uma aplicação. O termo "release" deve ser usado quando uma alteração de funcionalidade é liberada para os usuários finais, causando impacto no negócio. Usando técnicas como feature toggles e dark deployments, podemos implantar alterações em sistemas de produção com mais frequência, sem liberar funcionalidades. Deployments mais frequentes reduzem o risco associado às alterações, enquanto os stakeholders do negócio mantêm o controle sobre quando funcionalidades devem ser liberadas para usuários finais.



O design 'Just In Time' é um conceito importante e útil para o design visual, que o movimento **NoPSD** tenta capturar. Não é necessário fazer o design da aplicação inteira ou de cada elemento de UI com antecedência. Trabalhe no design de cada coisa à medida que você precisar, usando as ferramentas mais leves possíveis. Temos observado um crescimento no uso de ferramentas mais simples com curvas de aprendizagem mais rápidas, como **Sketch**, bem como um retorno crescente ao "papel e caneta" (principalmente quando combinados a um atual e robusto **guia de estilo digital**). Devido às limitações dos mock-ups planos em designs para telas, criar protótipos de fidelidade variável com ferramentas como **Invision**, **FramerJS** e **Origami** – ou simplesmente HTML/CSS e um pouco de JavaScript – também se tornou cada vez mais comum e valioso para comunicar o objetivo do design.

Temos defendido há bastante tempo a ideia de que pensar o desenvolvimento de software como um projeto – algo orçado e entregue durante um intervalo de tempo limitado – não atende às necessidades do mercado moderno. Esforços significativos para desenvolvimento de software precisam funcionar como um produto contínuo, que apoia e repensa o processo do negócio sustentado por ele. Esses esforços não estarão completos até que o processo do negócio e o seu software deixem de ser úteis. Nossa observação desta abordagem de **produtos acima de projetos**, tanto em relação aos nossos projetos quanto externamente, leva-nos a determinar que essa é a abordagem a ser usada em praticamente todos os casos.

Com o número de violações graves de segurança nos últimos meses, times de desenvolvimento de software já estão convencidos da importância de escrever código seguro e lidar com dados de usuários de forma responsável. No entanto, os times se deparam com uma perigosa curva de aprendizagem, e o grande

ADOTE
1. Testes de contrato guiados pelo consumidor
2. Desacoplando deployment e release
3. Diagrama automatizado de infraestrutura
4. NoPSD
5. Produtos acima de projetos
6. Modelagem de ameaças

EXPERIMENTE
7. BEM
8. Back-end para front-ends (BFF)
9. Docker para builds
10. Event Storming
11. Flux
12. Filtro de idempotência
13. iFrames para sandboxing
14. NPM para tudo
15. Aplicações web "offline first"
16. Ambientes Fênix
17. QA em produção

AVALIE
18. Dados exclusivamente acumulativos
19. Recompensas por bugs
20. Lago de Dados
21. IDE's hospedados
22. Monitoramento de invariantes
23. Arquiteturas reativas

EVITE
24. Gitflow
25. Inveja da alta performance/inveja da escalabilidade web
26. Inveja de microsserviços
27. Estratégia de desenvolvimento em camadas
28. Programar em sua ferramenta de integração contínua/Entrega Contínua
29. SAFe™
30. DevOps como um time

TÉCNICAS *continuação*

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, em geral de uma perpectiva defensiva, que ajudam a entender e a classificar potenciais ameaças. Transformados em "estórias de usuários mal-intencionados", modelos de ameaças podem proporcionar ao time uma abordagem gerenciável e efetiva para tornar seus sistemas mais seguros.

Encontrar problemas em CSS pode ser difícil. Quantas vezes você já teve que vasculhar milhares de estilos sobrescritos para chegar à fonte do seu problema? Isso levou muitos dos nossos times a introduzir diversas diretrizes, como evitar o efeito cascata, fazer estilos opt-in e enfatizar a atenção aos nomes. **BEM** é uma convenção simples de nomes (a sigla representa Bloco, Elemento, Modificador) que ajuda a proporcionar clareza semântica e estrutura ao seu CSS. Usar BEM facilita a comprensão de quais regras de CSS estão influenciando a aparência de um elemento e, mais importante, o objetivo dessas regras. Essa abordagem pode ser vista como uma forma de transportar a lição de OO de favorecer a composição em relação à herança para o mundo do CSS.

Serviços valiosos dão suporte a diversas variações de clientes, como mobile versus web e diferentes formas de interface web. É tentador projetar um back-end único para dar suporte a todos os clientes através de uma API reutilizável. Mas as necessidades dos clientes variam, como também variam as restrições, por exemplo, largura de banda para dispositivos móveis versus o desejo por grandes quantidades de dados em conexões web velozes. Consequentemente, muitas vezes é **melhor definir diferentes serviços de back-end para cada tipo de cliente front-end**. Esses back-ends devem ser desenvolvidos por times alinhados com cada tipo de front-end, para assegurar que cada back-end se adeque às necessidades dos respectivos clientes.

Um dos muitos usos inovadores do Docker que temos visto em nossos projetos consiste em uma técnica para gerenciar dependências de tempo do build. Antigamente, era comum executar agentes de build em um sistema operacional ampliado com dependências necessárias para o build. Mas com o Docker é possível executar a etapa da compilação em um ambiente isolado completo com dependências, sem contaminar o agente de build. Essa técnica de uso do **Docker para build** se provou particularmente útil para compilar binários Golang, e o contêiner golang-builder está disponível exatamente para esse propósito.

Event Storming é uma maneira útil de fazer modelagem de domínio veloz "de fora para dentro": começando com os eventos que ocorrem no domínio em vez de um modelo de dados estático. Funcionando como um workshop facilitado, o Event Storming tem como foco descobrir eventos-chave de domínio, posicionando-os em uma linha do tempo, identificando seus gatilhos e então explorando suas relações. Essa abordagem é particularmente útil para quem está adotando CQRS ou Event Sourcing. Reunir as pessoas certas na sala é importante – uma mistura de perfis técnicos e de negócios que trarão tanto as perguntas quanto as respostas. Assegurar que há espaço suficiente na parede para modelar é a segunda chave para o sucesso. Procure entender o todo, tendo como objetivo a compreensão coletiva do domínio em toda sua complexidade, antes de partir para as soluções.

Flux é uma arquitetura de aplicação apresentada pelo Facebook. Geralmente mencionado em conjunto com **React.js**, Flux baseia-se em um fluxo de dados unidirecional na pipeline de renderização. Flux adota a visão da web moderna de aplicações JavaScript lado do cliente, de modo a evitar os veneráveis clichés MV*. Os times da ThoughtWorks estão começando a ganhar experiência com esse estilo de arquitetura e a descobrir que ele funciona bem com orientação a serviços e soluciona alguns dos problemas inerentes às ligações de dados de mão dupla.

Muitos serviços, principalmente serviços legados, são escritos sob a suposição de que qualquer requisição vai ocorrer uma única vez. Sendo as redes da forma que são, isso pode ser difícil de se garantir. Um **filtro de idempotência** é um componente simples que apenas checa se existem requisições duplicadas e assegura que elas sejam enviadas ao fornecedor de serviços uma única vez. Esse filtro deve realizar apenas essa tarefa e ser usado como um Decorator acima de chamadas a serviços existentes.

TÉCNICAS *continuação*

Páginas web modernas tendem a conter uma variedade de widgets e snippets JavaScript de diversas fontes de terceiros. Isso pode provocar um impacto negativo tanto na segurança quanto na performance. Enquanto ainda estamos esperando por um isolamento mais completo do JavaScript com componentes web, nossos times têm se beneficiado do uso de **iFrames HTML5 para sandboxing** de JavaScript não confiável.

O mundo do JavaScript tem uma diversidade de ferramentas para gerenciamento de dependências e pacotes, todas as quais dependem do Node Package Manager (NPM). Times estão começando a ver essas ferramentas extras como redundantes e recomendando usar, quando possível, somente NPM para gerenciamento de pacotes e dependências. A simplificação do uso de **NPM para tudo** ajuda a reduzir parte da agitação no campo de ferramentas de JavaScript.

O tempo gasto com o provisionamento e atualização de ambientes continua sendo um gargalo significativo em muitos projetos de desenvolvimento de software. Ambientes Fênix podem ajudar a reduzir esse atraso estendendo a ideia dos Servidores Fênix a ambientes inteiros. Julgamos essa uma técnica tão valiosa e eficaz para poupar tempo que você deve considerar testar essa abordagem. Usando automação, podemos criar ambientes inteiros – incluindo configuração de rede, balanceamento de carga e portas de firewall – por exemplo usando **CloudFormation** em AWS. Podemos assim provar que o processo funciona ao destruir ambientes e recriá-los do zero regularmente. **Ambientes Fênix** podem dar suporte ao provisionamento de novos ambientes para testes, desenvolvimento, UAT e recuperação de desastres. Assim como nos Servidores Fênix, esse padrão nem sempre é aplicável e precisamos pensar com cuidado sobre condições como estado e dependências. Tratar o ambiente inteiro como uma implantação azul/verde pode ser uma opção quando a reconfiguração de ambiente é necessária.

Tradicionalmente, QAs têm se concentrado em avaliar a qualidade de um produto de software em um ambiente de pré-produção. Com a ascensão da Entrega Contínua, o papel de QA tem mudado para incluir a análise de qualidade do produto de software em produção. Isso envolve monitorar os sistemas de produção, criando condições de alerta para detectar falhas urgentes, determinar problemas de qualidade em curso e descobrir quais medidas podem ser usadas no ambiente de produção para que isso funcione. Embora exista um risco de que algumas empresas possam ir longe demais e negligenciar QA na pré-produção, nossa experiência demonstra que **QA em produção** é uma ferramenta valiosa para empresas que já atingiram um grau razoável de Entrega Contínua.

Estruturas de dados imutáveis estão se tornando cada vez mais populares, com linguagens funcionais como Clojure e Scala fornecendo imutabilidade por padrão. Imutabilidade permite que o código seja mais facilmente escrito, lido e pensado. Usar um armazenamento de **dados exclusivamente acumulativos** pode conferir alguns desses benefícios à camada de banco de dados, bem como simplificar a auditoria e a consulta de histórico. As opções de implementação variam de soluções específicas de armazenamento de dados acumulativos como Datomic ao simples uso de uma abordagem "adicione-não-atualize" com um banco de dados tradicional. **'Exclusivamente acumulativo'** é uma estratégia de design através da qual dados são removidos por retratação ao invés de atualização; **'Exclusivamente aditivo'** é uma técnica de implementação.

Um número crescente de empresas está começando a distribuir **recompensas por bugs** para incentivar que bugs – frequentemente relacionados a segurança – sejam reportados, e para ajudar a melhorar a qualidade dos softwares de forma geral. Para dar suporte a esses programas, empresas como HackerOne e BugCrowd oferecem soluções para gerenciar este processo de forma mais simples. Nós mesmos temos pouca experiência com esse tipo de oferta, mas gostamos da ideia de incentivar que pessoas se apresentem e apontem algo que pode facilmente ser uma vulnerabilidade prejudicial, de uma forma aberta e transparente. Vale observar que pode haver alguns problemas legais em incentivar usuários a achar vulnerabilidades no seu software, então, por favor, cheque isso primeiro.

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. Enquanto os armazéns de dados mais comuns filtram e processam os dados antes de armazená-los, o lago apenas captura dados brutos, deixando que os usuários

TÉCNICAS *continuação*

desses dados façam a análise específica que eles precisam. Alguns exemplos são HDFS e HBase com Hadoop, Spark ou Storm. Em geral, apenas um grupo pequeno de cientistas de dados trabalha com dados brutos, desenvolvendo fluxos de dados processados que são armazenados em repositórios periféricos para usuários consultarem. Um Lago de Dados deve ser usado apenas para análise e geração de relatórios. Para colaboração entre sistemas preferimos usar serviços designados para esse propósito.

Muitas empresas querem alavancar o desenvolvimento distribuído ou offshore, mas se preocupam com a segurança de seus códigos e outras propriedades intelectuais que podem acabar saindo do seu controle. Como consequência disso, é comum o uso de soluções de desktop remoto de alta latência para o desenvolvimento, que agregam benefícios ao controle de segurança de uma empresa, mas prejudicam a produtividade de quem desenvolve. Uma alternativa é usar um **IDE hospedado** ligado a um navegador via VPN. IDE, código e ambiente de compilação ficam hospedados na nuvem privada da empresa, amenizando as preocupações com segurança e melhorando a experiência de quem desenvolve de maneira significativa. Alguns exemplos de ferramentas nessa área são Orion e Che, da Eclipse Foundation, Cloud9 e Code Envy.

Em monitoramento, a abordagem mais comum é antecipar condições de erro e definir alertas para quando elas acontecerem. Porém, normalmente é difícil enumerar a grande quantidade de modos de falha em um sistema de software. **Monitoramento de invariantes** é uma abordagem complementar para estabelecer margens esperadas, geralmente através da análise de histórico de comportamento, e alertar quando um sistema extrapola esses limites.

Acreditamos veementemente que branches de controle de versão de vida longa prejudicam práticas valiosas de engenharia, como integração contínua, e essa crença sustenta nossa aversão pelo **Gitflow**. Adoramos a flexibilidade do Git por trás, mas abominamos ferramentas que incentivam práticas ruins de engenharia. Branches de vida muito curta prejudicam menos, mas a maioria dos times que vemos usando Gitflow se sente empoderada para abusar de seu fluxo de trabalho carregado de branches, que incentiva a integração tardia (inibindo assim uma integração verdadeiramente contínua).

Observamos muitos times que se deparam com problemas porque escolheram ferramentas, frameworks ou arquiteturas complexos, sob a justificativa de uma "possível necessidade de escalar". Empresas como Twitter e Netflix têm que ser capazes de suportar cargas extremas, por isso elas precisam dessas arquiteturas. Mas elas também contam com times de desenvolvimento extremamente habilidosos, capazes de lidar com essa complexidade. A maioria das situações não requer esse tipo de façanha de engenharia. Times deveriam controlar sua **inveja da escalabilidade web** e priorizar soluções mais simples, mas que ainda assim atendem suas necessidades.

A abordagem da estratégia de desenvolvimento em camadas do Gartner (**Pace-layered Application Strategy**) parece estar gerando um foco desnecessário na ideia de camadas em arquiteturas. Consideramos a reflexão sobre o ritmo de alteração para diferentes **competências do negócio** (que podem ser compostas por diversas camadas de arquitetura) um conceito muito mais útil. O risco de focar em camadas é que muitos tipos de alteração interferem em múltiplas camadas. Por exemplo, a possibilidade de adicionar uma nova classe de ações a um website não depende apenas de ter um CMS fácil de alterar. Também é preciso atualizar banco de dados, pontos de integração, sistemas de armazenamento, etc. O reconhecimento de que algumas partes de uma arquitetura precisam ser mais maleáveis que outras é útil. Entretanto, o foco em camadas está se revelando desnecessário.

Scaled Agile Framework® ou **SAFe™** continua ganhando progressiva repercussão em muitas empresas. Além disso, ferramentas e certificações estão se tornando aspectos significativos na adoção do SAFe™. Continuamos receosos de que as adoções de fato estejam propensas à padronização em excesso e tendendo a práticas de grandes releases, o que dificulta a adoção da metodologia Ágil. Nas situações apropriadas, continuamos a recomendar abordagens lean que incluem experimentação e incorporam práticas de melhoria contínua, como Katas de Melhoria que oferecem às empresas modelos melhores para escalar Ágil.

Scaled Agile Framework® e SAFe™ são marcas da Scaled Agile, Inc.

PLATAFORMAS

Segurança de senha ainda é um tópico discutido acaloradamente, com o governo do Reino Unido advogando em favor de controles técnicos que permitam aos usuários lembrar senhas mais simples e o conselho de Edward Snowden sobre senhas sendo classificado apenas como segurança superficial. Senhas são geralmente um dos elos mais fracos na cadeia de segurança, por isso recomendamos a aplicação da **autenticação de duas etapas**, que pode melhorar a segurança significantemente. Senha temporária de acesso único (**TOTP**) é o algoritmo padrão nesse campo, com implementações simples no lado do servidor e aplicativos gratuitos de autenticação para smartphones do Google e da Microsoft.

Mesos é uma plataforma que abstrai recursos computacionais de camadas subjacentes, facilitando o desenvolvimento de sistemas distribuídos altamente escaláveis. Pode ser usada para fornecer uma camada de agendamento para o Docker ou atuar como uma camada de abstração para coisas como AWS. O Twitter a utilizou com grande sucesso para ajudar a escalar sua infraestrutura. Ferramentas construídas a partir do Mesos já começam a aparecer, como o Chronos, que é um substituto distribuído e tolerante a falhas para o cron. Histórias de sucesso proeminentes também começam a surgir, como o Siri da Apple, rearquitetado para usar Mesos.

A AWS libera uma enorme quantidade de novas funcionalidades mensalmente, por isso às vezes pode ser difícil para qualquer nova oferta de serviço se destacar. Mas o **Lambda** definitivamente tem conseguido atrair atenções. Inicialmente suportando apenas JavaScript, mas agora também adicionando suporte a aplicações baseadas na JVM (e certamente com mais por vir), o Lambda permite iniciar processos de vida muito curta, ou por reação a um evento ou através de uma chamada da API Gateway relacionada. Para serviços sem estado, isso significa que não é preciso se preocupar com quaisquer máquinas de vida longa, potencialmente reduzindo custos e melhorando a segurança. Apesar de outras incursões ao campo do PaaS pela AWS, o Lambda parece ser o mais próximo de acertar.

Fastly, uma das várias CDNs no mercado, tem uma grande e crescente adesão entre os projetos da ThoughtWorks, e é usado por muitos nomes importantes da web em escala, como GitHub e Twitter. Seu conjunto de funcionalidades, velocidade e preço combinados a tornam uma opção muito atrativa quando se está procurando por uma solução de cache de borda. Temos observado também reduções significativas de custo em projetos que migraram de outra CDN para essa plataforma. Se você está procurando por uma CDN, não deixe de pesquisar sobre esta.

ADOTE
31. TOTP Autenticação de duas etapas

32. EXPERIMENTE
33. Apache Mesos
34. Apache Spark
35. AWS Lambda
36. Cloudera Impala
37. Fastly
38. H2O
39. HSTS

40. AVALIE
41. Apache Kylin
42. AWS ECS
43. Ceph
44. CoreCLR e CoreFX
45. Deis
46. Kubernetes
47. Módulos de segurança do Linux
48. Mesosphere DCOS
49. Microsoft Nano Server
50. Particle Photon/Particle Electron
51. Presto
52. Rancher
53. Banco de dados de séries temporais

54. EVITE
55. Servidores de aplicação
56. API Gateways excessivamente ambiciosos
57. SPDY
58. Nuvem privada superficial

PLATAFORMAS *continuação*

Análise preditiva vem sendo usada cada vez mais em produtos, muitas vezes diretamente em funcionalidades para o usuário final. **H2O** é um pacote interessante de código aberto (suportado por uma startup) que faz análises preditivas acessíveis para times de desenvolvimento, oferecendo uma grande variedade de análises, excelente performance e fácil integração em plataformas baseadas na JVM. Ao mesmo tempo, integra-se com as ferramentas favoritas dos cientistas de dados, R e Python, bem como Hadoop e Spark.

Segurança de transporte HTTP estrito (**HSTS**) é atualmente uma política amplamente suportada, que permite que websites se protejam de ataques downgrade. Um ataque downgrade, no contexto do HTTPS, é aquele que pode direcionar usuários do seu site para o HTTP ao invés de HTTPS, permitindo outros ataques, como ataques man-in-the-middle. Usando o cabeçalho do servidor, você informa aos navegadores que eles devem usar apenas HTTPS para acessar seu site, e devem ignorar tentativas de downgrade para contactar o site via HTTP. O suporte dos navegadores é hoje difundido o suficiente para que essa funcionalidade simples de implementar seja considerada por qualquer site usando HTTPS.

Elastic Container Service (ECS) é a entrada da AWS no campo do Docker multihost. Embora exista muita competição nessa área, não há muitas soluções gerenciadas fora de instalações internas. Ainda que o ECS pareça um bom começo, estamos receosos de que seja exageradamente complicado no momento, e careça de uma boa camada de abstração. Caso você queira executar Docker na AWS, no entanto, essa ferramenta certamente deve estar no topo de sua lista. Apenas não espere que ela seja fácil de usar no início.

Ceph é uma plataforma de armazenamento que pode ser usada como armazenamento de objetos, armazenamento de blocos e como um sistema de arquivos, em geral executando em um cluster de servidores commodity. Tendo seu primeiro grande release acontecido em julho de 2012, o Ceph certamente não é um produto novo. Queremos destacá-lo no radar como um importante bloco de construção para nuvens privadas. Ele é particularmente atrativo porque seu componente RADOS Gateway é capaz de expor o armazenamento de objetos através de uma interface RESTful que é compatível com Amazon S3 e as APIs OpenStack Swift.

Kubernetes é a reposta do Google para o problema de implantar contêineres em um cluster de máquinas, o que tem se tornado um cenário cada vez mais 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. Docker e Rocket são suportados como formatos de contêiner e os serviços oferecidos incluem gerenciamento de saúde, replicação e descoberta. Uma alternativa similar nesse campo é o **Rancher**, uma solução de código aberto que também permite implantação de contêineres em um cluster de máquinas e fornece soluções como gerenciamento de ciclo de vida, monitoramento, checagens de saúde e descoberta. Também inclui um sistema operacional completamente conteinerizado baseado no Docker. O foco amplo em conteinerização e um footprint muito pequeno são vantagens chave do Rancher.

Mesosphere DCOS é uma plataforma construída a partir do Mesos. Fornece uma abstração sobre máquinas subjacentes, proporcionando um pool de armazenamento e computação que permite que serviços construídos para DCOS operem em grande escala (já existe suporte para Hadoop, Spark e Cassandra, entre outros). No momento, provavelmente é exagero para cargas de trabalho mais modestas (nesses casos, o bom e velho Mesos ainda pode ser uma boa opção), mas será interessante observar se o Mesosphere vai tentar posicionar o DCOS como um sistema de propósito mais geral.

Em contraste com a nuvem moderna e soluções de contêiner baseadas em Linux, até mesmo o Windows Server Core é grande e pesado. A Microsoft está reagindo e liberou os primeiros previews do **Nano Server**, uma versão mais simples do Windows Server que oferece GUI stack, suporte para Win32 32-bit, logins locais e suporte a desktop remoto, tendo como resultado tamanho em disco aproximado de 400MB. Os primeiros previews são difíceis de trabalhar e a solução final será restrita ao uso do CoreCLR, mas para empresas interessadas em executar soluções baseadas em .NET, o Nano Server definitivamente merece ser examinado nesse estágio.

Presto é um mecanismo de consulta SQL distribuído de código aberto projetado e otimizado para executar cargas de trabalho de análise interativa. A arquitetura de processamento massivamente paralela do Presto

PLATAFORMAS *continuação*

– combinada com técnicas avançadas de geração de código e pipelines de processamento na memória – o torna altamente escalável. Ele suporta um grande subconjunto de ANSI SQL, incluindo consultas complexas, junções, agregações e funções de janela. O Presto oferece suporte para uma grande gama de fontes de dados, incluindo Hive, Cassandra, MySQL e PostgreSQL, unindo assim a interface de análise interativa aos armazenamentos de dados de uma empresa. Aplicações podem se conectar ao Presto usando sua interface JDBC.

Uma das nossas reclamações comuns é colocar inteligência de negócio em middleware, o que resulta em servidores de aplicações e barramentos de serviços corporativos que ambicionam executar uma lógica de aplicação crítica. Isso requer programação complexa em ambientes que não atendem bem a esse objetivo. Temos observado um preocupante reaparecimento dessa condição com produtos de **API Gateways excessivamente ambiciosos**. API Gateways podem ser úteis para lidar com algumas preocupações genéricas – por exemplo, autenticação e limitação de variação – mas quaisquer inteligência de domínio como transformação de dados ou processamento de regras devem residir em aplicações ou serviços nos quais eles possam ser controlados por times de produto trabalhando próximos aos domínios que eles suportam.

Temos observado ganhos de produtividade incontestáveis vindos da implantação de aplicações e serviços em provedores de nuvem maduros. Muito desse ganho se dá pela capacidade de times de implantar e operar seus próprios serviços com um alto grau de autonomia e responsabilidade. Estamos hoje nos deparando regularmente com ofertas de **nuvens privadas superficiais** em empresas nas quais plataformas básicas de virtualização estão sendo rotuladas como “nuvens”. É comum que times consigam auto-fornecer um conjunto restrito de tipos de serviço fixos, com acesso limitado e capacidade reduzida de personalizar as convenções corporativas gerenciadas de forma central, o que resulta em soluções improvisadas. O andamento da implantação frequentemente fica limitado por infraestruturas manualmente gerenciadas, como rede, firewall e armazenamento. Encorajamos empresas a considerar com mais consistência os custos do uso de uma oferta inadequada de nuvem privada.

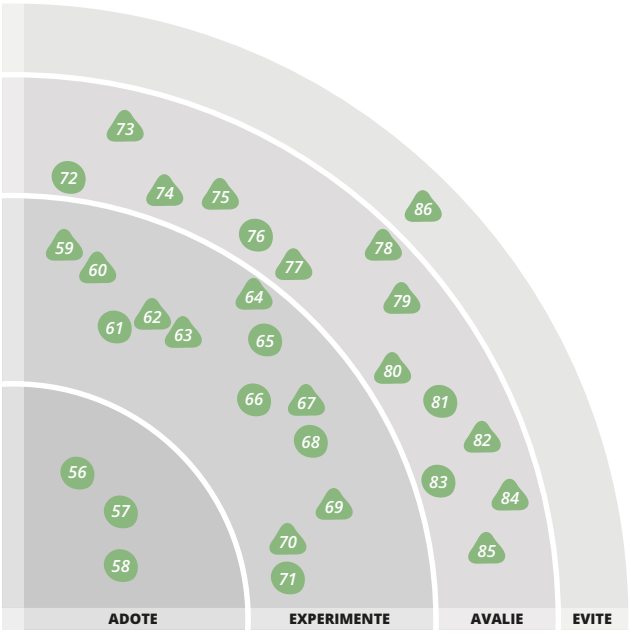
FERRAMENTAS

Temos recebido elogios de vários times de ThoughtWorkers que estão usando o **Browsersync**. Se o número de dispositivos para os quais desenvolvemos aplicações web cresce, cresce também o nível de esforço que deve ser despendido aos testes para esses diferentes dispositivos. Browsersync é uma ferramenta gratuita de código aberto que pode reduzir drasticamente esse esforço ao sincronizar testes manuais em múltiplos navegadores mobile ou desktop. Oferecendo opções tanto CLI quando UI, essa ferramenta é amigável a build pipelines e automatiza tarefas repetitivas como preenchimento de formulários.

O gerenciamento de dependências em projetos iOS e OS X costumava ser inteiramente manual ou interamente automático, quando usado o CocoaPods. Com **Carthage**, abriu-se a possibilidade de um meio-termo. Carthage gerencia dependências – baixa, constrói e atualiza frameworks – mas deixa a integração dos frameworks na compilação do projeto para o projeto. Isso em comparação com o CocoaPods, que basicamente assume o controle da estrutura do projeto e da configuração da compilação. Deve-se observar que o Carthage pode ser usado apenas com frameworks dinâmicos, que não estão disponíveis no iOS 7 e versões anteriores.

Anteriormente, recomendamos boot2docker como uma forma de executar facilmente o Docker em sua máquina Windows ou OS X. **Docker Toolbox** substitui boot2docker, adicionando ainda algumas ferramentas. Incluídos agora, o Kitematic gerencia contêineres e o Docker Compose gerencia configurações multi-Docker (apenas para Mac). Pode ser usado com segurança como um substituto imediato para o boot2docker, e irá até mesmo conduzir a atualização para você.

Armazenar segredos – como senhas e tokens de acesso – com segurança em repositórios de código é atualmente suportado por um número crescente de



ferramentas, por exemplo git-crypt e Blackbox, as quais mencionamos na edição anterior do Radar. Apesar da disponibilidade dessas ferramentas, infelizmente ainda é muito comum que segredos sejam armazenados de forma desprotegida. Na verdade, é tão comum, que softwares maliciosos automatizados são usados para encontrar credenciais AWS e iniciar instâncias EC2 para minerar Bitcoins, deixando as Bitcoins para o invasor e a conta a pagar para o usuário. **Gitrob** usa uma abordagem semelhante e escaneia os reporsitórios GitHub de uma empresa, sinalizando todos os arquivos que podem conter informações delicadas que não deveriam ter sido colocadas no repositório. Esta é obviamente uma abordagem de resposta. Gitrob pode apenas alertar times quando já é (quase) tarde demais. Por esse motivo, Gitrob deve ser vista apenas como uma ferramenta complementar, com intuito de minimizar estragos.

59. ADOTE
60. Composer
61. Mountebank
62. Postman

EXPERIMENTE
63. Browsersync
64. Carthage
65. Consul
66. Docker Toolbox
67. Gitrob
68. GitUp
69. Hamms
70. IndexedDB
71. Polly
72. REST-assured
73. Sensu
74. SysDig
75. ZAP

AVALIE
76. Apache Kafka
77. Concourse CI
78. Espresso
79. Gauge
80. Gor
81. ievms
82. Let's Encrypt
83. Pageify
84. Prometheus
85. Quick
86. RAML
87. Security Monkey
88. Sleepy Puppy
89. Visual Studio Code

EVITE
90. Citrix para desenvolvimento

FERRAMENTAS *continuação*

O Git pode ser confuso. Muito confuso. E até mesmo quando é usado em um simples processo de desenvolvimento realizado no tronco do projeto, ainda há tantas nuances sobre seu funcionamento que é fácil se perder às vezes. Quando isso acontece, ter uma boa compreensão de como o Git funciona debaixo dos panos é muito útil. **GitUp** é uma ferramenta para Macs que proporciona exatamente isso. O GitUp fornece uma representação gráfica do que está acontecendo à medida que você digita comandos normais do Git no terminal. Você pode aprender os diversos comandos do Git e ao mesmo tempo entender o que cada um faz conforme você usa. GitUp é uma ferramenta útil tanto para novos usuários de Git quando para usuários mais experientes.

Vários dos nossos times que trabalham em projetos .NET recomendaram **Polly** como uma ferramenta útil para a construção de sistemas baseados em microsserviços. Polly encoraja a expressão fluente de políticas transitórias de tratamento de exceção e do padrão Circuit Breaker, incluindo políticas como Retry, Retry Forever e Wait and Retry. Bibliotecas similares já existem em outras linguagens (Hystrix para Java, por exemplo) e Polly é uma adição bem-vinda para a comunidade .NET. Apresentando boa integração com Polly há o **Brighter**. Brighter é outra pequena biblioteca .NET de código aberto que fornece recursos básicos para implementar chamadas de comando. Combinar as duas bibliotecas proporciona uma funcionalidade útil de quebra de circuito, principalmente no contexto do padrão Ports and Adaptors e CQRS. Embora possam ser usados separadamente, no cenário atual nossos times acreditam que eles funcionem bem juntos.

Muitas ferramentas de monitoramento são construídas em torno do conceito de máquina ou instância. O crescente uso de padrões como Servidor Fênix e ferramentas como Docker mostra que esta é uma forma cada vez menos útil de modelar infraestruras: instâncias estão se tornando transientes enquanto serviços são persistentes. **Sensu** permite que uma instância se registre para desempenhar uma função específica e o Sensu então a monitora nessa condição. Ao longo do tempo, diferentes instâncias desempenhando aquela função podem ir e vir. Dados esses fatores e a crescente maturidade da ferramenta, sentimos que era hora de trazer o Sensu de volta ao Radar.

Embora o **SysDig** não seja uma novidade no Radar, ainda estamos surpresos com a quantidade de pessoas que não conhece a ferramenta. Funcionando como um CLI plugável de código aberto para rastrear e resolver problemas em sistemas Linux, o SysDig possui algumas funcionalidades poderosas. Uma das principais características de que gostamos é a capacidade de gerar dados de rastreamento do sistema em uma máquina que está enfrentando dificuldades, que podem ser investigados posteriormente para que se descubra o que estava acontecendo. SysDig também oferece suporte para trabalhar com contêineres, o que torna a ferramenta já útil em uma ferramenta ainda mais poderosa.

Muitos times de desenvolvimento estão migrando de servidores simples de integração contínua para pipelines de Entrega Contínua, normalmente abrangendo múltiplos ambientes e alcançando a produção. Para implementar tal pipeline com sucesso e operá-la de forma sustentável é necessária uma ferramenta de CI/CD que trata build pipelines e artefatos como cidadãos de primeira classe, e infelizmente não há muitas. **Concourse CI** é um estreante promissor nesse campo e nossos times que o testaram estão animados com sua configuração, que permite builds que executam em contêineres, possui uma interface limpa e usável e desencoraja servidores de compilação não reproduzíveis.

Espresso é uma ferramenta de testes funcionais para Android. Sua API principal sucinta esconde os confusos detalhes de implementação e ajuda a escrever testes concisos, com execução mais rápida e confiável.

Gauge é uma ferramenta leve de automação de testes multiplataforma. As especificações são escritas em Markdown, para que casos de teste possam ser escritos na linguagem de negócio e possam ser incorporados a um formato existente de documentação. As linguagens suportadas são implementadas como plugins a um núcleo único, o que assegura consistência entre implementações de linguagem. Essa ferramenta, com o código aberto pela ThoughtWorks, também suporta execução paralela para todas as plataformas suportadas.

Apesar do uso decrescente do Internet Explorer, para muitos produtos, a base de usuários do IE não é uma parcela insignificante do mercado, e a compatibilidade do navegador precisa ser testada. Isso é particularmente problemático caso você prefira as vantagens de um sistema para desenvolvimento baseado em UNIX. Para contornar esse dilema, o **ievms** oferece um script utilitário que une imagens VM distribuídas pelo Windows e VirtualBox para automatizar a configuração e a testabilidade de várias versões do IE, do 6 ao Edge.

FERRAMENTAS *continuação*

Embora um número crescente de sites esteja implementando HTTPS para auxiliar na proteção de seus usuários e melhorar a integridade da web como um todo, existem muitos outros sites ainda por adotar. Somando-se a isso, vemos mais e mais pessoas usando HTTPS em suas empresas para proporcionar garantias de segurança adicionais. Um dos principais obstáculos para uma adoção mais ampla tem sido o processo de obtenção do certificado. Além do custo, o processo em si está longe de ser simples. **Let's Encrypt**, uma nova Autoridade de Certificação, busca resolver essa questão. Primeiro, fornecendo certificados gratuitos. Segundo, e indiscutivelmente mais importante, fornecendo uma API de linha de comando extremamente fácil de usar, facilitando a automação do processo de emissão, atualização e instalação de certificados. Acreditamos que o Let's Encrypt, em versão beta no momento, pode ser revolucionário no sentido de ajudar uma porção maior da web a adotar HTTPS, ao mesmo tempo mostrando como devem ser boas ferramentas automatizáveis para quem se preocupa com segurança.

Pageify é uma biblioteca Ruby para construção de Page Objects para testes de automação de UI, tendo como foco execuções de testes mais rápidas e legibilidade do código. Ela oferece APIs simples para definir, operar e declarar Page Objects de forma dinâmica, permitindo códigos legíveis mesmo ao lidar com elementos com hierarquias complexas no DOM. Permite integração com **WebDriver** e **Capybara**.

O SoundCloud recentemente abriu o código de seu conjunto de ferramentas de monitoramento e alerta, o **Prometheus**. Desenvolvido como resposta às dificuldades com Graphite em seus sistemas em produção, o Prometheus essencialmente suporta um modelo HTTP baseado em pull (embora modelos push no estilo do Graphite também sejam suportados). Prometheus também vai além ao dar suporte a alertas, tornando-se parte ativa do seu conjunto de ferramentas operacionais. Até a data dessa publicação, o Prometheus estava apenas no release 0.15.1, mas evoluindo muito rápido. Estamos satisfeitos de ver o recente foco do produto em bancos de dados de séries temporais e recursos de indexação multidimensionais, também permitindo a exportação para uma maior variedade de ferramentas gráficas de front-end.

Com um crescente panorama de serviços oferecendo APIs RESTful, está se tornando cada vez mais importante documentá-las. Já mencionamos anteriormente o Swagger, e nesta edição do Radar gostaríamos de destacar a linguagem de modelagem de APIs RESTful (**RAML**). Nossos times acreditam que, em comparação com Swagger, ela seja mais leve e mude o foco de adicionar documentação a APIs existentes para projetar APIs.

Sleepy Puppy é um framework de gestão de delayed cross-site scripting (XSS) payload cujo código foi recentemente aberto pelo Netflix. Permite testar vulnerabilidades para XSS que vão além da aplicação destino, quando o invasor pretende atacar um sistema subjacente secundário. Sendo o XSS um dos top 10 do OWSAP, vemos esse framework como um auxiliar para verificações de segurança automatizadas em várias aplicações. Simplifica a captura, gestão e rastreio da propagação do XSS durante longos períodos de tempo, com corpos de dados personalizáveis. Sleepy Puppy também exibe uma API que pode ser integrada com ferramentas de vulnerabilidade como ZAP, para checagens de segurança automatizadas.

Visual Studio Code é o editor/IDE gratuito da Microsoft, disponível para várias plataformas. Achamos a integração de controle de versão com Git muito positiva para promover práticas de integração contínua. O Visual Studio Code também fornece um meio de integração com ferramentas externas através de tarefas, com autodetecção de tarefas grunt/gulp, eliminando a necessidade de executar tarefas grunt/gulp via terminais, simplesmente usando o editor. Com o crescimento do ecossistema do Docker, esse IDE oferece suporte para o dockerfile com snippets e definições de comandos válidos.

Muitas empresas ainda estão insistindo para que times de desenvolvimento usem **Citrix desktop remoto para desenvolvimento**. Embora isso proporcione um modelo de segurança simples – ativos teoricamente nunca deixam os servidores da empresa – usar desktops remotos para desenvolvimento prejudica imensamente a produtividade de quem desenvolve. Não há muito sentido em pagar um valor-hora mais barato para profissionais de desenvolvimento se você pretende impor os fardos da distribuição e do desktop remoto, e esperamos que fornecedores offshore admitam essas desvantagens a seus clientes. É muito melhor usar um ambiente offshore limpo e seguro, no qual desenvolvimento local possa ser realizado, ou um IDE hospedado, por exemplo ievms.

LINGUAGENS & FRAMEWORKS

Por muitos anos, o JavaScript cresceu para se tornar provavelmente a linguagem de programação mais usada no mundo. Todavia, a linguagem em si tem alguns problemas que muitas pessoas já tentaram resolver usando bibliotecas ou até mesmo implementando suas próprias linguagens executadas a partir do JavaScript (das quais mencionamos CoffeeScript e ClojureScript. **ECMAScript 6**, a nova versão do JavaScript, resolve muitos dos problemas das versões anteriores atualmente em uso. Embora o suporte a navegadores deixe a desejar, o suporte de transpilers maduros como Babel permite programar no ECMAScript 6 e viabilizar seu suporte em navegadores mais antigos. Para novos projetos, sugerimos veementemente usar ECMAScript 6 desde o início.

Um ano depois de seu lançamento para o público, **Swift** é agora nossa escolha padrão para desenvolvimento no ecossistema Apple. Com a recente liberação do Swift 2, a linguagem se aproxima de um nível de maturidade que proporciona a estabilidade e a performance necessárias para a maior parte dos projetos. O Swift ainda tem problemas, principalmente em relação a suporte de ferramentas, refatoração e testes. Entretanto, acreditamos que estes não sejam significativos o suficiente para evitá-lo. Ao mesmo tempo, portar grandes bases de código Objective-C existentes dificilmente vale a pena. O anúncio de que o Swift vai ser tornar um software de código aberto é outro sinal positivo. Esperamos que isso não seja apenas mais um despejo de código internamente desenvolvido em um repositório público, já que a Apple afirmou claramente que contribuições da comunidade são incentivadas e serão aceitas.

A maioria dos frameworks de template como Mustache ou FreeMarker mistura código com marcação em um único arquivo para implementar conteúdo complexo e dinâmico. **Enlive** é um framework de template baseado em Clojure que separa completamente linguagem de programação de marcação HTML e usa seletores de CSS para localizar e substituir partes do documento. O Enlive demonstra a força da programação funcional para implementar comportamento complexo através de uma série de funções simples e combináveis atuando em uma abstração comum. Nossos times trabalhando com Clojure o têm achado uma ferramenta muito útil e objetiva.

Temos várias ressalvas em relação ao uso de WebSockets HTML5. Permitindo que o servidor inicie ações no navegador, WebSockets se afasta do modelo requisição/resposta com conexões transientes que sustenta a World Wide Web hoje. Segurança é outro grande risco associado com WebSockets. Por exemplo, o padrão não impõe nenhuma política de requisições de origem cruzada (cross-origin). Entretanto, reconhecemos que em determinadas aplicações de monitoramento ou alerta, WebSockets pode ser muito útil. Caso você precise construir um servidor de WebSockets .NET, **SignalR** convenientemente implementa boa parte do código adicional que você precisa para uma aplicação de produção robusta. Isso inclui algumas práticas de segurança recomendadas, como validação de tokens de conexão e ativação SSL quando criptografia é necessária. Embora os times da ThoughtWorks

ADOTE
91. ECMAScript 6
92. Nancy
93. Swift

EXPERIMENTE
94. Enlive
95. React.js
96. SignalR
97. Spring Boot

AVALIE
98. Axon
99. Ember.js
100. Frege
101. HyperResource
102. Material UI
103. OkHttp
104. React Native
105. TLA+
106. Traveling Ruby

EVITE

LINGUAGENS & FRAMEWORKS *continuação*

estejam muito satisfeitos com SignalR, ainda existem problemas fundamentais com WebSockets que devem ser considerados antes de abraçar completamente a ideia.

Spring Boot permite uma configuração simples de aplicações independentes baseadas em Spring. É ideal para erguer novos microsserviços e fácil de implementar. Também torna o acesso a dados menos penoso, graças aos mapeamentos Hibernate com menos código duplicado. Gostamos do fato de que o Spring Boot simplifica serviços Java construídos com Spring, mas aprendemos a ser cautelosos com as muitas dependências. O Spring ainda está por debaixo dos panos. Caso esteja escrevendo microsserviços com Java, você também pode considerar usar **DropWizard** ou um microframework como Spark para obter os benefícios do Spring Boot sem o enorme peso do Spring.

Embora ainda tenhamos algumas reservas em relação ao CQRS como um padrão geral, a abordagem pode funcionar muito bem em lugares específicos. Nessas situações específicas, no entanto, sobra muito trabalho para quem desenvolve executar o CQRS corretamente. **Axon** é um framework que pode ajudar com isso na JVM e já obtivemos algum sucesso utilizando-o. Ainda que certamente não possa ser considerado uma solução perfeita atualmente, Axon continua a evoluir e pode fazer muito mais sentido que tentar escrever tudo do zero.

Seguindo o caminho de muitas outras linguagens de programação, uma das favoritas absolutas dos geeks, Haskell, agora está disponível também na JVM, através do **Frege**. Isso adiciona uma linguagem de programação puramente funcional à plataforma, permitindo fácil interoperabilidade com outras linguagens e bibliotecas JVM.

HyperResource é um framework Ruby para construção de clientes de APIs RESTful. O framework aceita JSON em formato **HAL** e gera de forma dinâmica um objeto modelo completo com links hipermídia. Embora o framework ainda seja muito novo, gostamos do fato de ele adotar REST nível 3 no modelo de maturidade de Richardson para maior visibilidade do serviço e protocolos autodocumentados.

Material UI oferece componentes reutilizáveis para uso em aplicações React que implementam a linguagem Material Design do Google. Ocupando um espaço similar ao do Twitter Bootstrap, ele facilita um rápido início de funcionamento, mas não apresenta os mesmos inconvenientes à medida que a aplicação cresce. Elemental UI é uma alternativa que vale uma investigação.

OkHttp é uma biblioteca em Java para comunicação via HTTP do Square que fornece uma interface fluente para criar conexões, bem como suporte para o protocolo HTTP/2, mais rápido. Mesmo usando HTTP/1.1, OkHttp pode proporcionar melhorias de desempenho através de pooling de conexão e compressão gzip transparente. Suportanto tanto chamadas síncronas bloqueantes quanto assíncronas não-bloqueantes, também pode ser usado como um substituto imediato para o amplamente usado HttpClient da Apache.

Mais um estreante no mundo do desenvolvimento mobile multiplataforma, o **React Native** do Facebook leva o modelo de programação React.js a quem desenvolve para iOS e Android. Programas React Native são escrito em JavaScript, mas ao contrário de um framework híbrido como Ionic, React Native proporciona a quem desenvolve acesso a componentes nativos de UI na plataforma de destino. Esta é uma abordagem que já vimos antes (Calatrava, por exemplo), mas React Native já inspirou uma porção significativa da comunidade de desenvolvimento, e se apoia no ímpeto gerado pelo React.js. Esse framework pode assumir um papel importante no futuro do desenvolvimento de aplicativos móveis.

Construir sistemas usando microsserviços requer que pensemos mais profundamente sobre isolamento de falhas e testes. **TLA+** é uma linguagem de especificação formal que pode ser útil em ambos esses cenários. Para isolamento de falhas, TLA+ pode ser usada para identificar invariantes em seu sistema, que podem ser monitoradas diretamente. Uma invariante pode ser a razão entre o número de requisições para um serviço e o número de requisições para um segundo serviço, por exemplo. Qualquer alteração nessa razão provocaria um alerta. TLA+ também tem sido usada para identificar falhas de design sutis em sistemas distribuídos. A Amazon, por exemplo, usava checagem de modelo baseada em uma especificação formal escrita em TLA+ para identificar bugs sutis no Dynamo DB antes de seu lançamento para o público. Para a maioria dos sistemas, o investimento necessário para criar a especificação formal e depois executar a checagem de modelo é provavelmente muito alto. Entretanto, para sistemas críticos – complexos ou com muitos usuários –, acreditamos que seja muito válido ter outra ferramenta em seu conjunto.

Traveling Ruby possibilita a distribuição de binários Ruby de forma portátil, pronta para execução e independente de plataforma, sem a necessidade de instalação de um interpretador, pacotes ou gems adicionais. Dissocia ainda a execução de aplicações Ruby do ambiente de desenvolvimento no qual elas são executadas.

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®