ThoughtWorks®

TECHNOLOGY RADAR

Nossas ideias sobre as tecnologias e tendências que moldam o futuro

JANEIRO 2015

thoughtworks.com/radar

O QUE HÁ DE NOVO?

Aqui estão as tendências em destaque nesta edição:

CRESCIMENTO EXPLOSIVO NA ARENA DEVOPS

Muito do trabalho neste radar envolveu avaliar um mar de tecnologias relacionadas a DevOps, que continua crescendo e inovando a uma velocidade estonteante. Com o advento da contentorização, das ofertas em nuvem, e várias permutas e combinações, não é por acaso que a inovação nesse setor é enorme. O crescimento e a popularidade de estilos arquitetônicos como o de micro-serviços enfatiza a intersecção entre arquitetura e DevOps, portanto esperamos que o ritmo de inovação se mantenha.

PLATAFORMAS DE DADOS DE ÚLTIMA GERAÇÃO GANHAM TRAÇÃO

Grandes volumes de dados não são novidade - na verdade, temos falado em não entrar na onda por algum tempo - mas estamos percebendo que tecnologias relacionadas vêm se incorporando e sendo úteis na Empresa. Técnicas como Data Lake e Arquitetura Lambda são novas maneiras de olhar para plataformas de dados corporativos, e são úteis independente do seu volume de dados.

DESENVOLVEDORES FOCADOS EM FERRAMENTAS DE SEGURANÇA

Toda semana aparecem novas histórias sobre vazamento ou mau uso de dados, e a demanda por sistemas seguros que respeitem a privacidade dos dados está crescendo. Blackbox, autenticação de duplo fator TOTP e OpenID Connect - todas ferramentas abordadas nesta edição do Radar - ajudam os desenvolvedores a criarem sistemas e infraestruturas seguros.

CONTRIBUIDORES

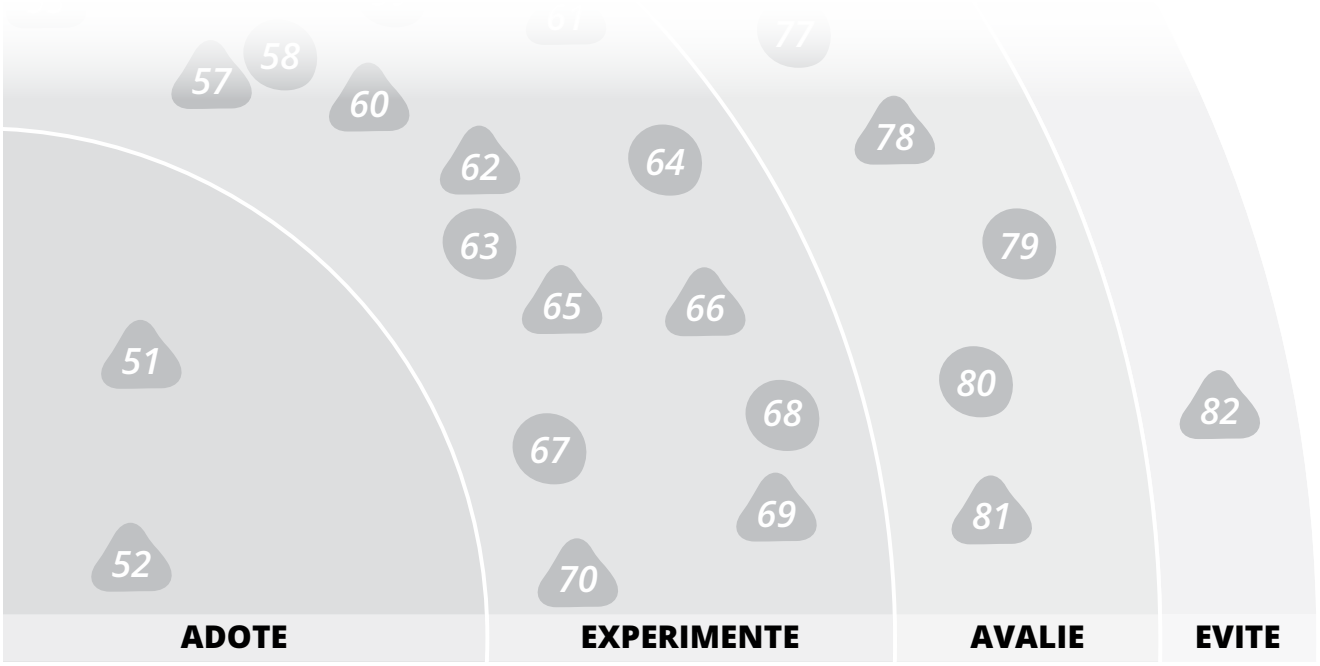
O TAB da ThoughtWorks é composto por:

Rebecca Parsons (CTO)
Martin Fowler (Cientista Chefe)
Badri Janakiraman
Brain Leke
Claudia Melo
Erik Doernenburg
Evan Bottcher
Hao Xu
Ian Cartwright
James Lewis
Jeff Norris
Jonny LeRoy
Mike Mason
Neal Ford
Rachel Laycock
Sam Newman
Scott Shaw
Srihari Srinivasan
Thiyagu Palanisamy

SOBRE O TECHNOLOGY RADAR

'ThoughtWorkers' são apaixonados por tecnologia. Nós desenvolvemos, pesquisamos, testamos, contribuímos com código livre, escrevemos e visamos a sua constante melhoria - para todos. A 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, criou o radar. Eles se reúnem 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 interessados, de CIOs a desenvolvedores. O conteúdo é concebido para ser um resumo conciso. Nós o encorajamos a explorar essas tecnologias para obter mais detalhes. O radar é gráfico por natureza, agrupando os itens em técnicas, ferramentas, plataformas, linguagens e frameworks. Quando itens do radar puderem ser classificados em mais de um quadrante, escolhemos aquele que parece mais adequado. Além disso, agrupamos esses itens em quatro anéis para refletir a nossa posição atual em relação a eles:



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 a 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. Nos interessamos em muito mais itens do que seria razoável em um documento desse 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. Foco no tempo médio de recuperação
2. Forward Secrecy
3. Logs estruturados

EXPERIMENTE
4. Implantações Canário
5. Datensparsamkeit
6. Instrumentação de front-end
7. Batch hipster
8. Registros humanizados
9. Manobra inversa de Conway
10. Guia de estilo de CSS dinâmico
11. Sincronização de armazenamento local
12. NoPSD
13. Partição de infraestrutura alinhada com limites do time
14. REST sem PUT
15. Geradores de sites estáticos
16. Modelo de serviços sob medida

AVALIE
17. Armazenamento de dados append-only
18. Blockchain além do bitcoin
19. Data Lake corporativo
20. Pipelines de imagem de máquina
21. Estratégia de desenvolvimento em camadas

EVITE
22. Subir e migrar para máquina
23. Branches de vida longa com Gitflow
24. Inveja de micro-serviços
25. Programar em sua ferramenta de integração contínua / entrega contínua
26. SAFe™
27. Sanduíche de segurança
28. DevOps como um time
29. Testes como uma organização separada
30. Velocidade como produtividade

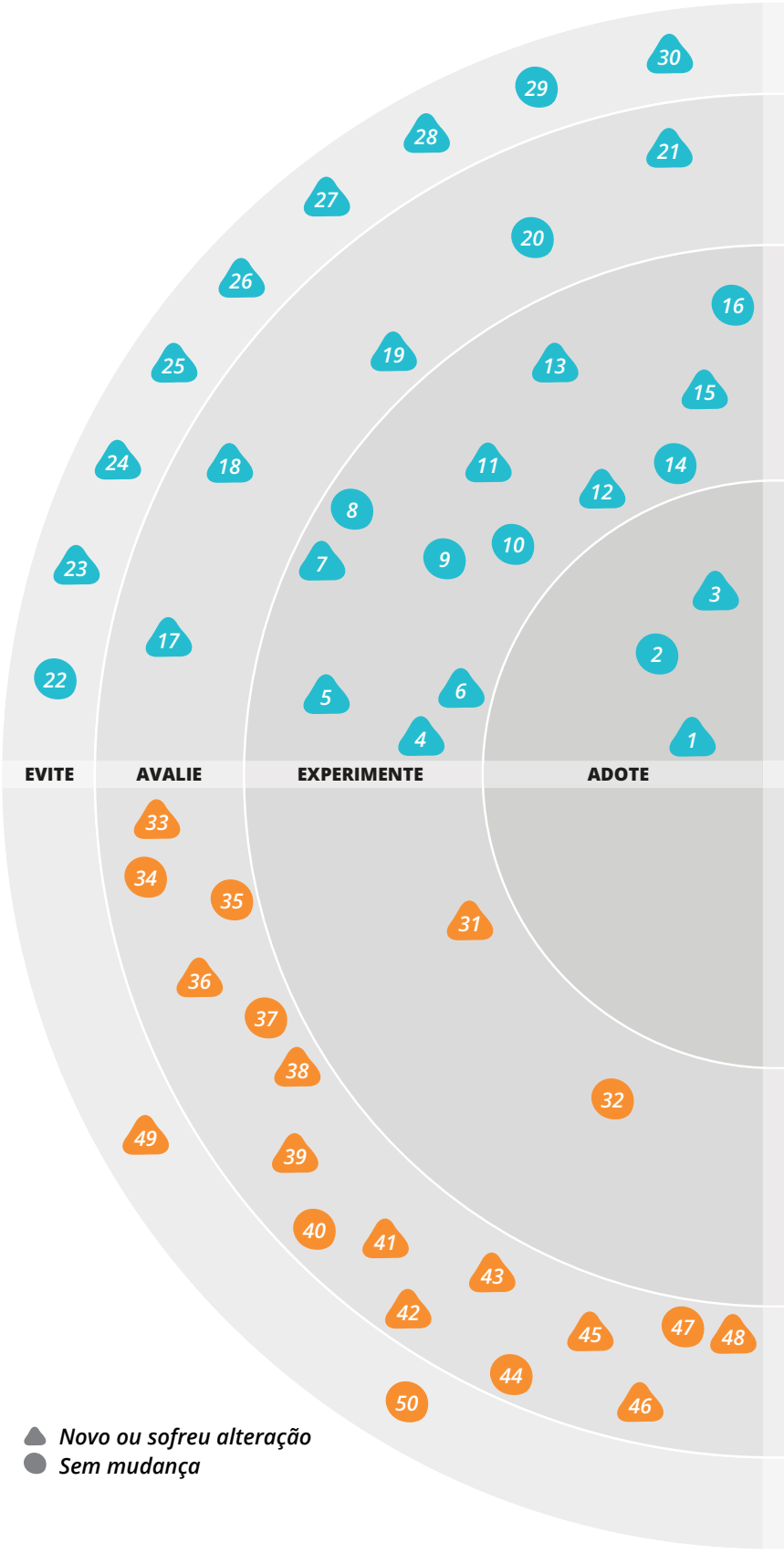
PLATAFORMAS

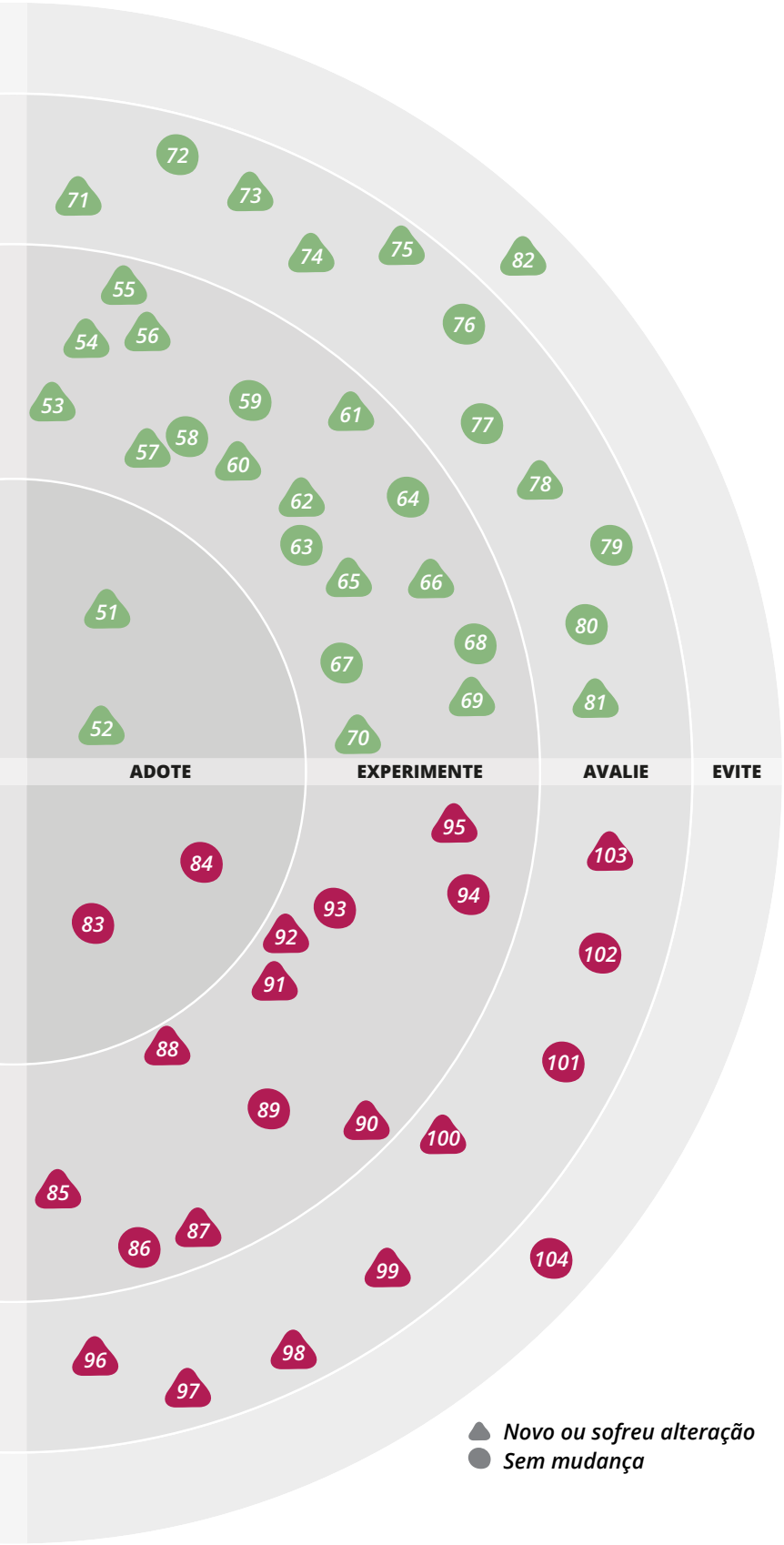
ADOTE

EXPERIMENTE
31. DigitalOcean
32. iBeacon

AVALIE
33. Apache Mesos
34. ARM SoC
35. CoAP
36. CoreOS
37. EventStore
38. Jackrabbit Oak
39. Módulos de segurança do Linux
40. Mapbox
41. MariaDB
42. Netflix OSS Full stack
43. OpenAM
44. OpenID Connect
45. Redes definidas por software (SDN)
46. Text it as a service / RapidPro.io
47. Autenticação de dois fatores TOTP
48. U2F

EVITE
49. CMS como plataforma
50. OSGi





O RADAR

FERRAMENTAS

ADOTE
51. Flyway
52. Go CD

EXPERIMENTE
53. Appium
54. Boot2docker
55. Composer
56. Cursive
57. Docker
58. Foreman
59. GenyMotion
60. Gitlab
61. Grunt.js
62. IndexedDB
63. Packer
64. Pact & Pacto
65. Papertrail
66. Postman
67. SnapCI
68. Snowplow Analytics & Piwik
69. Swagger
70. Xamarin

AVALIE
71. Blackbox
72. Consul
73. DC.js
74. Gorilla REPL
75. Gulp
76. Leaflet.js
77. Mountebank
78. Packetbeat
79. Roslyn
80. Spark
81. Terraform

EVITE
82. Citrix para desenvolvimento

LINGUAGENS & FRAMEWORKS

ADOTE
83. A Linguagem Go
84. Java 8

EXPERIMENTE
85. AngularJS
86. Core Async
87. Dashing
88. Django Rest
89. HAL
90. Framework Ionic
91. Nashorn
92. Om
93. Q & Bluebird
94. R como plataforma de computação
95. Retrofit

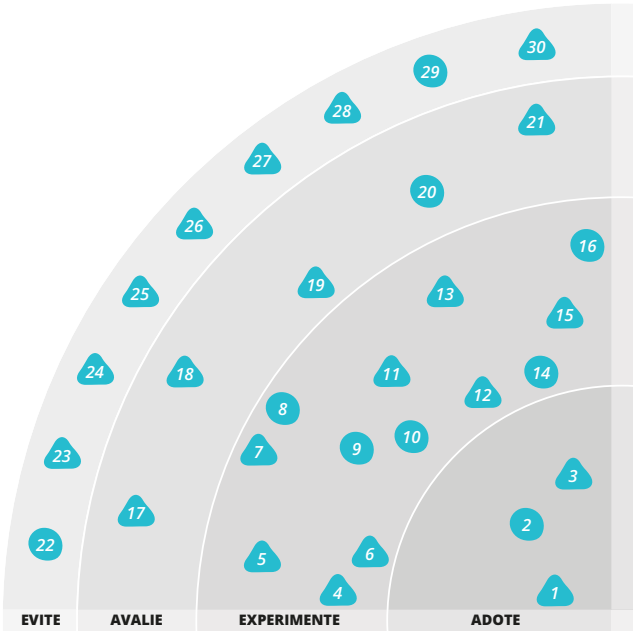
AVALIE
96. Flight.js
97. Biblioteca Haskell Hadoop
98. Lotus
99. React.js
100. Reagent
101. Rust
102. Spring Boot
103. Swift

EVITE
104. JSF

TÉCNICAS

Muitos projetos têm dependências de código externas, muitas das quais fornecidas por projetos de código aberto. Para garantir que os builds sejam replicáveis, integramos com versões conhecidas deles, mas isso pode significar que leve algum tempo para integrarmos com versões mais novas dessas bibliotecas, levando a esforços de combinação ainda maiores mais adiante. Uma abordagem que procuramos para evitar isso é uma **Implantação Canário** noturna, que tenta puxar a última versão de todas as dependências. Se o build estiver verde, sabemos que podemos atualizar as versões das quais dependemos.

O termo **Datensparsamkeit** (martinfowler.com/bliki/Datensparsamkeit.html) foi retirado da legislação alemã sobre privacidade e descreve a ideia de apenas armazenar informações pessoais que sejam absolutamente necessárias para o negócio ou exigidas pelas leis aplicáveis. A privacidade do cliente continua um tópico relevante. Empresas como a Uber (www.washingtonpost.com/blogs/the-switch/wp/2014/12/01/is-ubers-rider-database-a-sitting-duck-for-hackers) estão, aparentemente, coletando dados altamente pessoais de clientes, além de serem bem negligentes com a segurança. Isso é um desastre esperando para acontecer. Seguir datensparsamkeit ou usar técnicas de des-identificação (en.wikipedia.org/wiki/De-identification), mesmo em jurisdições onde não sejam legalmente obrigatórios, pode permitir a redução das informações armazenadas. Se você nunca armazenar as informações, não precisa se preocupar que alguém possa roubá-las.



O uso recente de feeds de eventos estilo ATOM em vez de HTTP como método de integração tem recebido bastante atenção. Ao invés de manter um serviço ao vivo para expor esses feeds, muitas vezes é aceitável a utilização de estilos antigos de processamento e batch para criar e publicar arquivos de feed. Quando combinado com a tecnologia em nuvem, como armazenamento de arquivos do Amazon S3 e linking em hipermídia, isso pode criar uma solução altamente disponível, além de simples e testável. Nossos times já começaram a chamar este encontro entre o velho e o novo de **"Batch hipster"**.

Ao implementar aplicações de página única, mais cedo ou mais tarde a questão do uso offline vai aparecer. Sabendo o quão difícil é inserir modo offline numa aplicação existente, há uma tendência para implementação de aplicações de página única com uma mentalidade "primeiro offline". Uma técnica importante

ADOTE
1. Foco no tempo médio de recuperação
2. Forward Secrecy
3. Logs estruturados

EXPERIMENTE
4. Implantações Canário
5. Datensparsamkeit
6. Instrumentação de front-end
7. Batch hipster
8. Registros humanizados
9. Manobra inversa de Conway
10. Guia de estilo de CSS dinâmico
11. Sincronização de armazenamento local
12. NoPSD
13. Partição de infraestrutura alinhada com limites do time
14. REST sem PUT
15. Geradores de sites estáticos
16. Modelo de serviços sob medida

AVALIE
17. Armazenamento de dados append-only
18. Blockchain além do bitcoin
19. Data Lake corporativo
20. Pipelines de imagem de máquina
21. Estratégia de desenvolvimento em camadas

EVITE
22. Subir e migrar para máquina
23. Branches de vida longa com Gitflow
24. Inveja de micro-serviços
25. Programar em sua ferramenta de integração contínua / entrega contínua
26. SAFe™
27. Sanduíche de segurança
28. DevOps como um time
29. Testes como uma organização separada
30. Velocidade como produtividade

TÉCNICAS *continuação*

de implementação que temos usado com sucesso é a **sincronização de armazenamento local**. Com essa técnica, o código de interface do usuário nunca faz requisições para o backend. Ele recupera dados apenas do armazenamento local. Um serviço em background sincroniza os dados do armazenamento local com os sistemas backend, geralmente utilizando chamadas para alguma API REST.

NoPSD (thoughtworks.github.io/p2/issue02/continuous-design) é um movimento para integrar as atividades de design nos ciclos iterativos de feedback necessários para a construção de grandes softwares. O nome visa desconstruir a ideia do PSD como artefato final de design, sem querer falar mal do software da Adobe. Ao invés de se prender a uma especificação de design no começo de um projeto, os times são convidados a adotar o Design Contínuo: envolver designers no time de entrega, usar técnicas lo-fi para prototipagem e colaborar no refinamento do design para a tecnologia de interface de usuário pretendida (normalmente HTML e CSS). Essa abordagem aumenta a velocidade de resposta ao feedback do usuário real, permitindo testes de design em diferentes dispositivos e formatos, empregando a natureza dinâmica tanto dos produtos digitais quanto do processo de criação de produtos.

Muitos dos nossos clientes têm feito do DevOps uma realidade nas suas organizações, com times de entrega que desenvolvem, implantam e suportam suas próprias aplicações e serviços. Infelizmente, um bloqueio constante nessa jornada é a permissão para que os times tenham privilégios de super usuários no ambiente de produção. Na maioria das organizações, o ambiente de produção é compartilhado, tornando arriscado o fornecimento de acesso amplo. É eficaz podermos fazer a **partição de infraestrutura alinhada com limites do time**, de forma que eles tenham acesso seguro e isolado para trabalhar, sem risco de impacto nos outros sistemas. Em ambientes em nuvem isso é muito mais simples de implementar, alinhando a estrutura de contas conforme os limites do time.

Geradores de sites estáticos como Middleman ou Jekyll tornaram-se populares para a criação de sites simples ou blogs, mas estamos vendo cada vez mais o seu uso como parte de stacks de aplicações mais complexas. A suposição padrão de que todo o conteúdo entregue através de HTTP deve ser criado dinamicamente por demanda está mudando, com mais times buscando usar conteúdo estático gerado previamente.

Estruturas de dados imutáveis estão se tornando cada vez mais populares, com linguagens funcionais como Clojure fornecendo imutabilidade por padrão. A imutabilidade permite que o código seja mais facilmente escrito, lido e pensado. Usando um **armazenamento de dados append-only** é possível conferir alguns desses benefícios na camada de base de dados, bem como simplificar auditoria e histórico de consultas. As opções de implementação podem variar, de um armazenamento de dados append-only específico como Datomic (www.datomic.com), até o simples uso de uma abordagem "append-don't-update" com um banco de dados tradicional.

Enquanto o aspecto monetário do Bitcoin e de outras criptomoedas ganha atenção na mídia, estamos igualmente animados com a possibilidade de usar o **Blockchain além do Bitcoin** e de transações financeiras. O Blockchain é um mecanismo para verificar o conteúdo de registros compartilhados sem depender de um serviço centralizado. Já vemos o Blockchain (tanto a tecnologia por baixo quanto o Bitcoin Blockchain público) sendo usado no núcleo de sistemas muito variados, como identidade, propriedade, manutenção de registros, votação, armazenamento na nuvem e até gestão de redes de dispositivos inteligentes. Se você está criando sistemas que requerem confiança através de redes decentralizadas, o Blockchain é uma tecnologia que vale considerar.

Um **Data Lake corporativo** é um repositório de dados imutáveis, em grande parte "brutos" e não processados, que atua como uma fonte para outros fluxos de processamento, mas que também é disponibilizado diretamente a um número significativo de consumidores técnicos internos através de algum mecanismo eficiente de processamento. Exemplos incluem HDFS ou HBase em um framework de processamento como Hadoop, Spark ou Storm. Podemos contrastar isso com um sistema típico que coleta dados brutos em um espaço muito restrito, que só é disponibilizado a esses consumidores como resultado final de um processo de ETL altamente controlado.

Abraçar o conceito do data lake é eliminar pontos de estrangulamento devido à falta de desenvolvedores ETL disponíveis ou projeto inicial excessivo do modelo de dados. É sobre empoderar desenvolvedores a criarem seus próprios pipelines de processamento de dados de forma ágil, quando e como precisarem (dentro de limites razoáveis). Por isso, tem muito em comum com outro modelo que apreciamos, o modelo de DevOps.

TÉCNICAS *continuação*

O **Gitflow** é um padrão rigoroso de branching para releases, usando Git. Embora não seja um padrão inerentemente ruim, frequentemente o vemos sendo usado incorretamente. Se os branches de features e desenvolvimento forem de vida curta e seu merge for feito com frequência, você realmente está usando o poder do git, que facilita essas atividades. Porém, o problema que vemos com frequência é que eles se tornam **branches de vida longa**, o que resulta nos temidos conflitos de merge dos quais muitas pessoas usam Git para escapar. Um merge é um merge, independente da ferramenta ou padrão de source control que você usa. Se você esperar mais que um ou dois dias para fazer o merge, poderá se deparar com um conflito de merge enorme. Isso se transforma em um problema real se você tem um time grande. Se você tem várias pessoas esperando para fazer o merge, isso pode causar um engarrafamento grave. Introduzir padrões como Gitflow requer a disciplina de fazer o merge com frequência para ter sucesso. Portanto, certamente use esse padrão, mas só se você tem disciplina para prevenir branches de vida longa.

Ainda estamos convencidos de que micro-serviços podem oferecer vantagens significativas às organizações, em termos de melhorar a autonomia do time e a alta frequência de mudanças. A complexidade que advém do fato de serem sistemas distribuídos requer níveis maiores de maturidade e investimento. Estamos preocupados que alguns times estejam com pressa em adotar micro-serviços sem o entendimento das mudanças em desenvolvimento, testes e operações necessárias para que sejam bem executados. Nosso conselho permanece simples. **Evite a inveja dos micro-serviços** e comece com um ou dois serviços antes de se apressar em desenvolver mais, permitindo que seus times se ajustem e entendam o correto nível de granularidade.

Um pré-requisito comum para fazer integração contínua tem sido ter um comando único que possa ser executado em um clone de seu repositório em uma máquina nova para obter um ambiente de trabalho que possa construir, testar e empacotar sua aplicação. Isso foca a atenção na automação, facilita o início do trabalho para novos usuários, reduz as chances de ambientes de build únicos e difíceis de reproduzir e resulta em builds confiáveis que podem ser recriados sob demanda tanto no ambiente de integração contínua como localmente.

No entanto, muitas vezes vemos times ignorando isso e recorrendo à **programação em sua ferramenta de integração contínua/entrega contínua**, o que resulta não só em falhas de build difíceis de depurar, como na perda do benefício de ter um ponto de entrada único para a configuração e desenvolvimento de seus projetos. Nesta versão do radar, gostaríamos de recomendar fortemente a não adoção dessa prática que sacrifica muitos dos benefícios da integração contínua por conta da conveniência.

Tornar-se ágil é um desafio contínuo para as empresas. Várias abordagens vem sendo propostas, com o **SAFe™** ganhando grande destaque. Ainda que o SAFe™ forneça uma lista útil de áreas de foco, elas são facilmente mal utilizadas, com a introdução das mesmas tendências de lançamentos grandes, como lançamento em cascata e processos de controle fechados, que a abordagem ágil remove. Empresas em particular buscam um grau de conformidade entre empreendimentos que o SAFe™ pode fornecer, promovendo padronizações agressivamente quando algum grau de customização produz valor significativo. Outras abordagens Lean que incluem a experimentação e a incorporação de práticas de melhorias contínuas, como Katas de aprendizagem, oferecem um modelo muito melhor de transformação ágil. Scaled Agile Framework® e SAFe™ são marcas registradas da Scaled Agile, Inc.

As abordagens tradicionais de segurança têm contado com especificação de up-front seguida de validação no final. Esse "**sanduíche de segurança**" é difícil de integrar em times ágeis, uma vez que grande parte do design acontece durante todo o processo e não aproveita as oportunidades de automação que a entrega contínua oferece. As organizações devem ir em busca de como injetar práticas de segurança em todo o ciclo de desenvolvimento ágil. Isso inclui: avaliar o nível certo de modelação de ameaça para fazer up-front; quando classificar preocupações de segurança como estórias em si, critérios de aceitação, ou requisitos não funcionais transversais; inclusão de testes dinâmicos e estáticos de segurança automáticos na sua pipeline de construção; e maneiras de incluir testes profundos, como testes de penetração, em atualizações em um modelo de entrega contínua. Assim como o DevOps reformulou a maneira como grupos historicamente adversários podem trabalhar juntos, o mesmo está acontecendo com os profissionais de segurança e desenvolvimento.

PLATAFORMAS

A **Mesos** (mesos.apache.org) é uma plataforma que abstrai recursos computacionais de camadas subjacentes, facilitando o desenvolvimento de sistemas distribuídos bastante escaláveis. Pode ser usada para prover 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 escalonar sua infraestrutura, e ferramentas construídas a partir do Mesos já começam a aparecer, como o Chronos (nerds.airbnb.com/introducing-chronos), um substituto do cron tolerante a falhas.

O **CoreOS** é uma distribuição Linux e foi projetado para rodar sistemas grandes e escaláveis. Todas as aplicações implementadas em uma instância do CoreOS rodam em containers Docker separados, e o CoreOS fornece uma suíte de ferramentas que ajudam a gerenciá-los, incluindo o etcd, seu próprio armazenamento de configurações distribuídas. Serviços mais novos, como o fleet, auxiliam o gerenciamento de cluster ao garantir que um número específico de instâncias de serviço esteja sempre rodando. O FastPatch permite atualizações atômicas do CoreOS usando um esquema de root partition ativo-passivo e auxilia com rollbacks rápidos em caso de problemas. Estes novos desenvolvimentos fazem com que o CoreOS mereça uma chance se você já está confortável com o Docker.

O **Jackrabbit Oak** (jackrabbit.apache.org/oak), anteriormente chamado Jackrabbit 3, é uma implementação escalável e de alto desempenho de repositório de conteúdo para uso como base de sistemas de gerenciamento de conteúdo. Além da solução baseada em armazenamento de arquivos, suporta também os repositórios MongoDB e SGBD, preferidos em cenários de uso de alto volume. Embora implementado em Java, pode ser facilmente acessado em diversas plataformas através de padrões como JCR.

Ainda que a proteção do servidor seja uma técnica antiga, considerada bastante comum por administradores de sistemas que já geriram sistemas de produção, ainda não é comum entre a comunidade de desenvolvedores. No entanto, o aumento na cultura DevOps resultou em foco renovado em ferramentas como o SELinux, AppArmor e Grsecurity, que visam simplificar isso, pelo menos no ecossistema Linux. Cada uma dessas ferramentas vem com seus próprios pontos fortes e fracos e é difícil escolher uma como sendo

ADOTE

EXPERIMENTE
31. DigitalOcean
32. iBeacon

AVALIE
33. Apache Mesos
34. ARM SoC
35. CoAP
36. CoreOS
37. EventStore
38. Jackrabbit Oak
39. Módulos de segurança do Linux
40. Mapbox
41. MariaDB
42. Netflix OSS Full stack
43. OpenAM
44. OpenID Connect
45. Redes definidas por software (SDN)
46. Text it as a service / RapidPro.io
47. Autenticação de dois fatores TOTP
48. U2F

EVITE
49. CMS como plataforma
50. OSGi

PLATAFORMAS *continuação*

a única que você vai precisar. Dito isso, é altamente recomendável que todos os times pelo menos avaliem quais ferramentas funcionariam para eles e façam da segurança e proteção de servidores uma parte de seu fluxo de trabalho de desenvolvimento.

Depois da aquisição do MySQL pela Oracle, mais e mais módulos estão sendo adicionados na sua edição paga. Há algumas preocupações em relação ao futuro do MySQL. O **MariaDB** (mariadb.org) é um fork do MySQL, desenvolvido pela comunidade, com o objetivo de ser completamente código aberto para outros desenvolvedores, e ainda assim com total compatibilidade e competitividade com o MySQL. Alguns nomes de respeito como Google e Wikipédia já adotaram, assim como algumas distribuições Linux como RedHat e SUSE.

Por mais que hesitemos um pouco em recomendar a adoção completa do **Netflix OSS Full Stack** (a não ser que você esteja entrando na área de vídeo streaming de distribuição global), o stack está cheio de ideias interessantes, completo com implementações de código aberto. Algumas das ferramentas, como por exemplo Asgard, são altamente acopladas com uma arquitetura praticamente "chave na mão", tornando-as difíceis de usar individualmente. Outras ferramentas como Ice e Hystrix, que já mencionamos no radar anteriormente, podem ser usadas sozinhas. Acreditamos que os times devem entender as ideias e metodologias encapsuladas nessas ferramentas mesmo quando escolhem não usar o full stack.

Quando a Oracle cessou o desenvolvimento do OpenSSO da Sun, uma plataforma de gerenciamento de acesso de código aberto, a ForgeRock o integrou em sua suíte Open Identify. Agora chamado de **OpenAM** (forgerock.com/products/open-identity-stack/openam), ele atende à demanda por uma plataforma escalável e de código aberto que suporta uma variedade de padrões de federação de identificação, incluindo OpenID Connect e SAML 2.0. Esses padrões são extremamente necessários para a implementação segura de micro-serviços.

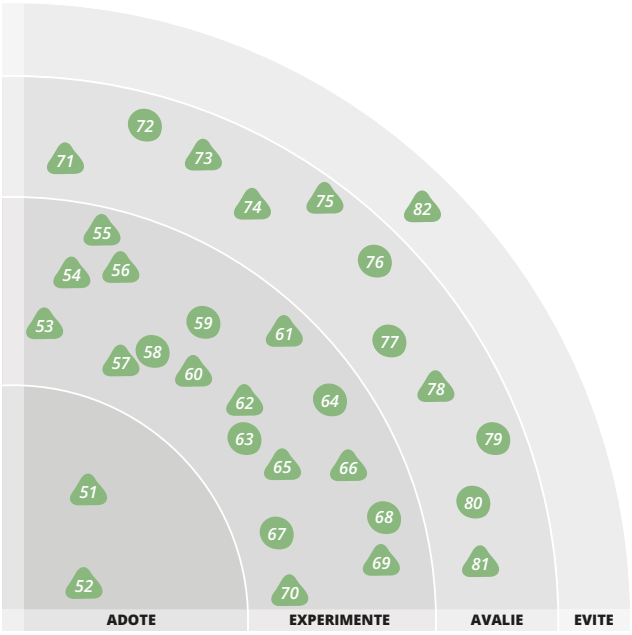
Redes definidas por software (**SDN**, do inglês Software Defined Networking) é um tema amplo, mas está se tornando cada vez mais importante. A capacidade de configurar nossos dispositivos de rede usando software está tornando difícil de definir os limites de onde terminam nossas implementações de aplicação. Isso engloba tudo, desde aparelhos de redes virtuais como balanceadores de carga da AWS ou Flannel de CoreOS (github.com/coreos/flannel), até equipamentos de rede que suportam padrões como OpenFlow (www.opennetworking.org/sdn-resources/onf-specifications/openflow). Enquanto os provedores de nuvem focavam anteriormente em computação e armazenamento, esperamos que a crescente gama de ferramentas SDN ofereça maior eficiência à forma como lidamos com os nossos sistemas, tanto locais quanto remotos.

Text-it-as-a-service / RapidPro (rapidpro.io) oferece a empresas a habilidade de facilmente criar ou modificar aplicações de serviço de mensagens curtas, mesmo que complexas, sem muita necessidade de desenvolvedores. Com custos de mensagens de texto mais baixos que de sessões USSD, isso fornece uma maneira menos custosa de construir aplicativos escaláveis direcionados a telefones celulares, e temos visto muito sucesso em nossos projetos. Os fluxos são simples de construir e as ações – como envio de SMS, e-mail ou até chamadas a APIs externas – podem ser convocadas a qualquer momento.

Segurança de contas online é ao mesmo tempo extremamente importante e notavelmente difícil. A autenticação em dois passos aumentou bastante a segurança, e temos recomendado TOTP como uma boa solução. Uma novidade nesse campo é o Universal 2nd Factor (**U2F**), uma solução baseada em criptografia de chave pública e tokens USB baratos. Inicialmente desenvolvido no Google, se tornou um padrão mantido pela aliança FIDO. Gostamos da promessa de melhor proteção contra phishing e man-in-the-middle attacks, mas estamos preocupados pois o padrão atualmente referencia um algoritmo específico de assinatura digital de curva elíptica que é considerado falho.

FERRAMENTAS

Com técnicas como a Entrega Contínua tornando-se mais difundidas, a migração automática de banco de dados é uma capacidade base para vários times de software. Embora existam várias ferramentas nessa área, continuamos recomendando **Flyway** pela sua abordagem simplificada. Flyway tem uma comunidade de código aberto vibrante, e suporte tanto para bancos de dados tradicionais quanto baseados em nuvem como o Amazon Redshift e o Google Cloud SQL.



Entregar continuamente software de alta qualidade para produção de forma rápida e confiável requer coordenação de muitas etapas automatizadas. O **Go CD** (go.cd) é uma ferramenta de código aberto desenvolvida (pela ThoughtWorks) para lidar exatamente com esse cenário; com o conceito de pipelines de implantação (deployment pipelines) em seu núcleo, lida com fluxos de trabalho complexos ao longo de muitos nós e permite a promoção de artefatos entre ambientes de forma transparente e rastreável. Embora seja possível criar pipelines de implantação em cima de ferramentas de integração contínua, os nossos times veem o benefício de uma ferramenta construída especificamente para esse propósito.

Boot2docker é uma distribuição leve do Linux que roda Docker, projetada como uma MV para OSX e Windows. É uma ótima maneira de começar a experimentar com Docker. Para os times que usam micro-serviços, esta também pode ser uma forma eficaz de executar vários serviços em uma máquina local para fins de desenvolvimento e testes, onde um grande número de MVs Vagrant pode chegar a consumir muitos recursos.

Apesar da ideia de gerenciamento de dependências não ser nova e já ser considerada uma prática fundamental de desenvolvimento, ela não é amplamente adotada pela comunidade PHP. **Composer** (getcomposer.org) é uma ferramenta para gerenciamento de dependências em PHP. Ela é fortemente influenciada pelas ferramentas de outros stacks, tais como npm de Node e Bundler de Ruby.

Cursive (cursiveclojure.com) é uma IDE Clojure que funciona como um plugin para Intelij, e já durante os

ADOTE
51. Flyway
52. Go CD

EXPERIMENTE
53. Appium
54. Boot2docker
55. Composer
56. Cursive
57. Docker
58. Foreman
59. GenyMotion
60. Gitlab
61. Grunt.js
62. IndexedDB
63. Packer
64. Pact & Pacto
65. Papertrail
66. Postman
67. SnapCI
68. Snowplow Analytics & Piwik
69. Swagger
70. Xamarin

AVALIE
71. Blackbox
72. Consul
73. DC.js
74. Gorilla REPL
75. Gulp
76. Leaflet.js
77. Mountebank
78. Packetbeat
79. Roslyn
80. Spark
81. Terraform

EVITE
82. Citrix para desenvolvimento

FERRAMENTAS *continuação*

primeiros acessos percebemos grande utilidade ao trabalhar com bases grandes de código Clojure. Cursive fornece ótimo suporte para renomeação e navegação, se mostrou estável e confiável, e é ótimo para ambientes com várias linguagens JVM. Para organizações adotando Clojure, Cursive tem ajudado a diminuir a barreira de entrada para seus desenvolvedores.

Gitlab é uma plataforma de hospedagem local de repositórios Git que dá a times de desenvolvimento de software proprietário o fluxo de trabalho familiar e ubíquo que os serviços hospedados de controle de versão como Github e BitBucket fornecem aos desenvolvedores de software livre. Embora esteja disponível um software de edição da comunidade livre, a opção comercial para empresas fornece suporte e profunda integração com servidores LDAP.

Com o aumento da viabilidade e da disseminação de aplicações web em página única e offline, cresce a demanda pelo armazenamento de dados no navegador. LocalStorage é muito simples de utilizar e bem suportado pelos navegadores. Para os casos de uso mais complexos existe o **IndexedDB**. Ao mesmo tempo que ele pode ser uma boa solução, recomendamos seu uso apenas em casos de absoluta necessidade. Isso se deve ao aumento da complexidade e a uma API um pouco confusa. Também tivemos experiências positivas com o framework LocalForage (github.com/mozilla/localForage), que provê uma camada de abstração sobre as várias soluções de persistência.

Postman (www.getpostman.com/features) é uma extensão para o Chrome que atua como um cliente REST em seu navegador, permitindo a criação de requisições e a análise de respostas, uma ferramenta útil no desenvolvimento de API ou ainda na implementação de um cliente para utilizar uma API. Ele oferece um conjunto de extensões que permitem usá-lo como um completo executor de testes também, ainda que não recomendemos o estilo de testes gravar e repetir que ele promove.

Blackbox (github.com/StackExchange/blackbox) é uma ferramenta simples para criptografar arquivos específicos presentes em seu repositório de código-fonte. É particularmente útil se você precisa armazenar senhas ou chaves privadas. Blackbox funciona com Git, Mercurial e Subversion e usa GPG para a criptografia com cada usuário tendo sua própria chave, tornando mais fácil revogar o acesso em um nível granular.

Já recomendamos D3.js no passado e neste Radar queremos estender nossa recomendação para **DC.js**, uma biblioteca gráfica baseada em D3 para explorar grandes conjuntos de dados multidimensionais. Ela compartilha com D3 a facilidade de criação de belos gráficos interativos. É diferente no balanceamento da flexibilidade para criar quase qualquer tipo de visualização de dados para um modelo de programação mais simples, criando gráficos comuns.

GorillaREPL (gorilla-repl.org) é uma ferramenta para a criação de documentos bem renderizados consistindo de texto, código Clojure e plotagem. Em alguns aspectos semelhante aos blocos iPython, GorillaREPL pode ser útil especialmente para analistas de dados ou tutoriais de código. Mas, além disso, GorillaREPL é divertido! É uma forma criativa de demonstrar o poder de abstrações simples do Clojure sobre valores imutáveis.

À medida que sistemas distribuídos se tornam mais complexos, pode ser útil dispor de ferramentas que ajudem a entender como o sistema está se comportando em produção. O **Packetbeat** (packetbeat.com) é uma ferramenta de código aberto que utiliza agentes para monitorar o tráfego entre nós de um sistema distribuído, permitindo que você observe padrões de tráfego, taxas de erro e outras informações úteis. Ele requer Elasticsearch (www.elasticsearch.org/overview/elasticsearch) e Kibana (www.elasticsearch.org/overview/kibana) para funcionar, mas se você já está usando essas ferramentas para agregação de logs, o Packetbeat pode ser facilmente adicionado para fornecer mais detalhes que ajudarão a compreender seu sistema em produção.

Com **Terraform** a infraestrutura de nuvem pode ser gerenciada através de definições declarativas. A configuração dos servidores instanciados pelo Terraform geralmente é deixada para ferramentas como Puppet, Chef, ou Ansible. Gostamos do Terraform porque a sintaxe de seus arquivos é bastante legível e porque ele suporta vários provedores de nuvem, ao mesmo tempo em que não tenta criar nenhuma abstração artificial

FERRAMENTAS *continuação*

através desses provedores. Neste momento, Terraform é novo e nem tudo está implementado ainda. Descobrimos também que o seu gerenciamento de estados é frágil, muitas vezes necessitando de trabalho manual, difícil de resolver.

Por razões de segurança e conformidade, os times offshore são algumas vezes instruídos a usar **Citrix para desenvolvimento** conectando-se a um desktop virtual onshore. Embora seja uma boa ferramenta para alguns casos de uso, Citrix oferece uma experiência extremamente pobre de desenvolvimento remoto e muitas vezes incapacita um time offshore. Existem várias soluções melhores, tais como desktop remoto NoMachine ou Cloud9 IDE, que podem proporcionar uma experiência mais viável. Uma solução ainda melhor é lidar com as preocupações de segurança e conformidades subjacentes: dado que você está confiando ao time remoto acesso para lidar com seu código fonte e acessar seu repositório, você deveria buscar um ponto onde você também confie a eles acesso local ao código fonte. Eles serão muito mais produtivos!

LINGUAGENS & FRAMEWORKS

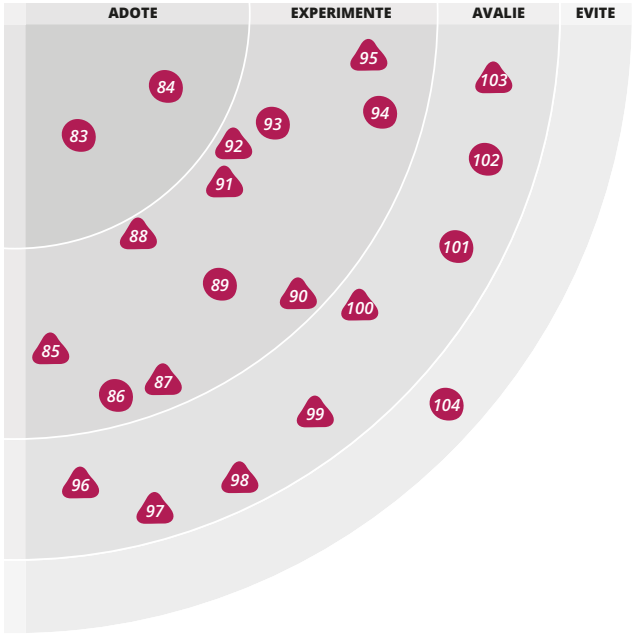
A importância de monitores grandes e visíveis em áreas compartilhadas por times já recebeu muita atenção anteriormente, e certamente valorizamos a abordagem de ajudar todos a verem e entenderem informações fundamentais sobre como o software e o time estão. **Dashing** (dashing.io) é um sistema de dashboard baseado em ruby que temos usado já há alguns anos para criar displays definidos e visíveis, otimizados para monitores maiores. Ele é super hackeável, permitindo que você puxe informações de uma variedade de fontes, desde sistemas de build, ferramentas de rastreamento de ticket e de estórias, ou sistemas de monitoramento de produção.

Já usamos em vários projetos o **framework Django Rest** (www.django-rest-framework.org), um framework flexível e personalizável que facilita a construção de APIs web. Ele oferece a capacidade de criar APIs RESTful em Python com Django, expondo pontos de conexão com a API que podem ser acessados por um front-end de consumidor. O framework Django Rest disponibiliza uma interface navegável para a API, permitindo aos desenvolvedores visualizar os dados que são transferidos através da API e exemplos de respostas que serão recebidas pela aplicação do consumidor. Ele fornece uma série de esquemas de autenticação prontos para serem usados e também permite a implementação de esquemas personalizados.

Framework Ionic (ionicframework.com) é um framework de front-end de código aberto que oferece uma biblioteca HTML otimizada para dispositivos móveis, componentes CSS e JavaScript e ferramentas para criação de aplicações altamente interativas. Ele é construído com SASS e otimizado para AngularJS. Vimos sucesso em vários de nossos projetos que empregam esse framework, com a sua facilidade de instalação e teste. Recomendamos investigar esse framework se você está obcecado por desempenho e em busca de um framework de front-end perfeitamente integrado.

Nashorn é um novo engine de JavaScript para Java que foi lançado com o Java 8. Se exatamente o mesmo código deve ser executado no navegador e no servidor, o que frequentemente é o caso para validação e lógica de migração de dados, é a ferramenta mais usada no mundo Java, e esse é o caso apesar de algumas partes não muito boas. Porém, não estamos convencidos de que seja uma boa ideia usar o Nashorn para hospedar aplicações inteiras através do suporte do Node ou do projeto Avatar.

Retrofit (square.github.io/retrofit) oferece uma maneira confiável de construir clientes HTTP em projetos Android através da conversão de uma API REST em uma interface Java. Retrofit integra com okhttp e oferece aos desenvolvedores a possibilidade de fornecer tratamento de erro personalizado para requisições. Ele faz o parsing de JSON automaticamente usando GSON e tem muito apoio da comunidade.



ADOTE
83. A Linguagem Go
84. Java 8

EXPERIMENTE
85. AngularJS
86. Core Async
87. Dashing
88. Django Rest
89. HAL
90. Framework Ionic
91. Nashorn
92. Om
93. Q & Bluebird
94. R como plataforma de computação
95. Retrofit

AVALIE
96. Flight.js
97. Biblioteca Haskell Hadoop
98. Lotus
99. React.js
100. Reagent
101. Rust
102. Spring Boot
103. Swift

EVITE
104. JSF

LINGUAGENS & FRAMEWORKS *continuação*

Em meio a tantos frameworks JavaScript, gostaríamos de destacar o **Flight.js** como uma alternativa a ser considerada. Flight é extremamente leve e se sai bem sem muitos truques ao adicionar comportamentos a nós do DOM. Por ser orientado a eventos e baseado em componentes, induz naturalmente códigos desacoplados. Isso facilita muito testar componentes individualmente. Entretanto, é preciso tomar cuidado com a interação dos componentes entre si, pois há pouco suporte para testes e um grande risco de se perder em uma bagunça de eventos. Gostamos do fato de que utiliza mixins funcionais para comportamento, isto é, composição ao invés de herança.

Apesar de termos muitos fãs do **Haskell** entre os ThoughtWorkers devotados a linguagens, raramente o vemos nos tipos de projetos que trabalhamos—até recentemente. Vários projetos de código aberto agora unem o map/reduce jobs do **Hadoop** à sintaxe do Haskell, que atrai muitos desenvolvedores e/ou cientistas de dados.

Não sabemos quem batizou **Lotus** (lotusrb.org), mas supomos que tenha sido alguém jovem demais para ter trabalhado com um determinado produto de colaboração em escritório. Lotus é um novo framework MVC escrito em Ruby, baseado em Rack, que pode ser implantado modularmente; assim você é livre para usar apenas as partes do framework que você precisa. É uma alternativa moderna para o framework monolítico Ruby-on-Rails (que completou 10 anos, recentemente). Lotus tem o potencial de fazer o desenvolvimento MVC full-stack em Ruby tão fácil quanto contar até três.

Um benefício da avalanche contínua de frameworks front-end JavaScript é que ocasionalmente uma ideia nova surge que nos faz pensar. **React.js** é um framework de IU e Visualização em que funções JavaScript geram HTML em um fluxo de dados reativo. Embora sejamos cautelosos com a mistura de código e marcação, isso resulta em componentes de interface que são bem encapsulados e combináveis. React.js está recebendo muita atenção dos desenvolvedores e será beneficiado pela disponibilização de mais ferramentas e exemplos.

Reagent (holmsand.github.io/reagent) tem surgido como uma alternativa leve e minimalista ao Om para empacotar o React.js no Clojurescript. Enquanto o Om fornece um compreensivo framework front-end Clojure idiomático, Reagent pega a vantagem da expressividade do Clojure para focar em componentes simples e uma DSL legível para escrever HTML. Representando HTML em dados Clojure, Reagent mantém o desempenho e a compreensibilidade do React.js sem incorporar marcação externa ao código.

Swift (www.apple.com/swift), a nova linguagem de programação da Apple, contém muitas melhorias em relação à perene Objective-C, incluindo ênfase em programação funcional e uma sintaxe moderna. Sob vários pontos de vista, trata-se de um upgrade se você está programando na plataforma Apple.

A ThoughtWorks é uma empresa de software e comunidade de indivíduos apaixonados e guiados por princípios, especialistas em consultoria, entrega e produtos em software. Pensamos disruptivamente para entregar tecnologias que lidem com os desafios mais ambiciosos de nossos clientes, ao mesmo tempo em que buscamos revolucionar a indústria de TI e criar mudanças sociais positivas. Criamos ferramentas pioneiras para times de software que querem ser ótimos. Nossos produtos ajudam organizações a melhorar continuamente e entregar software de qualidade para suprir suas necessidades mais fundamentais. Fundada a mais de 20 anos, a ThoughtWorks cresceu a partir de um pequeno grupo em Chicago para uma empresa de mais de 3.000 pessoas, espalhadas por 30 escritórios em 12 países: Austrália, Brasil, Canadá, China, Equador, Alemanha, Índia, Singapura, África do Sul, Uganda, Reino Unido e Estados Unidos.

ThoughtWorks®