ThoughtWorks®

TECHNOLOGY RADAR

Nossas ideias sobre as tecnologias e tendências que moldam o futuro

MAIO 2015

thoughtworks.com/radar

O QUE HÁ DE NOVO?

Aqui estão as tendências em destaque nesta edição:

INOVAÇÃO EM ARQUITETURA

Organizações aceitaram que a "nuvem" é a plataforma *de-facto* do futuro, os benefícios e flexibilidade que ela traz conduziram a um renascimento em arquitetura de software. A infraestrutura disponível permitiu o surgimento da primeira arquitetura "nativa de nuvem": micro-serviços. Entrega contínua, técnica que vem transformando radicalmente a forma como negócios baseados em tecnologia evoluem, amplifica o impacto da nuvem como uma arquitetura. Esperamos que a inovação em arquitetura continue, com tendências como o uso de serviços de execução e gerenciamento de contêineres, assim como redes definidas por software (software-defined networking ou SDN), fornecendo mais opções técnicas e capacidade.

UMA NOVA ONDA DE ABERTURA NA MICROSOFT

Embora a Microsoft já tenha se envolvido em código aberto no passado, como em sua plataforma de hospedagem CodePlex, os ativos estratégicos da empresa continuaram a ser proprietários, com seus segredos muito bem guardados. No entanto, a Microsoft agora parece estar adotando uma nova estratégia de abertura, liberando grandes porções da plataforma .NET e runtime como projetos de código aberto no GitHub. Estamos otimistas que isso possa sedimentar o caminho de Linux como uma plataforma de hospedagem para .NET, permitindo que a linguagem C# possa competir ao lado das inúmeras linguagens atuais baseadas em JVM.

ESFORÇOS EM SEGURANÇA CONTINUAM NAS EMPRESAS

Apesar da crescente atenção à segurança e privacidade, a indústria não tem feito muito progresso desde o último Radar e continuamos a destacar a questão. Desenvolvedores estão respondendo com um aumento de ferramental e infraestrutura de segurança, colocando ferramentas de teste automatizado, como o Zed Attack Proxy, em pipelines de implantação. Tais ferramentas são, naturalmente, apenas parte de uma abordagem holística para segurança. Acreditamos que todas as organizações precisam melhorar seu desempenho nessa área.

CONTRIBUIDORES

O TAB da ThoughtWorks é 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

Jeff Norris

Jonny LeRoy

Mike Mason

Neal Ford

Rachel Laycock

Sam Newman

Scott Shaw

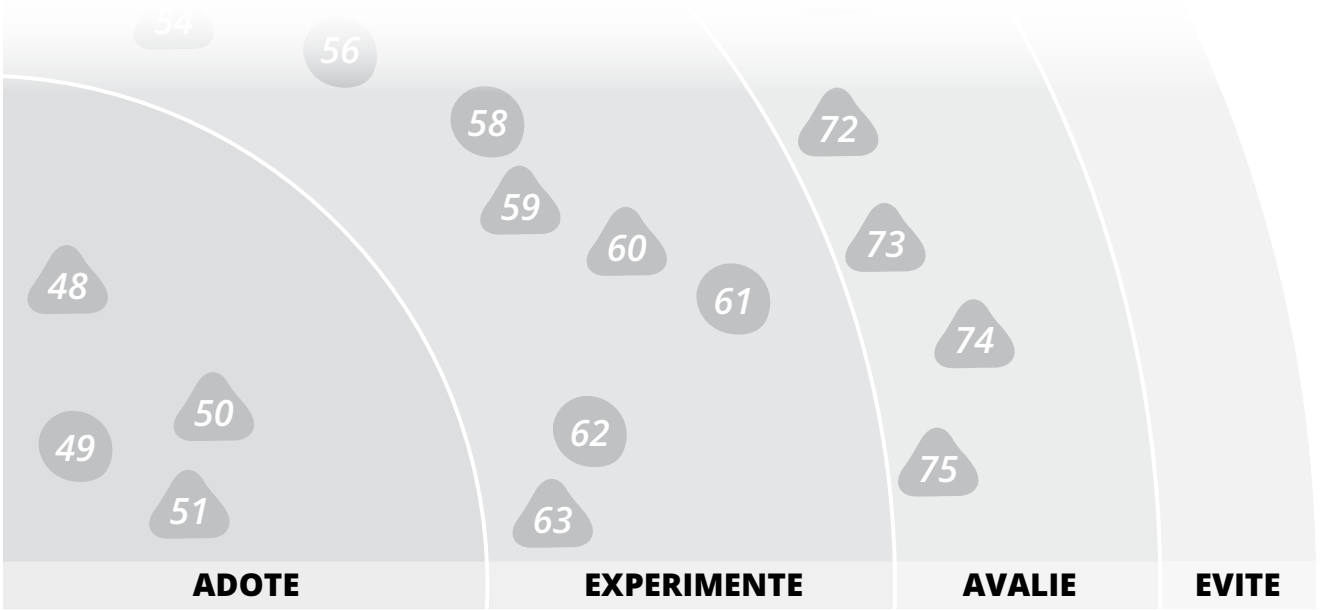
Srihari Srinivasan

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 Advisoy 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 arcabouços. 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 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. 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. Testes de contrato guiados pelo consumidor novo
2. Foco no tempo médio de recuperação
3. Diagrama automatizado de infraestrutura novo
4. Logs estruturados

EXPERIMENTE
5. Implantações Canário
6. Datensparsamkeit
7. Sincronização de armanezamento local
8. NoPSD
9. Aplicações web “offline first” novo
10. Produtos acima de projetos novo
11. Modelagem de ameaças novo

AVALIE
12. Armazenamento de dados append-only
13. Blockchain além do bitcoin
14. Data Lake corporativo
15. Flux novo
16. SGC baseado em Git/Git além do código novo
17. Ambientes Phoenix novo
18. Arquiteturas reativas novo

EVITE
19. Branches de vida longa com Gitflow
20. Inveja de micro-serviços
21. Programar em sua ferramenta de integração contínua/entrega contínua
22. SAFe (TM)
23. Sanduíche de segurança
24. DevOps como um time

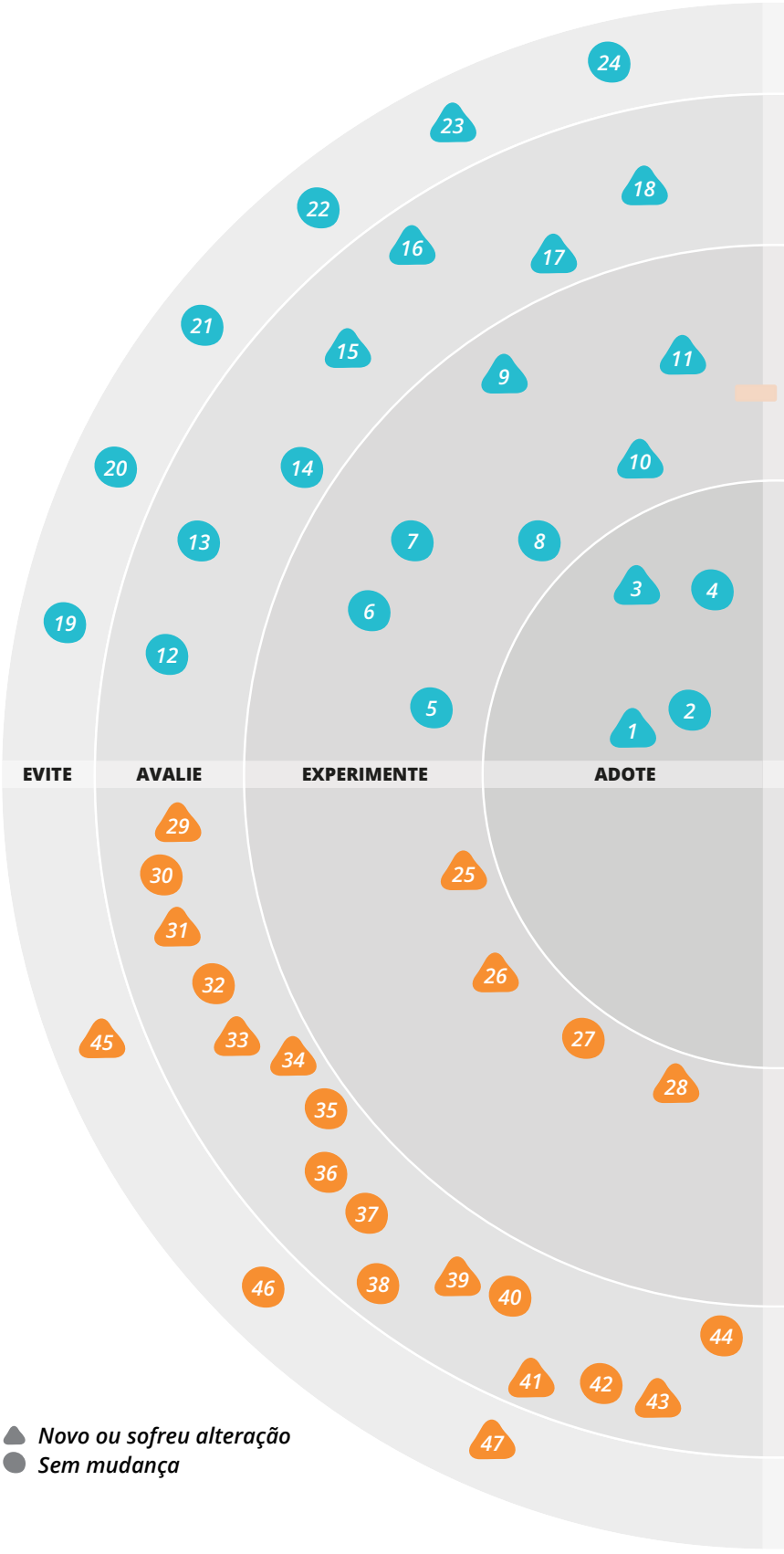
PLATAFORMAS

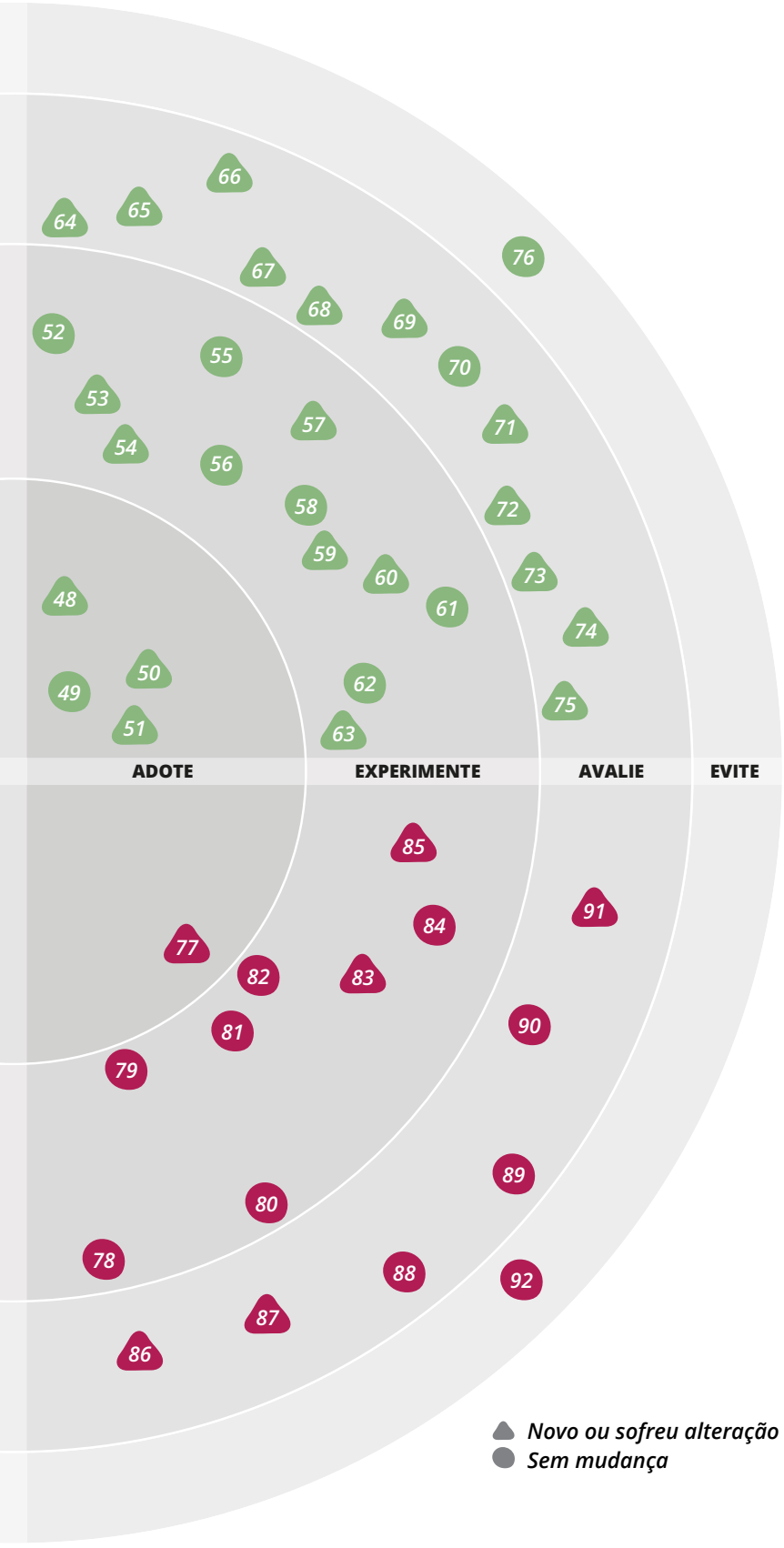
ADOTE

EXPERIMENTE
25. Apache Spark
26. Cloudera Impala novo
27. DigitalOcean
28. TOTP Autenticação de duas etapas

AVALIE
29. Apache Kylin novo
30. Apache Mesos
31. CoreCLR e CoreFX novo
32. CoreOS
33. Deis novo
34. H2O novo
35. Jackrabbit Oak
36. Módulos de segurança do Linux
37. MariaDB
38. Netflix OSS full stack
39. OpenAM
40. Redes definidas por software (SDN)
41. Spark photon / electron novo
42. Text it as a service / RapidPro.io
43. Banco de dados de séries temporais novo
44. U2F

EVITE
45. Servidores de aplicação novo
46. OSGi
47. SPDY novo





O RADAR

FERRAMENTAS

ADOTE
48. Composer
49. Go CD
50. Mountebank
51. Postman

EXPERIMENTE
52. Boot2docker
53. Brighter novo
54. Consul
55. Cursive
56. Gitlab
57. Hamms novo
58. IndexedDB
59. Polly novo
60. REST-assured novo
61. Swagger
62. Xamarin
63. ZAP novo

AVALIE
64. Apache Kafka novo
65. Blackbox
66. Bokeh/Vega novo
67. Gor novo
68. NaCl novo
69. Origami novo
70. Packetbeat
71. pdfmake novo
72. PlantUML novo
73. Prometheus novo
74. Quick novo
75. Security monkey novo

EVITE
76. Citrix para desenvolvimento

LINGUAGENS & ARCABOUÇOS

ADOTE
77. Nancy

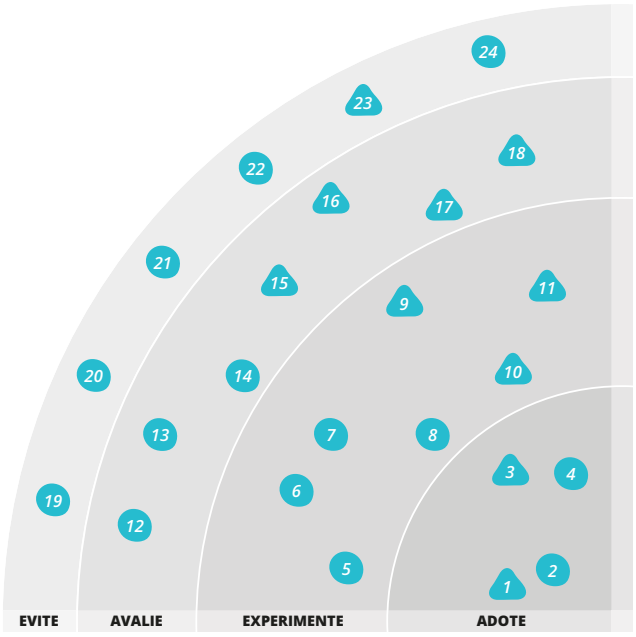
EXPERIMENTE
78. Dashing
79. Django REST
80. Arcabouço Ionic
81. Nashorn
82. Om
83. React.js
84. Retrofit
85. Spring Boot

AVALIE
86. Ember.js novo
87. Flight.js
88. Biblioteca Haskell Hadoop
89. Lotus
90. Reagent
91. Swift

EVITE
92. JSF

TÉCNICAS

Quando dois serviços independentes colaboram, mudanças na API do serviço provedor podem causar falhas em todos os consumidores. Serviços consumidores normalmente não podem ser testados utilizando uma versão 'live' dos serviços provedores, visto que tais testes são lentos e fáceis de quebrar (martinfowler.com/articles/nonDeterminism.html#RemoteServices). Logo, é melhor utilizar dublês de teste (martinfowler.com/bliki/TestDouble.html), mas com o risco dos dublês ficarem dessincronizados em relação ao serviço provedor real. Times desenvolvedores de serviços consumidores podem se proteger dessas falhas usando Testes de Integração de Contrato (martinfowler.com/bliki/IntegrationContractTest.html), testes que comparam a resposta real de um serviço com valores de teste. Mesmo que os testes de contrato sejam valiosos, eles são ainda mais úteis quando os serviços consumidores provêem esses testes ao provedor. Ele então pode rodar todos os testes de contrato do consumidor para determinar se as mudanças feitas causam problemas, adotando contratos guiados pelo consumidor (martinfowler.com/articles/consumerDrivenContracts.html). Os **testes de contratos guiados pelo consumidor** são parte essencial de um portfólio maduro de testes de micro-serviços (martinfowler.com/articles/microservice-testing/).



Quando precisamos de um diagrama que descreva a infraestrutura ou a arquitetura física atuais, geralmente usamos nossas ferramentas favoritas de diagramação técnica. Se você está usando tecnologias de virtualização ou cloud, isso não faz mais sentido, pois podemos utilizar as APIs disponibilizadas para examinar a infraestrutura atual e gerar um **diagrama automatizado de infraestrutura** com ferramentas simples como GraphViz (graphviz.org) ou produzindo um SVG.

Aplicações web com o conceito **"offline first"** possibilitam o desenho de funcionalidades para acesso offline através de mecanismos de caching e atualização. A implementação requer o uso de uma flag no DOM para verificar se o dispositivo de acesso está offline ou online, acessando o armazenamento local quando offline e sincronizando os dados quando online. Os principais navegadores já dão suporte a um modo offline, com informação local acessível através da definição de um atributo no manifesto do HTML, o que inicia o processo de download e armazenamento de recursos como HTML, CSS, JavaScript, imagens etc. Existem algumas ferramentas que ajudam a simplificar a implementação do conceito de offline first, como Hoodie (hood.ie) e o CouchDB (couchdb.apache.org), que também oferece a capacidade de interagir com uma aplicação instalada localmente usando um banco de dados local.

A maioria do esforço em desenvolvimento de software é feito usando o modelo mental de projeto, algo planejado, executado e entregue dentro de um tempo estipulado. O desenvolvimento ágil desafiou esse modelo, substituindo uma grande quantidade inicial de requisitos por um processo contínuo de descoberta que ocorre simultaneamente ao desenvolvimento. Técnicas lean startup, como teste A/B de requisitos observados (martinfowler.com/bliki/ObservedRequirement.html),

ADOTE
1. Testes de contrato guiados pelo consumidor
2. Foco no tempo médio de recuperação
3. diagrama automatizado de infraestrutura
4. Logs estruturados

EXPERIMENTE
5. Implantações Canário
6. Datensparsamkeit
7. Sincronização de armanezamento local
8. NoPSD
9. Aplicações web "offline first"
10. Produtos acima de projetos
11. Modelagem de ameaças

AVALIE
12. Armazenamento de dados append-only
13. Blockchain além do bitcoin
14. Data Lake corporativo
15. Flux
16. SGC baseado em Git/Git além do código
17. Ambientes Phoenix
18. Arquiteturas reativas

EVITE
19. Branches de vida longa com Gitflow
20. Inveja de micro-serviços
21. Programar em sua ferramenta de integração contínua/entrega contínua
22. SAFe (TM)
23. Sanduíche de segurança
24. DevOps como um time

TÉCNICAS *continuação*

corroboram com essa mentalidade. Consideramos que a maioria dos esforços de software devem seguir o exemplo de Lean Enterprise (info.thoughtworks.com/lean-enterprise-book.html) e levar em conta a construção de produtos que dão suporte aos processos fundamentais do negócio. Tais produtos não geram uma entrega final, mas um processo contínuo para explorar a melhor forma de apoiar e otimizar os processos de negócios, enquanto o negócio tiver valor. Por esses motivos, encorajamos empresas a pensarem em termos de **produtos acima de projetos**.

Atualmente, a maioria dos times de desenvolvimento já está ciente da importância de se escrever código seguro e de lidar com os dados de seus usuários de maneira responsável. Entretanto, esses times ainda se deparam com uma alta curva de aprendizado e um vasto número de potenciais ameaças, desde o crime organizado e espionagem governamental, até adolescentes que atacam sistemas só por diversão. **Modelagem de ameaças** (owasp.org/index.php/Category:Thread_modeling) é um conjunto de técnicas, principalmente em um aspecto defensivo, que ajudam a entender e a classificar possíveis ameaças. Quando transformadas em "histórias de usuários malignos", essas técnicas dão ao time uma estratégia mais gerenciável e efetiva de tornar seus sistemas mais seguros.

Flux (facebook.github.io/flux) é uma arquitetura de aplicação que o Facebook tem adotado no desenvolvimento de suas aplicações web. Geralmente mencionado em conjunto com react.js, Flux baseia-se em um fluxo de dados unidirecional na pipeline de renderização, acionado por usuários ou outros eventos externos, modificando bases de dados. Há muito tempo vimos surgir alguma alternativa à venerada arquitetura modelo-visão e o Flux abraça o cenário moderno da web, em que aplicações JavaScript client-side comunicam-se com múltiplos serviços back-end.

Hoje em dia, a maioria dos desenvolvedores de software está habituada a utilizar o git para controle de versão de código e colaboração. Mas git pode ser usado como um mecanismo base para outras situações em que um grupo de pessoas precisa colaborar em documentos textuais (que possam ser facilmente unificados). Temos visto um número crescente de projetos usando **Git** (git-scm.com) como base para **SGCs** (Sistema de Gerenciamento de Conteúdo) leves, com formatos de edição baseados em texto. Git tem funcionalidades poderosas para rastrear mudanças e explorar alternativas, com um modelo de armazenamento distribuído que é rápido e tolerante a problemas de rede. O maior problema com uma adoção mais abrangente é que git não é tão fácil de aprender para não-programadores, mas nós esperamos ver mais ferramentas construídas sobre a base do git. Essas ferramentas simplificam o fluxo de trabalho para públicos específicos, como autores de conteúdo. Também gostaríamos de ter mais ferramentas para dar suporte à comparação e unificação de documentos não-textuais.

A ideia de servidores Phoenix (martinfowler.com/bliki/PhoenixServer.html) já está bem estabelecida e traz vários benefícios quando aplicada aos tipos certos de problemas. Mas o que dizer sobre o ambiente no qual fazemos implantação desses servidores? O conceito de **ambientes Phoenix** pode ajudar. Podemos fazer o uso de automação para criar ambientes inteiros, incluindo configurações de rede, balanceamento de carga e portas do firewall, por exemplo através do **CloudFormation** (AWS). Podemos comprovar que o processo funciona destruindo esses ambientes e os recriando a partir do zero regularmente. Os ambientes Phoenix podem dar suporte ao provisionamento de novos ambientes de testes, desenvolvimento, testes de aceitação, entre outros. Eles também permitem simplificar a criação de ambientes de recuperação de desastres. Assim como acontece com servidores Phoenix, esse padrão nem sempre é aplicável e nós precisamos pensar cuidadosamente sobre coisas como estado e dependências. Uma abordagem é adotar o conceito de implantações azul-verde (martinfowler.com/bliki/BlueGreenDeployment.html) para todo o ambiente nos casos em que haja a necessidade de sua reconfiguração.

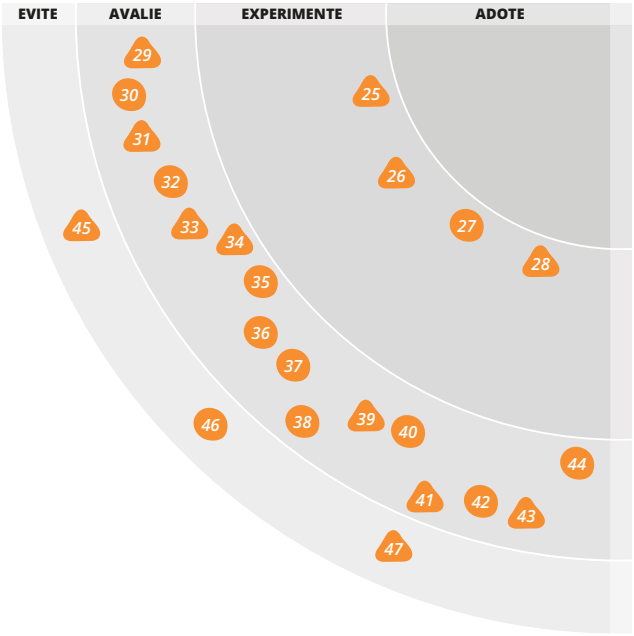
As técnicas de programação funcional reativa têm experimentado um aumento progressivo de uso nos últimos anos e estamos vendo interesse em estender esse conceito a arquiteturas de sistemas distribuídos. Inspirado em parte pelo Manifesto Reativo (reactivemanifesto.org), essas **arquiteturas reativas** são baseadas em um fluxo assíncrono e unidirecional de eventos imutáveis através uma rede de processos independentes (talvez implementados como micro-serviços). Na configuração correta, esses sistemas são escaláveis, resilientes e diminuem o acoplamento entre unidades individuais de processamento. No entanto, arquiteturas totalmente baseadas na passagem de mensagem assíncrona introduzem complexidade e frequentemente dependem de arcabouços

proprietários. Indicamos avaliar as necessidades de desempenho e escalabilidade do seu sistema antes de se comprometer com o uso desse estilo arquitetural como padrão.

Abordagens tradicionais de segurança baseiam-se em uma especificação inicial seguida por uma validação final. Essa abordagem de **"sanduíche de segurança"** é difícil de integrar em times ágeis, pois grande parte do design acontece ao longo do processo e não usufrui das oportunidades de automatização trazidas pela entrega contínua. As organizações devem avaliar como podem inserir práticas de segurança em todo o ciclo de desenvolvimento ágil. Isso inclui: avaliar o nível certo de modelagem de ameaças logo no começo; classificar as preocupações de segurança a partir de suas próprias histórias, critérios de aceitação, ou requisitos não-funcionais; incluir testes automatizados de segurança estáticos e dinâmicos na sua integração contínua; e como incluir testes mais profundos nas entregas, como testes de penetração, dentro de um modelo de entrega contínua. Da mesma forma que DevOps mudou como grupos historicamente adversários podem trabalhar juntos, o mesmo está acontecendo para profissionais de segurança e desenvolvimento. E apesar da nossa antipatia pelo modelo sanduíche de segurança, é bem melhor que não considerar segurança alguma, o que infelizmente ainda é uma situação comum.

PLATAFORMAS

O **Apache Spark** (spark.apache.org) vem conquistando espaço como um motor para processamento de dados em larga escala. É escrito em Scala e seu uso é apropriado em aplicações que reutilizam um conjunto de dados de trabalho em diferentes operações paralelas. O Spark foi desenvolvido para funcionar como um cluster independente ou como parte de um cluster Hadoop YARN. Ele acessa dados de fontes como HDFS, Cassandra, S3, etc. O Spark também oferece várias operações de alto nível para facilitar o desenvolvimento de aplicações com dados paralelos. Como uma plataforma de processamento de dados genérica, o Spark possibilita o desenvolvimento de várias ferramentas de alto nível como um SQL interativo (Spark SQL), 'streaming' em tempo real (Spark Streaming), bibliotecas de aprendizagem de máquina (MLib), R-on-Spark etc. Há algum tempo a comunidade do Hadoop tenta implementar capacidade de SQL interativo e de baixa latência na plataforma Hadoop (também conhecido como SQL-on-Hadoop). Esse esforço resultou em alguns sistemas de código aberto como o Claudera Impala, Apache Drill, Presto (do Facebook) e outros, todos sendo ativamente desenvolvidos ao longo de 2014. Acreditamos que essa tendência no SQL-on-Hadoop sugere uma mudança importante, pois altera a proposta original do Hadoop de ser uma tecnologia orientada a batch que complementa bancos de dados para uma tecnologia que possa competir com eles.



Cloudera Impala (cloudera.com/content/cloudera/en/products-and-services/cdh/impala.html) foi uma das primeiras plataformas SQL-on-Hadoop. É um mecanismo de consulta distribuído, massivamente paralelo e baseado em C++. O componente central dessa plataforma é o "daemon Impala", que coordena a execução da consulta SQL através de um ou mais nós do cluster Impala. Impala é feito para ler dados de arquivos armazenados em HDFS em todos os formatos de arquivo comuns. Ele aproveita o catálogo de metadados do Hive para compartilhar bancos de dados e tabelas entre as duas plataformas de banco de dados. Impala traz um "shell" e drivers para aplicações JDBC e ODBC.

Senhas continuam sendo um mecanismo pobre para autenticar usuários e percebemos recentemente que empresas como a Yahoo! migraram para uma solução "sem senhas" - um código que aparece apenas uma vez é enviado para o seu celular sempre que você precisa se logar de um novo navegador. Se você ainda utiliza senhas, recomendamos que use a **autenticação em duas etapas**, pois melhora a segurança significativamente. Senha temporária de acesso único (**TOTP**) (en.wikipedia.org/wiki/Time-based_One-time_Password_Algorithm) é o algoritmo padrão nesse âmbito, com aplicativos grátis de autenticação para smartphones do Google (play.google.

ADOTE

EXPERIMENTE
25. Apache Spark
26. Cloudera Impala
27. DigitalOcean
28. TOTP Autenticação de duas etapas

AVALIE
29. Apache Kylin
30. Apache Mesos
31. CoreCLR e CoreFX
32. CoreOS
33. Deis
34. H2O
35. Jackrabbit Oak
36. Módulos de segurança do Linux
37. MariaDB
38. Netflix OSS full stack
39. OpenAM
40. Redes definidas por software (SDN)
41. Spark photon / electron
42. Text it as a service / RapidPro.io
43. Banco de dados de séries temporais
44. U2F

EVITE
45. Servidores de aplicação
46. OSGi
47. SPDY

PLATAFORMAS *continuação*

com/store/apps/details?id=com.google.android.apps.authenticator2) e da Microsoft (windowsphone.com/en-us/store/app/authenticator/e7994dbc-2336-4950-91ba-ca22d653759b).

Apache Kylin (kylin.io) é uma solução analítica em código aberto da eBay Inc. que permite análise multidimensional baseada em SQL para conjuntos de dados muito grandes. Kylin é projetada para ser uma solução híbrida OLAP (HOLAP), baseada em Hadoop e que eventualmente vai dar suporte a dois estilos de análise multidimensional: MOLAP e ROLA. Com Kylin pode-se definir cubos por meio de um Cube Designer e inicia-se um processo offline que cria os cubos. O processo offline faz uma etapa "pre-join" para juntar tabelas de fatos e de dimensão em uma estrutura plana. Depois há uma fase de pré-agregação quando cuboids individuais são criados por tarefas Map Reduce. Os resultados são salvos em arquivos de sequência HDFS e posteriormente carregados no HBase. As requisições de dados podem ser originadas a partir de consultas SQL submetidas por um cliente SQL. O mecanismo de consulta (baseado no **Apache Calcite**) determina se o conjunto de dados alvo existe no HBase. Se sim, o mecanismo acessa diretamente os dados procurados no HBase e retorna o resultado com latência de sub-segundo. Se não, o mecanismo roteia as consultas para o **Hive** (ou qualquer outra solução de SQL em Hadoop que esteja disponível no cluster).

CoreCLR (github.com/dotnet/coreclr) e **CoreFX** (github.com/dotnet/corefx) são a plataforma e o arcabouço centrais para .NET. Embora não sejam novos, foram recentemente liberados pela Microsoft como código aberto. Uma mudança substancial é que essas dependências são binariamente instaláveis, ou seja, não precisam ser previamente instaladas em uma máquina. Isso facilita implantações de lado a lado, permitindo que as aplicações possam usar versões de arcabouço diferentes sem conflito. Algo escrito em .NET é então um detalhe de implementação e pode-se instalar uma dependência .NET em qualquer ambiente. Do ponto de vista de dependência externa, uma ferramenta .NET não é diferente de algo escrito em C, tornando-a uma opção bem mais atraente para aplicações de propósito geral e utilitários. CoreFX também vem sendo desenvolvido como dependências NuGet individuais, para que aplicações possam puxar o que elas precisam, mantendo o resíduo para aplicações e bibliotecas .NET pequeno e deixando mais fácil substituir parte do arcabouço.

Heroku, com o seu modelo de aplicação de 12 fatores (12factor.net), mudou a maneira como pensamos sobre construção, implantação e hospedagem de aplicações web. **Deis** (deis.io) encapsula o modelo Heroku PaaS (Plataforma como um Serviço) em um arcabouço de código aberto que faz implantação em contêineres Docker hospedados em qualquer lugar. Deis ainda está evoluindo, mas para aplicações que se encaixam no modelo de 12 fatores existe um grande potencial de simplificar os processos de implantação e hospedagem em um ambiente de sua escolha. Deis é mais um exemplo do rico ecossistema de plataformas e ferramentas emergindo ao redor do Docker.

A análise preditiva vem sendo usada cada vez mais em produtos, muitas vezes diretamente em funcionalidades para o usuário final. **H2O** (docs.0xdata.com) é um pacote novo e interessante de código aberto (desenvolvido por uma startup) que faz análises preditivas acessíveis às equipes de projeto pela sua interface fácil de usar. Ao mesmo tempo, ele se integra com as ferramentas favoritas dos cientistas de dados, R e Python, assim como Hadoop e Spark. Oferece ótimo desempenho e, pela nossa experiência, fácil integração em tempo de execução, especialmente em plataformas baseadas na JVM.

Quando a Oracle parou o desenvolvimento do OpenSSO da Sun – uma plataforma de gerenciamento de acesso em código aberto – a ForgeRock o assumiu e o integrou à suíte deles, o Open Identity Suite. Atualmente chamado **OpenAM** (forgerock.com/products/open-identity-stack/openam), ele preenche uma lacuna de plataforma escalável de código aberto que dê suporte ao OpenID Connect e SAML 2.0. Porém, o longo histórico do OpenAM resultou em uma base de código espalhada, cuja documentação pode ser incompreensível. Espera-se que em breve surja uma alternativa enxuta com melhor suporte para implantação e provisionamento.

Spark (spark.io) é um conjunto de soluções completo para dispositivos conectados na nuvem. **Spark Photon** é um microcontrolador com um módulo wifi. **Spark Electron** é uma variação que se conecta com uma rede de celulares. Spark OS adiciona REST API para os dispositivos. Isso simplifica a entrada na Internet das Coisas e a construção dos seus próprios dispositvos conectados.

Um **banco de dados de séries temporais** (TSDB) é um sistema otimizado para lidar com dados de séries temporais. Ele permite que os usuários possam fazer operações tipo CRUD em várias séries de tempo organizadas como objetos do banco de dados. Também permite a possibilidade de fazer cálculos estatísticos nas séries como um todo. Apesar de TSDBs não serem uma tecnologia totalmente nova, vemos interesse renovado nesses bancos de dados, principalmente no ramo de aplicações de Internet das Coisas. Esse interesse é facilitado por várias plataformas comercias e de código aberto (como **OpenTSDB**, **InfluxDB**, **Druid**, **BlueFloodDB**, etc.) que têm surgido recentemente. Vale mencionar também que alguns desses sistemas usam outros bancos de dados distribuídos, como **Cassandra** e **HBase**, como mecanismo interno de armazenamento.

A popularidade do uso de contêineres, servidores Phoenix e entrega contínua tem nos distanciado da abordagem padrão de implantação de aplicações web. Tradicionalmente, construímos um artefato e o instalamos em um servidor de aplicação. O resultado dessa abordagem é um longo ciclo de feedback para mudanças, aumento no tempo dos builds e um considerável custo de gerenciamento desses servidores de aplicação em produção. Muitos deles são difíceis de automatizar também. A maioria dos nossos times são a favor de embutir um servidor HTTP nas suas aplicações web. Existem diversas opções disponíveis: Jetty, SimpleWeb, Webbit e Owin Self-Host, entre outras. Automação e implantação mais fáceis e redução na quantidade de infraestrutura a ser gerenciada nos levam a recomendar servidores embutidos em vez de **servidores de aplicação** em futuros projetos.

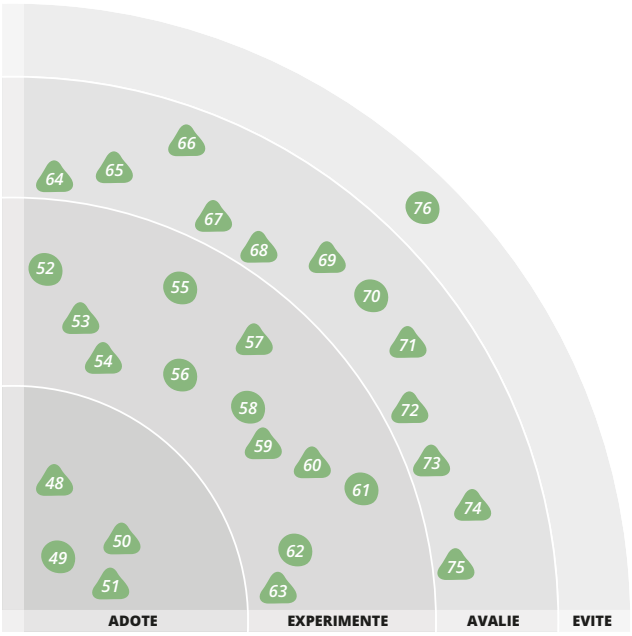
O protocolo **SPDY** (chromium.org/spdy/spdy-whitepaper) foi desenvolvido pelo Google, em 2009, como um experimento para fornecer um protocolo alternativo que superasse as limitações de performance existentes no HTTP/1.1. O novo padrão HTTP/2 inclui muitas das principais funcionalidades do SPDY e o Google anunciou que vai deixar de dar suporte ao SPDY no browser no começo de 2016. Se sua aplicação requer funcionalidades do SPDY, recomendamos que você passe a considerar a adoção do HTTP/2.

FERRAMENTAS

Embora a ideia de gerenciamento de dependências não seja nova, e considerada uma prática fundamental de desenvolvimento, ela não é adotada amplamente pela comunidade PHP. **Composer** (getcomposer.org) é uma ferramenta para gerenciamento de dependências em PHP. Ela é fortemente influenciada por ferramentas de outras tecnologias como o npm, do Node, e o Bundler, do Ruby. Agora estamos vendo uma grande adoção do Composer em projetos PHP e ele está razoavelmente maduro. Ainda pode ser necessário fazer alguns ajustes para usá-lo com bibliotecas internas, mas é possível adotá-lo para a maioria das bibliotecas externas.

Testar adequadamente componentes em um sistema corporativo é um trabalho de extrema relevância. Especialmente diante do crescimento na utilização de separação de componentes em serviços e automação de implantações, fatores críticos para o sucesso de micro-serviços, melhores ferramentas são necessárias neste espaço. O termo "Virtualização de Serviços" refere-se a ferramentas que podem emular componentes específicos nesse tipo de ambiente. Temos visto vários casos de sucesso com o uso da ferramenta **Mountebank** (www.mbtest.org), uma ferramenta leve para a criação de mocks e stubs HTTP, HTTPS, SMTP e TCP.

Postman (getpostman.com/features) é uma extensão do Google Chrome que funciona como um cliente REST no seu navegador, permitindo a criação de requisições e a inspeção de respostas. É uma ferramenta útil para o desenvolvimento de uma API ou para a implementação de um cliente para chamar uma API existente. Postman dá suporte aos tokens OAuth1 e OAuth2, permitindo adicioná-los onde for necessário. A resposta é disponibilizada como um JSON ou XML amigável. Com o Postman, é possível recuperar o histórico das requisições realizadas para edições rápidas e testar a resposta da API para diferentes dados. O Postman também oferece um conjunto de extensões que permitem usá-lo como um pleno executor de testes, apesar de que desencorajamos o estilo de gravação e de reprodução que ele promove.



Brighter (iancooper.github.io/Paramore/Brighter.html) é uma biblioteca de código aberto para .Net que fornece suporte para implementação de Invocação de Comandos. Nós tivemos bom feedback das equipes que o estão usando, especialmente em conjunto com o padrão 'ports and adapters' e **CQRS**. Os times gostam especialmente da sua boa integração com **Polly** para prover funcionalidades de 'circuit breaking'.

Continuamos bem impressionados com o **Consul** (consul.io), uma ferramenta para descoberta de serviços (service discovery) que dá suporte à DNS e mecanismos de descoberta baseados em HTTP. Ele vai além das outras ferramentas de descoberta porque também provê verificações de integridade personalizadas para os serviços registrados, garantindo que instâncias instáveis de serviços sejam marcadas apropriadamente. Surgiram outras ferramentas para trabalhar com o Consul e torná-lo ainda mais poderoso. Consul Template (github.

ADOTE
48. Composer
49. Go CD
50. Mountebank
51. Postman

EXPERIMENTE
52. Boot2docker
53. Brighter
54. Consul
55. Cursive
56. Gitlab
57. Hamms
58. IndexedDB
59. Polly
60. REST-assured
61. Swagger
62. Xamarin
63. ZAP

AVALIE
64. Apache Kafka
65. Blackbox
66. Bokeh/Vega
67. Gor
68. NaCl
69. Origami
70. Packetbeat
71. pdfmake
72. PlantUML
73. Prometheus
74. Quick
75. Security monkey

EVITE
76. Citrix para desenvolvimento

FERRAMENTAS *continuação*

com/hashicorp/consul-template) permite que arquivos de configuração possuam informações do Consul, tornando mais fáceis operações como load-balancing de clientes utilizando mod_proxy. No mundo Docker, Registrator (github.com/gliderlabs/registrator) pode automaticamente registrar containers no Consul assim que eles apareçam com pouquíssimo esforço, tornando muito mais fácil o gerenciamento de configurações baseadas em contêineres.

Muitas, mas muitas histórias de fracassos na nossa indústria são causadas pela suposição de que redes são sempre confiáveis e que servidores HTTP respondem sempre correta e rapidamente. **Hamms** (github.com/kevinburke/hamms) é uma ferramenta de código aberto interessante que age como um servidor HTTP mal comportado, disparando uma série de falhas incluindo falhas de conexão e respostas lentas e/ou malformadas. Pode ser útil para testar se seu software lida com falhas graciosamente.

Muitas de nossas equipes trabalhando em projetos .Net recomendaram **Polly** (github.com/michael-wolfenden/Polly) como sendo útil ao criar sistemas baseados em micro-serviços. Ela incentiva a expressão fluente de políticas de tratamento de falhas transitórias e o padrão "Circuit Breaker" incluindo políticas como "Retry", "Retry for Ever" e "Wait and Retry Again". Bibliotecas já existem em outras línguas, Histrix para Java é um exemplo, e a Polly é uma adição bem-vinda para a comunidade .Net.

REST-assured (code.google.com/p/rest-assured) é uma linguagem específica de domínio em Java para testar e validar serviços RESTful. Ele simplifica o teste de serviços baseados em REST construídos sobre HTTP Builder. REST-assured dá suporte a diferentes requisições REST e pode ser usado para validar e verificar as respostas das APIs. Também provê um esquema de validação JSON e pode assim ser usado para verificar que se serviços estão retornando os tipos corretos entre os dados esperados.

ZED Attack Proxy (ZAP) (owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project) é um projeto da OWASP que permite investigar vulnerabilidades de segurança em uma página web de forma automatizada. Pode ser usado como parte dos testes periódicos de segurança, ou, então, integrado a uma pipeline de entrega contínua que proporciona verificações contínuas para as vulnerabilidades mais comuns. O uso de uma ferramenta como ZAP não substitui a necessidade de pensar cuidadosamente sobre segurança e testes mais completos, porém, por se tratar de outra ferramenta que ajuda a assegurar que nossos sistemas são mais seguros, é bom adicioná-la a caixa de ferramentas.

Muitos desenvolvimentos recentes em software corporativo envolvem sequências assíncronas de sequências de eventos imutáveis, em vez de requisições síncronas, ponto-a-ponto e de estado variável. **Apache Kafka** (kafka.apache.org) é um arcabouço de código aberto para gerenciamento de mensagens que dão suporte a esse estilo arquitetural publicando sequências de mensagens ordenadas para muitos consumidores leves e independentes. O design único do Kafka permite que o número de consumidores escale enquanto a ordenação das mensagens é fortemente mantida.

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, usa GPG para a criptografia e cada usuário possui sua própria chave, tornando mais fácil revogar o acesso granularmente. Há muita coisa acontecendo nesse espaço e há algumas outras opções a serem consideradas, incluindo **git-crypt** e **Trousseau**.

No mundo de ciência de dados e analíticos, muito do trabalho é feito com Python e R, linguagens que infelizmente oferecem poucas opções para plotagem de visualizações acessíveis pela web. Uma abordagem é converter o resultado da análise em algo que possa ser visualizado facilmente e com o qual se possa interagir dentro do browser. Sabemos de duas ferramentas que são tentativas de fazer isso. **Bokeh** (bokeh.pydata.org) é uma biblioteca Python e JavaScript que permite criar visualizações interativas "no estilo D3.js" mas com alto desempenho em grandes conjuntos de dados ou em streaming. **Vega** (trifacta.github.io/vega) é uma gramática declarativa para visualização de D3 que consome conjuntos de dados JSON gerados em servidor e traduz descrições de visualizações para código D3.js.

Gor (github.com/buger/gor) é uma ferramenta de código aberto para capturar e reproduzir o tráfego HTTP existente em um ambiente de teste a fim de continuamente testar o sistema com dados reais. Essa ferramenta pode ser utilizada para aumentar a confiabilidade de implantações de código e de alterações de configuração e infraestrutura.

O **NaCl** (nacl.cr.yp.to, que se pronuncia 'Salt') é uma biblioteca que provê um conjunto de funcionalidades para encriptação, decriptação e assinaturas projetada para tornar mais fácil a implementação de comunicações seguras em rede ou outros requisitos baseados em criptografia. Apesar dessas funções existirem em outras bibliotecas, NaCl promete alto desempenho e uma API fácil de usar. Atualmente dá suporte a C e C++ (suporte a Python está sendo desenvolvido).

Origami (facebook.github.io/origami) é uma ferramenta gratuita para o design de protótipos de interface com o usuário, com vários atalhos de teclado para funções comuns. Ela oferece a opção de exportar estruturas de código de Objective-C para iOS, Java para Android e JavaScript para Web. Essa ferramenta pode ser usada para construir rapidamente interfaces de usuário e testar seu fluxo. Pela experiência que coletamos de diversos times nossos, recomendamos investigar a ferramenta quando aplicável.

Pdfmake (github.com/bpampuch/pdfmake) é uma biblioteca que permite a criação e impressão de documentos PDF diretamente no navegador. Pelo Pdfmake pode-se construir um objeto que dá suporte a vários elementos estruturais do documento (tais como tabelas, colunas e estilos de texto) para que métodos auxiliares permitam criar, imprimir ou fazer o download do PDF sem sair do cliente JavaScript.

Desenvolver sistemas de software criando primeiro um grande número de diagramas detalhados é uma abordagem que, em nossa experiência, não se compara favoravelmente com as alternativas. No entanto, descrever uma parte particularmente complexa e intrincada do sistema com um diagrama é geralmente uma boa idéia, e a UML em si oferece uma série de diagramas úteis e comumente compreendidos. Nós gostamos do **PlantUML** (plantuml.sourceforge.net) para a criação de diagramas, pois permite expressar a intenção por trás dos diagramas de forma clara e textual, sem ter que mexer com ferramentas gráficas sobrecarregadas. Ter um formato textual também permite o controle de versão e armazenamento junto ao código-fonte.

SoundCloud recentemente liberou como código aberto um substituto para o Graphite, o **Prometheus** (prometheus.io). Desenvolvido como reação às dificuldades encontradas com o **Graphite** em seus sistemas de produção, Prometheus trabalha de forma diferente ao preferir um modelo HTTP baseado em pull (embora o modelo push, tal qual o Graphite, também seja aceito). Ele também vai além do Graphite ao ser construído para dar suporte a alertas baseados em métricas capturadas, se tornando um membro muito mais ativo no seu ferramental de operações. Certa cautela é recomendada ao adotar novas tecnologias no âmbito de monitoramento de ambientes de produção, mas os relatos iniciais indicam que a SoundCloud está bastante feliz com sua produção e o Docker também está atualmente contribuindo com seu desenvolvimento.

Quick (github.com/Quick/Quick) é um arcabouço de testes para Swift e Objective-C que vem junto com o **Nimble**, um arcabouço de verificações para testes. Quick ajuda a verificar o comportamento de programas em Swift e Objective-C. Ele tem o mesmo estilo sintático do **RSpec** e **Jasmine** e é fácil de configurar. Também é bastante organizado, permite a asserção de tipos e facilita o teste de código assíncrono.

Security monkey (github.com/Netflix/security_monkey) é mais uma ferramenta do Simian Army do Netflix, um conjunto de ferramentas que auxiliam o desenvolvimento de sistemas resilientes. Além de prover uma avaliação (parametrizável) de qualquer potencial vulnerabilidade de segurança na sua instância AWS, ela também pode ser usada para monitorar mudanças continuamente e alertar diferentes grupos conforme necessário. Existem algumas intersecções com outros serviços da própria Amazon como Trusted Advisor Report (aws.amazon.com/premiumsupport/trustedadvisor) e Cloudtrail (aws.amazon.com/cloudtrail) uma vez que ela foi desenvolvida antes que esses serviços fossem disponibilizados publicamente, mas suas funcionalidades vão muito além desses serviços. Se esses serviços não atenderem todas suas necessidades, vale a pena conferir o Security monkey.

LINGUAGENS & ARCABOUÇOS

Desde que falamos sobre **Nancy** (nancyfx.org) pela última vez no Technology Radar, ele se tornou escolha padrão em nossos projetos. As arquiteturas construídas com foco em menores fatias verticais e micro-serviços requerem processos mais simples para implantação e ferramental mais enxuto.

Um benefício da atual avalanche de arcabouços JavaScript para front-end é que, ocasionalmente, surge uma nova ideia que nos faz refletir. **React.js** (facebook.github.io/react) é um arcabouço para UI/View em que funções JavaScript geram HTML em um fluxo de dados reativo. Temos visto vários projetos menores alcançarem sucesso com React.js e desenvolvedores têm sido atraídos por sua abordagem limpa à componentização.

O **Spring Boot** (projects.spring.io/spring-boot) permite a fácil configuração de uma aplicação standalone baseada em Spring. É ideal para levantar novos micro-serviços e fácil de implantar. Também faz com que o acesso aos dados seja menos trabalhoso devido aos mapeamentos do Hibernate com menos código clichê. Nós gostamos que Spring Boot simplifica serviços Java construídos com Spring, mas aprendemos a ser cautelosos com as suas muitas dependências. O Spring ainda espreita abaixo da superfície.

O vasto uso de AngularJS continua nos projetos da ThoughtWorks, embora nem todas as experiências tenham sido positivas. Continuamos a aconselhar os times a avaliar se a complexidade adicional de uma aplicação JavaScript de página única é necessária para atender aos requisitos do sistema. Mas também recomendamos a avaliação de arcabouços alternativos e, nesta edição do Radar, destacamos o **Ember.js** (emberjs.com), que tem se tornado cada vez mais popular na ThoughtWorks. Ember é elogiado pela sua sólida abordagem de convenção acima de configuração, um time de desenvolvedores responsivo, desempenho e suporte a ferramentas de compilação através da Ember CLI.

No populoso ecossistema dos arcabouços JavaScript, gostaríamos de destacar o **Flight.js** (flightjs.github.com) como um arcabouço leve para a construção de componentes. O Flight.js faz o seu trabalho sem muita

mágica ao adicionar comportamentos aos elementos do DOM. É um arcabouço guiado por eventos e a sua natureza baseada em componentes promove a escrita de código desacoplado. Isto faz com que o teste individual de componentes seja possível de maneira mais fácil. Deve-se tomar cuidado, no entanto, quando componentes interagem entre si: há pouco suporte para testes de integração de componentes e corre-se o risco de atingir o nível de "event hell". Nós também gostamos do fato de que ele adota o uso de "mixins" funcionais para a adição de comportamento, ou seja, composição ao invés de herança.

Com algumas experiências nossas no mundo real, o **Swift** (developer.apple.com/swift) continua se mostrando bastante promissor. Alguns problemas, como demora na compilação, estão sendo tratados. No entanto, alterações frequentes na linguagem provocam esforço extra de desenvolvimento e bastante trabalho na manutenção de códigos antigos. Testes e refatoração também continuam sofridos. No balanço geral, no entanto, você ainda deve considerar o uso de Swift no início de novos projetos de desenvolvimento para o ecossistema da Apple.

ADOPT
77.Nancy

EXPERIMENTE
78.Dashing
79.Django REST
80.Arcabouço Ionic
81.Nashorn
82.Om
83.React.js
84.Retrofit
85.Spring Boot

AVALIE
86.Ember.js
87.Flight.js
88.Biblioteca Haskell Hadoop
89.Lotus
90.Reagent
91.Swift

EVITE
92.JSF

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®