Thought®

TECHNOLOGY RADAR

Nossas ideias sobre as tecnologias e tendências que moldam o futuro

thoughtworks.com/radar

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.

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 signi f car 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-identi f cação (en.wikipedia.org/wiki/De-identi f cation), 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 o f ine vai aparecer. Sabendo o quão difícil é inserir modo o f ine numa aplicação existente, há uma tendência para implementação de aplicações de página única com uma mentalidade “primeiro o f ine”. Uma técnica importante

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.

O **Git flow** 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

A **Mesos** (mesos.apache.org) é uma plataforma que abstrai recursos computacionais de camadas

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 confável 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 fuxos 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 especifcamente 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 efcaz de executar vários serviços em uma máquina local para fns 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

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 confável, 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 fuxo 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 o fine, 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 específcos 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 criptografa 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 gráfca baseada em D3 para explorar grandes conjuntos de dados multidimensionais. Ela compartilha com D3 a facilidade de criação de belos gráfcos interativos. É diferente no balanceamento da fexibilidade para criar quase qualquer tipo de visualização de dados para um modelo de programação mais simples, criando gráfcos 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 confguração 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 artifcial

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

novovgil,rias er.uão
também as preocupação 1 & J - ¡À 9` p € 0 p @ V ¦ - H V P V ¦ ¢ Y C U H W B V J^ P D V 0experi fr2ment mais vigil,vel. Uma er.uão
também depois O W R V G H 0
mqs as dc _ s f u ʿ ʿ ῃ ʿ ʿ ʿ ʿ Ů

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 fuxo 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.