Construindo uma arquitetura moderna de microservices na empresa Gilt

Depois de viver com microservices por três anos, a empresa Gilt pôde ver as vantagens no domínio das equipes, as fronteiras definidas pelas APIs e problemas complexos quebrados em partes menores. Yoni Goldberg, em uma apresentação na conferência QCon London 2015, explicou os desafios que ainda enfrenta em relação a ferramentas, integração de ambientes e monitoramento.
Goldberg, engenheiro de software líder na Gilt, descreve a empresa como um negócio de vendas relâmpago. Uma venda típica oferece um estoque limitado, mas com descontos a partir de uma hora especificada e funciona por um período específico, geralmente 36 horas. Com dezenas de milhares de pessoas acessando o site web de uma só vez para comprar os itens, a Gilt experimenta um pico extremo e curto no tráfego que gera cerca de 80% das vendas. Todas as decisões que podem afetar o desempenho do site web precisam ser levadas em consideração, pois pode causar um pico no tráfego regular de 50 a 100 vezes.

Como uma startup tradicional de 2007, a Gilt usou Ruby on Rails, PostgreSQL e Memcached. As coisas ocorreram bem, mas dois anos depois eles tinham 200.000 linhas de código e milhares de processos Ruby - conectados a base de dados - sobrecarregados devido o aumento de tráfego. Com qualquer desenvolvedor trabalhando na mesma base de código, a publicação poderia demorar mais de duas semanas para testar todas as integrações necessárias. E o maior obstáculo era que se alguma coisa estivesse errada, era necessário trabalhar duro para encontrar a raiz do problema.

Nesse ponto, além de mover para a JVM, a Gilt entrou no que Goldberg chama de a era do macro/microservices. Ele distingue entre: um macroservice o que trata um domínio em específico, tal como: vendas ou pagamentos; e um microservice que se obtém quebrando um macroservice dividido em serviços menores. A Gilt criou 10 macroservices para o núcleo do negócio, serviços que ainda estão em uso. Com todos os outros serviços dependendo desses, o serviços principais precisam executar com bons SLAs para manter o tempo de inatividade ao mínimo. O serviço de pagamento é um dos serviços principais: quando este serviço não está respondendo, os usuários não podem fazer as encomendas e a empresa não faz qualquer dinheiro. Um conjunto de serviços de apoio menos críticos, como as preferências de usuários, também utilizam os serviços principais; isso é bom para a experiência de usuário, mas o negócio continuará funcionando se algum deles cair. No topo desses serviços de suporte, outro conjunto de serviços geram as *views* para todos os usuários. Durante a construção dos serviços da Gilt, também foi adicionado uma base de dados dedicada para cada serviço, fornecendo as melhores bases de dados para cada necessidade. Essa nova arquitetura resolveu 99% dos problemas de escalabilidade, mas deixou os desenvolvedores com alguns problemas, como os novos serviços eram semi monolíticos e uma carência no responsável pelo

código. Os problemas com as publicações e longos ciclos de integração se mantiveram. E o principal problema, entretanto, é que não era mais divertido desenvolver código.

Movendo para microservices

Para superar os problemas pendentes, a Gilt criou diversos microservices e deu para as equipes a responsabilidade não apenas para o desenvolvimento, mas também de testar, publicar e monitorar. Esta propriedade também esclareceu como uma equipe basicamente se torna dona de um serviço. De acordo com Goldberg, o maior benefício veio dos pequenos escopos dos microservices, que facilitaram sua compreensão. É mais fácil de entender um serviço composto apenas de alguns milhares de linhas, e para entender um microservice de outra equipe quando se muda para desenvolver em outra equipe. Essa arquitetura removeu um grande problema na dependência do desenvolvimento entre as equipes. Agora, eles podem facilmente seguir para o Continuous Deployment (implantação contínua) com cada equipe decidindo por si mesma, quando publicar, até mesmo múltiplas vezes por dia.

Durante essa mudança, a Gilt começou a trabalhar com o que ela chama de LOSA, "*lots of small applications* - muitas aplicações pequenas", quebrando as páginas web em pequenos pedaços - basicamente microservices para aplicações web. Isso deixou as equipes trabalharem mais independentemente de outras equipes e Goldberg criou muita inovação e foco na experiência do usuário.

Desafios atuais

Apesar do sucesso da Gilt de mover do Rails para uma arquitetura de microservices, Goldberg enfatiza que a empresa ainda possui alguns desafios principais.

Publicação

Para começar, cada equipe publicava os serviços semimanualmente com seus próprios métodos. A falta de integração dificultava a execução dos testes que garantiam se a mudança não quebrou alguma outra coisa. A Gilt resolveu isso construindo uma ferramenta em volta do sbt*,* o que ajudava as equipes na primeira publicação de um ambiente de integração e então liberava para a produção. Durante o último ano, a empresa veio trabalhando em trazer a parte de operações para as equipes, também adotando o Docker e movendo para a nuvem.
Uma desvantagem apontada por Goldberg é que as publicações agora estão mais lentas, mas ele espera que isso possa ser melhorado nos próximos anos.

APIs

Durante o último ano, a Gilt estava saindo do estilo de comunicação RPC e começando a construir APIs REST. A principal vantagem segundo Goldberg é que uma API bem definida resolve diversos problemas, mas a sua descoberta também é importante. Como todas as APIs estão disponíveis no mesmo local, encontrar o que está disponível pode ser feito com uma simples pesquisa. A API também deve fornecer documentação; visualizando os modelos e recursos, deve ser possível entender o que está disponível e como está destinado a funcionar. E com a documentação gerada a partir do código, ela estará sempre correta e refletirá qualquer mudança nos recursos expostos.

Dependências

Todos esses microservices possuem muitas dependências entre eles. O maior desafio que Goldberg vê para os desenvolvedores é que para cada mudança é necessário garantir que não irá quebrar nenhum outro serviço. Se fizerem uma alteração que quebre algo, é necessário fazer isso em pequenos passos e todos os clientes precisam ser alterados para um novo *endpoint* antes do antigo ser deletado. Outro problema que eles passaram é que muitas das pequenas aplicações web repetiam as mesmas chamadas para um ou mais serviços, por exemplo, na geração de um perfil de usuário. Para limitar essas chamadas redundantes, a Gilt criou o que Goldberg chama de microservice de camada intermediária, um serviço que conhece as chamadas feitas para criar o perfil de usuário e que as aplicações web podem chamar. Esse microservice de camada intermediária conhece como otimizar, fazendo caching, para reduzir o número de chamadas feitas.

Responsabilidade de cada equipe

Assim como em muitas organizações, as equipes mudam na Gilt. Com todos os microservices ao redor, a empresa precisa ter certeza de que tem desenvolvedores suficiente que entendam as diferentes bases de códigos - e para Goldberg, a principal solução é a revisão de código. Quando todos os commits precisam ser revisados por pelo menos um outro desenvolvedor, isso aumenta a possibilidade que mais de um desenvolvedor realmente entenda o código. Goldberg também enfatiza que a equipe e não as pessoas são donas dos serviços, porque mesmo que algum indivíduo possa sair, a equipe geralmente fica mais tempo. As equipes tem autonomia para desenvolver um serviço em conjunto com outra equipe, o que também gera valor para o projeto.

Outro conceito é a responsabilidade do código. A mudança de uma base de dados por microservice resultou em centenas de bases de dados relacionais, e a Gilt precisa gerenciar o esquema de cada uma. Goldberg descreve como a Gilt separou completamente o esquema do banco do código de cada serviço, o que traz alguma sutileza e detalhes importantes: as mudanças precisam ser incrementais e não há rollbacks, então a equipe realmente precisa ter consciência de cada mudança que é feita.

Monitoramento

Com muitas dependências entre os serviços, Goldberg enfatiza que o monitoramento é importante. A Gilt usa diversas ferramentas de código aberto para obter diferentes matrizes, tal como: New Relic, Boundary e Graphite e também desenvolveu seu próprio sistema de monitoria, o CAVE, que também tem código aberto. A função básica do CAVE é configurar regras e criar alertas, por exemplo: quando o total de pedidos de todas as entregas dos EUA durante qualquer janela de cinco minutos caírem abaixo de um limite ou se 99% do tempo de resposta exceder um conjunto de valores. Essa é a técnica que Goldberd acredita ser melhor que as métricas padrões.

Seguindo em frente

Para Goldberg, a maior vantagem que a Gilt obteve com os microservices é a separação de responsabilidade das equipes. Ele acredita que quando os membros da equipe são donos de um serviço, eles tendem a tratá-lo como seu bebê. Outra grande promessa para os microservices que ele menciona é a quebra de problemas complexos em partes pequenas que qualquer um pode entender, um serviço de cada vez.

Dois desafios que Goldberg ainda possui são: monitoramento, por falta de ferramentas, e o ambiente de integração e desenvolvimento.

No final Goldberg avisa para quem for usar microservices que pegue as funcionalidades que ainda não existem e construa elas como microservice. Ele acredita que é difícil receber aceitação para quebrar algo que já existe e já funciona, mas que construir coisas novas é muito mais fácil de serem aceitas pelas pessoas.

César Augusto Cordeiro
RA: 1510026316
Noturno 10º Semestre
Eng. Computação