Lucas Felipe Barbosa – Resenha para a disciplina de projeto de software do capítulo 7 do livro “Engenharia de Software Moderna” do autor Marco Tulio Valente

Arquitetura de software pode ser entendida como uma forma de organizar estruturalmente e definir a forma como o desenvolvimento será feito. Ela abrange desde qual linguagem de programação será utilizada, qual será o banco de dados, passando por quais serão os padrões de projeto e arquiteturais, como será feita a organização dos pacotes do sistema e indo até mesmo à definição de qual será o padrão para definir o nome das variáveis e métodos.

O primeiro tipo de arquitetura vista no livro é a em camadas. Esse padrão organiza as classes de forma que uma classe só pode ser usada pela classe imediatamente acima dela. Isso torna mais fácil a manutenção e o entendimento do sistema. A arquitetura em camadas é muito utilizada em implementação de protocolos de rede.

O segundo tipo de arquitetura é a em três camadas. Ela subdivide o sistema em interface com o usuário, lógica de negócio e banco de dados. A evolução que esse tipo trouxe foi a separação entre as partes de um sistema. Existe ainda a possibilidade de existirem apenas duas camadas nesse modelo, em que a camada de interface e de aplicação são unidas, porém como desvantagem temos que com isso todo o processamento ocorre no lado dos clientes, e isso exigiria um maior poder de computação da máquina deles.

Arquitetura MVC: model-view-controller. Separando ainda mais as partes de uma aplicação, a arquitetura mvc separa o sistema em uma camada de visão, onde ocorre a exibição dos itens para o usuário, uma camada de controle que é responsável por tratar e processar os eventos gerados pelo usuário, e uma camada de modelo, que armazena os dados a serem acessados e registrados pelas camadas de visão e controle e também armazena as regras de negócio a serem utilizadas. A camada de visão e de controle podem se comunicar com a camada de modelo, porém o contrário não é verdade. A arquitetura MVC é muito utilizada para construção de sistemas corporativos. Ela favorece a especialização do trabalho, por exemplo desenvolvedores front-end só ficariam responsáveis pela interface gráfica, enquanto que os de back-end ficariam responsáveis pela parte de modelo.

Microsserviços são uma alternativa de desenvolver código que se opõe ao tradicional método monolítico de desenvolvimento de aplicações. Com essa arquitetura, cada parte do sistema denominada “serviço” executas seus processos de forma completamente independente. Dessa forma, caso ocorra algum erro em algum dos módulos apenas essa parte da aplicação fica prejudicada, e todas as outras partes continuam funcionando normalmente.

Além da vantagem de tornar as aplicações menos sujeitas a falhas globais, a arquitetura de microsserviços propõe uma escalabilidade granularizada. Caso um monólito esteja incorrendo em uma sobrecarga de uso ou problemas com performance, é necessário duplicar ou aumentar em n vezes seu tamanho até que se

chegue em um resultado aceitável, com isso vários processos e serviços que não demandam e não necessitavam desse aumento de capacidade também são aumentados, gerando um gasto completamente desnecessário. Com os microsserviços, apenas os processos (serviços) que realmente necessitam de ser melhorados tem sua capacidade aumentada.

Apesar de parecer uma solução muito boa, implementar uma arquitetura de microsserviços é uma tarefa bastante complexa e após implementado existem problemas de latência e outros tipos de problema que surgem com transações distribuídas e a necessidade de a aplicação ser capaz de processar operações de forma atômica.

Arquiteturas orientadas a mensagens oferecem uma alternativa muito interessante de desacoplamento no e no tempo entre os componentes de uma aplicação. Com as mensagens, não é necessário que o servidor de um componente se relacione com o outro, um lado apenas produz e o outro apenas consome. Desacoplamento no tempo é a capacidade de mesmo que algum dos sistemas produtor ou consumidor estejam disponível, ainda é possível que o outro continue trabalhando normalmente, e quando ele voltar ao ar, a fila retornará os mesmos dados de antes de ficar indisponível e o processamento poderá continuar sendo feito normalmente.

Diferentemente da arquitetura de mensagens por fila, a arquitetura publish/subscribe fornece uma alternativa em que um único publisher pode fornecer informação para mais vários subscribers. Ao ocorrer um evento em que o subscriber previamente assinou e disse que quer ser notificado quando o publisher publicasse o evento no broker, essa notificação é feita. Não apenas 1 subscriber, mas todos os subscribers são notificados quando o publisher publica o evento. Isso é muito útil para situações em que uma única entrada deve ocasionar diversas ações, como por exemplo ao comprar uma camisa online, é necessário gerar a NF, enviar o e-mail de confirmação de compra, mandar o pedido para o estoque, etc. Todas essas ações podem ser unificadas e serem geradas através de uma arquitetura publish/subscribe.

Para descrever organizações de sistemas que não são recomendadas existe um termo chamado anti-padrão arquitetural, e o mais conhecido deles é o “big ball of mud”. Nesse sistema qualquer módulo comunica-se com qualquer outro módulo e qualquer alteração em um lugar reflete em outro e isso gera um grande desgaste ao ser necessário realizar melhorias e correções, por isso é fundamental conhecer sobre os diferentes tipos de arquitetura e quais os casos de uso delas.