Arquitetura de Software

Aluno: Guilherme Valmir de Andrade.
Professor: Paulo Lapolli.

Introdução

Hoje em dia não é difícil nos depararmos com softwares mal organizados. Quando vamos dar manutenção em algum sistema desenvolvido por terceiros muitas vezes não sabemos por onde começar. Não sabemos, por exemplo, as consequências que o sistema terá se mexermos numa determinada linha de código. Para resolver este problema, foram definidos ao longo do tempo certos padrões de desenvolvimento, entre eles as arquiteturas de software.

Arquitetura de Software nada mais é que a estrutura do software vista de cima, isto é, todos os componentes que formam o software e suas relações. Ter um padrão de software no seu desenvolvimento pode auxiliar tanto na organização do código em si quanto na possibilidade de reutilizar classes e funções (o que auxilia na produtividade), além de proporcionar uma facilidade na manutenção do software.

Durante o processo de documentação da arquitetura de um software são levados em consideração diversos fatores, como: requisitos, conhecimento externo e experiência por parte do desenvolvedor do documento.

Os pontos no documento de requisitos que são levados em consideração na construção de um documento arquitetural são, geralmente: Desempenho, Disponibilidade, Modificabilidade, Segurança, Testabilidade e Usabilidade.

Veremos abaixo informações sobre as principais arquiteturas de software.

1. Arquitetura Cliente – Servidor.

Conceito

A arquitetura cliente-servidor é uma arquitetura surgida nos anos 90 no meio corporativo onde todo o gerenciamento da informação é dividido em módulos distintos. Temos claramente três atores nesta arquitetura: o servidor, o cliente e a rede. O servidor é o local onde todos os arquivos do software são armazenados e consultados. O cliente se dá basicamente pelo software instalado na máquina, que determina como o software fará a consulta dos dados que estão no servidor. Já a rede é o meio de transporte de toda a informação do software.

Funcionamento

Há uma ou mais máquinas portadoras dos dados do software que chamamos de servidores. As máquinas que contém o software instalado

(clientes) fazem uma requisição para o servidor. O servidor retorna estes dados à máquina cliente através da rede.

Vantagens e desvantagens

A arquitetura cliente-servidor tem como principal vantagem a contribuição para o crescimento da empresa, isto é, com todos os dados concentrados em um só lugar e a possibilidade de acessá-los e gerenciá-los em tempo real (gerar relatórios para tomadas de decisões, por exemplo) resulta diretamente no crescimento da empresa. Já como a principal desvantagem temos o alto custo de implantação (servidores costumam ser caros) e a dificuldade na atualização dos sistemas (é necessário atualizar o sistema em cada máquina onde ele está instalado).

Aplicação

A aplicação mais comum deste tipo de arquitetura é em empresas. Podemos citar uma empresa com 60 computadores e um servidor local. O banco de dados e uma versão servidor do sistema (rodando de maneira contínua) são instalados no servidor e o software é instalado em cada máquina cliente que estão conectadas na rede.

2 – Arquitetura MVC

Conceito

A arquitetura MVC é um padrão de arquitetura criado na década de 80 na Xerox Parc que separa todos os processos de software em três camadas bem definidas: Model (dados), View (usuário) e Controller (camada de controle). O diferencial nessa arquitetura é que cada processo pertence somente à sua camada correspondente, facilitando o desenvolvimento e possibilitando que o desenvolvedor possa mexer isoladamente em um determinado processo sem afetar outras características do sistema.

Funcionamento

Para deixar mais claro, vamos imaginar uma tela de formulário que recebe valores e os soma. Primeiramente eu tenho a tela onde os dados são digitados, ou seja, a parte gráfica (view). O usuário digita os dados e clica em um botão de ação que irá receber os dados e solicitar a soma (controller). Já a camada que recebe os dados e a ordem do que fazer com os dados, executa a tarefa e devolve o resultado ao cliente é a camada model. Contudo, a camada controller é a intermediária que possibilita a comunicação entre a entrada e o processamento dos dados.

Vantagens e desvantagens

Entre as principais vantagens do MVC está a escalabilidade, isto é, com cada parte do código em sua respectiva camada é simples e muito mais seguro fazer alterações. Já como desvantagem podemos citar o maior tempo de

desenvolvimento do software em seu desenvolvimento inicial e também não compensa utilizar MVC em pequenos projetos.

Aplicação

Vamos considerar um sistema de login. O usuário acessa a tela de login e digita seus dados. Até aí estamos falando de interface, ou seja, da camada view. Ele digita os dados de acesso e clica no botão para logar, ou seja, ele está inserindo dados e requisitando uma resposta. Agora entra em ação a camada controller, que pega o que o cliente digitou e encaminha essas informações para processamento na camada seguinte. Agora a camada model, a pedido do controller, analisa as informações que o usuário mandou e verifica no banco de dados se há usuário e senha com essas informações. Sendo essa requisição positiva ou negativa (true ou false), a camada model informa o resultado à camada controller, que por sua vez, se comunica com a camada view para apresentar a página inicial ou a tela de erro caso o login falhe.

3 – Arquitetura P2P

Conceito

A arquitetura P2P (ou peer-to-peer) é uma arquitetura criada na década de 80 e difundida na década de 90 que não tem um servidor central. Neste tipo de arquitetura os arquivos são compartilhados entre vários computadores na rede. Cada máquina na rede é representada por um nó (ou peer). Quanto mais máquinas possuírem um determinado arquivo, mais rápido será o download por outra máquina que o solicita.

Funcionamento

Vamos considerar que há um usuário que deseja acessar um determinado arquivo através da rede. Ele faz a solicitação ao sistema que faz uma varredura entre as diversas máquinas pelo mundo que também têm o sistema instalado e possuem o arquivo solicitado na máquina. Após o sistema encontrar os peers que possuem o arquivo, estas máquinas assumem o papel de servidores e o arquivo é acessado na máquina solicitante via download.

Vantagens e desvantagens

A grande vantagem da arquitetura P2P é a facilidade de conexão, ou seja, não é necessário adquirir um servidor central (o que custa caro). Já como desvantagem temos a variação no tempo de resposta na requisição, pois a velocidade de resposta depende da quantidade de peers no processo.

Aplicação

Como exemplo podemos citar o Torrent. O cliente quer baixar um arquivo via torrent. Ele baixa o arquivo de configuração, carrega este arquivo no sistema (uTorrent, por exemplo) e uma pesquisa global por máquinas (peers) que possuam o arquivo requisitado inicia. Após o software achar pelo menos uma

máquina, o processo de download é iniciado e a máquina (ou máquinas) onde o arquivo está presente passa a ser o servidor. Quanto mais máquinas possuírem o arquivo e estiverem na rede, mais rápido será o download.

4 – Computação Distribuída

Conceito

Computação distribuída ao processo de adicionar o poder de processamento e dados de um super computador ou diversas máquinas a uma máquina só (usuária ou cliente). Temos como exemplos de computação distribuída a arquitetura cliente-servidor, onde diversas máquinas interligadas se comunicam com um servidor ou a arquitetura p2p, onde o acesso aos arquivos se dá de maneira descentralizada.

As sessões de funcionalidade, vantagens e desvantagens e aplicação já foram dadas em outros tipos de arquitetura de computação distribuídas citados anteriormente.

5 – Arquitetura SOA

Conceito

SOA (arquitetura orientada a serviços) é um conceito de arquitetura criado na década de 90 que visa interligar diversos sistemas a fim de otimizar e descentralizar o desenvolvimento de um software.

Antigamente cada setor de uma empresa tinha um sistema próprio ou módulos de um sistema que não se comunicavam. O conceito SOA é interligar todos os sistemas de uma empresa para que todos os setores possam trabalhar em conjunto em tempo real, tendo uma visão ampla dos processos da empresa e não somente do seu setor.

Funcionalidade

Podemos citar uma empresa que precisa de um ERP que se conecte com um sistema CRM e um SCM tercerizados. Ou seja, através deste sistema ERP o funcionário terá acesso ao relacionamento com o cliente e também à logística de matéria prima e controle de estoque através da comunicação com serviços externos.

Vantagens e desvantagens

Como grande vantagem está o reuso dos serviços externos. E equipe de desenvolvimento pode reutilizar um mesmo serviço tercerizado em diversos sistemas. Outra vantagem é que no processo de desenvolvimento de um sistema, o serviço que soluciona algum requisito é totalmente abstraído do processo de desenvolvimento, gerando um tempo menor na entrega do software. Como desvantagem temos a falta de controle dos serviços externos justamente por se tratarem de serviços de outras empresas. Então não há, por exemplo, a

reversão de alguma solicitação feita para este serviço ou controle caso ele apresente alguma falha.

Funcionamento

Podemos citar o software de um banco onde há a validação de CPF de um usuário. Há uma regra básica que valida a sintaxe de um CPF mas não verifica se ele existe de fato ou não. Para isso, os sistemas bancários solicitam a um serviço externo onde há um banco de dados com todos os CPFs cadastrados se o CPF em questão faz parte da lista ou não, removendo qualquer CPF inválido no cadastro.

Conclusão

Os padrões de arquitetura, quando bem utilizados e executados, sem dúvida alguma, auxiliam na manutenção do software desenvolvido. Se a equipe tem um documento para se basear durante o desenvolvimento as chances de sucesso do sistema na empresa são muito maiores.

O conceito SOA já é amplamente utilizado e eu acredito que seu crescimento seja cada vez maior nos próximos anos. Descentralizar serviços do sistema através de meios externos facilita e acelera bastante o processo de desenvolvimento e entrega do software.

Referências

SLIDESHARE. **Arquitetura Cliente Servidor**. Disponível em: https://pt.slideshare.net/marciaabrahim/arquitetura-cliente-servidor. Acesso em: 28 de Outubro de 2019.

TABLELESS. **O que é MVC**. Disponível em: https://tableless.com.br/mvc-afinal-e-o-que/. Acesso em: 28 de Outubro de 2019.

DEVMEDIA. **Padrão MVC – JAVA Magazine**. Disponível em: https://www.devmedia.com.br/padrao-mvc-java-magazine/21995. Acesso em: 28 de Outubro de 2019.

MACORATTI. **Padrões de Projeto : O modelo MVC - Model View Controller**. Disponível em: http://www.macoratti.net/vbn_mvc.htm. Acesso em: 28 de Outubro de 2019.

SLIDESHARE. **Arquitetura Peer to-Peer (p2p)**. Disponível em: https://pt.slideshare.net/niltonrpereira/arquitetura-peer-topeer-p2p. Acesso em: 28 de Outubro de 2019.

COMPARTILHAMENTO DE ARQUIVOS. **Vantagens e Desvantagens do Modelo P2P.** Disponível em: https://compartilhamentodearquivoss.wordpress.com/category/vantagens-e-desvantagens-do-modelo-p2p/. Acesso em: 28 de Outubro de 2019.

WEBARTIGOS. **Computação Distribuída.** Disponível em: https://www.webartigos.com/artigos/computacao-distribuida/2903. Acesso em: 28 de Outubro de 2019.

DEVMEDIA. **Vantagens e Desvantagens de SOA.** Disponível em: https://www.devmedia.com.br/vantagens-e-desvantagens-de-soa/27437. Acesso em: 28 de Outubro de 2019.

SPÍNOLA, Rodrigo Oliveira; BARCELOS, Rafael Ferreira. Fundamentos de Arquitetura de Software. **Engenharia de Software Magazine**, São Paulo, v. 6, p.28-34, 2008.