Arquitetura Cliente-Servidor

A tecnologia cliente/servidor é uma arquitetura na qual o processamento da informação é dividido em módulos ou processos distintos. Um processo é responsável pela manutenção da informação (servidores) e outros responsáveis pela obtenção dos dados (os clientes).

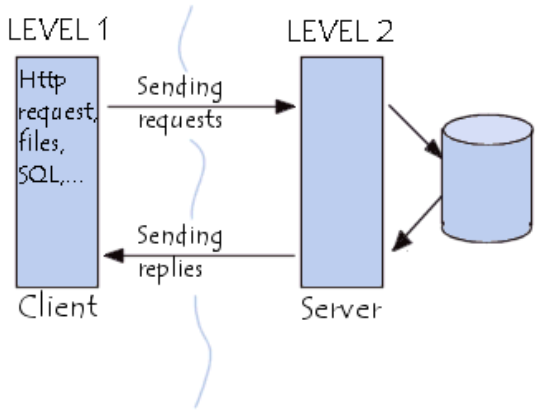
Os processos cliente enviam pedidos para o processo servidor, e este por sua vez processa e envia os resultados dos pedidos.

É no servidor que normalmente ficam os sistemas mais pesados da rede, tais como o banco de dados. As máquinas clientes são menos poderosas, pois não rodam aplicativos que requerem tantos recursos das máquinas.

O importante em uma máquina em arquitetura Cliente/Servidor não é que todas as máquinas sejam do mesmo fabricante ou do mesmo tipo. O que realmente é importante, é o fato de todas as máquinas poderem se interligar pela rede, com o mesmo tipo de protocolo de acesso (TCP/IP, NetBEUI).

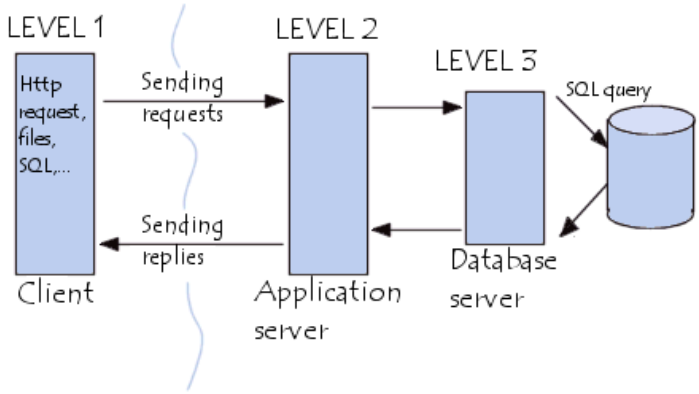
Como Funciona a Arquitetura em Dois Níveis

A arquitetura em dois níveis (também chamada de arquitetura 2-tier) caracteriza os sistemas cliente/servidor pelos quais o cliente pede um recurso e o servidor responde diretamente ao pedido, utilizando seus próprios recursos. Isso significa que o servidor não requer outro aplicativo para proporcionar parte do serviço:



Qual é a Estrutura da Arquitetura em Três Níveis

Na arquitetura em três níveis, existe um nível intermediário, o que significa que ela está compartilhada entre um cliente, ou seja, o computador que solicita recursos, equipado com uma interface de usuário (geralmente um navegador) encarregado da apresentação; o servidor de aplicativo (também chamado de software intermediário), cuja tarefa é proporcionar os recursos solicitados e o servidor de dados, que fornece os dados necessários ao servidor de aplicativo:



O emprego maciço do termo arquitetura em três níveis designa as arquiteturas que envolvem servidores de empresa ou de bancos de dados.

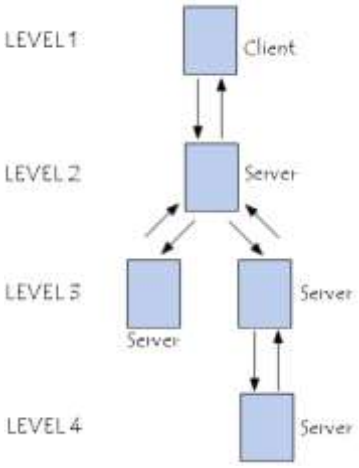
Comparação Entre os dois Tipos de Arquitetura

A arquitetura em dois níveis é uma arquitetura cliente/servidor, na qual o servidor é polivalente, ou seja, ele pode fornecer todos os recursos pedidos pelo cliente diretamente.

Em compensação, na arquitetura em três níveis, os aplicativos do servidor são descentralizados, o que quer dizer que cada servidor é especializado em uma tarefa (servidor web e servidor de banco de dados, por exemplo). A arquitetura em três níveis permite maior flexibilidade, maior segurança, pois ela pode ser definida independentemente para cada serviço e em cada nível, e melhor desempenho, dada a divisão das tarefas entre diversos servidores.

Como Funciona a Arquitetura Multiníveis

Na arquitetura em três níveis, cada servidor (níveis 2 e 3) efetua uma tarefa específica. Assim sendo, um servidor pode utilizar os serviços de um ou vários servidores para oferecer o seu próprio serviço. Consequentemente, a arquitetura em três níveis é potencialmente uma arquitetura de n (vários) níveis:



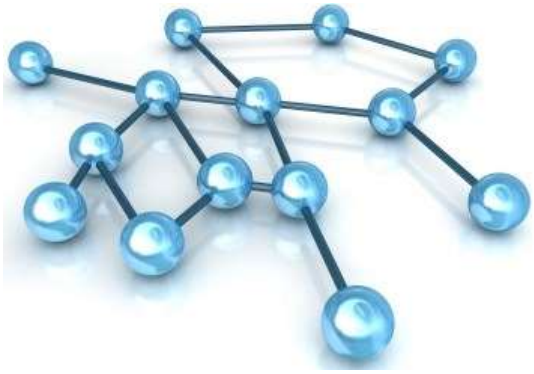
Internet é uma enorme rede de computadores interligados em escala mundial. A idéia de interligar computadores não é recente, pois na década de 60, nos Estados Unidos já havia a intenção de conectar computadores.

Naquela época, a Guerra Fria estava ficando cada vez mais quente, por isso, o Governo estadunidense precisava que as trocas de informações entre bases militares fossem mais seguras. Em decorrência desta necessidade, surgiu a ARPANET (Advanced Research Projects Agency Network), considerada a mãe da Internet.

Este projeto consistia em realizar a interconexão dos computadores existentes através de um sistema conhecido como comutação. Neste sistema, as informações eram divididas em pequenos “pacotes” que continham trecho de dados, desta forma cada computador enviava seu trecho e este, era montado no computador que havia solicitado os dados.

Como cada vez mais pessoas estavam se conectando à ARPANET, foi necessário desenvolver outra forma de comunicação que fosse mais eficiente. Um esquema denominado “Protocolo de Internet”, o TCP/IP, permitia que o fluxo das informações transitasse pela rede.

Como a Internet se tornou uma rede cada vez maior de computadores interligados, houve necessidade de dedicar alguns computadores para prover serviços à rede, enquanto os demais acessariam estes serviços. Portanto, estes computadores que proviam serviços eram denominados Servidores, enquanto ao que acessavam os serviços eram chamados de clientes.



Existem vários tipos de servidores. Os mais conhecidos são: Servidor de Fax, de arquivos, web, email, imagens, FTP. Cada um destes servidores executa uma função, por exemplo, para você visualizar esta página no Baixaki, você está utilizando o Servidor web, o qual é responsável pelo armazenamento das páginas de um site.

Para que você consiga abrir estas páginas web, você utiliza um navegador, portanto o navegador é o cliente e o site é o servidor, pois o primeiro acessa informações disponibilizadas pelo segundo. Desta forma, as redes que utilizam servidores são chamadas do tipo Cliente-Servidor

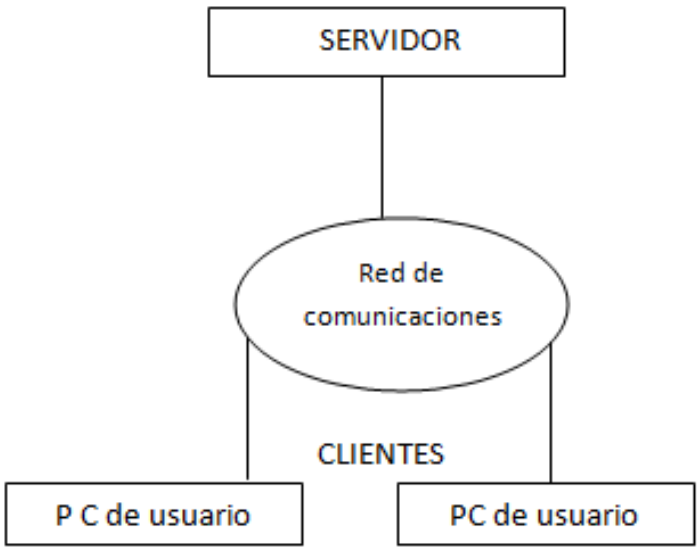
Esta arquitetura se divide em duas partes claramente diferenciadas, a primeira é a parte do servidor e a segunda a de um conjunto de clientes.

Normalmente o servidor é uma máquina bastante potente que atua como depósito de dados e funciona como um sistema gerenciador de banco de dados (SGBD).

Por outro lado, os clientes costumam ser estações de trabalho que solicitam vários serviços ao servidor.

Ambas partes devem estar conectadas entre si mediante uma rede.

Uma representação gráfica deste tipo de arquitetura seria a seguinte.



Este tipo de arquitetura é a mais utilizada atualmente, devido ao fato de ser a mais avançada e a que melhor evoluiu nestes últimos anos.

Podemos dizer que esta arquitetura necessita três tipos de software para seu correto funcionamento:

Software de gerenciamento de dados: Este software se encarrega da manipulação e gerenciamento de dados armazenados e requeridos pelas diferentes aplicações. Normalmente este software se hospeda no servidor.

Software de desenvolvimento: este tipo de software se hospeda nos clientes e só naqueles que se dediquem ao desenvolvimento de aplicações.

Software de interação com os usuários: Também reside nos clientes e é a aplicação gráfica de usuário para a manipulação de dados, sempre é claro, a nível de usuário (consultas principalmente).

A parte destes existem mais aplicações software para o correto funcionamento desta arquitetura, mas já estão condicionados pelo tipo de sistema operacional instalado, o tipo de rede na qual se encontra, etc.

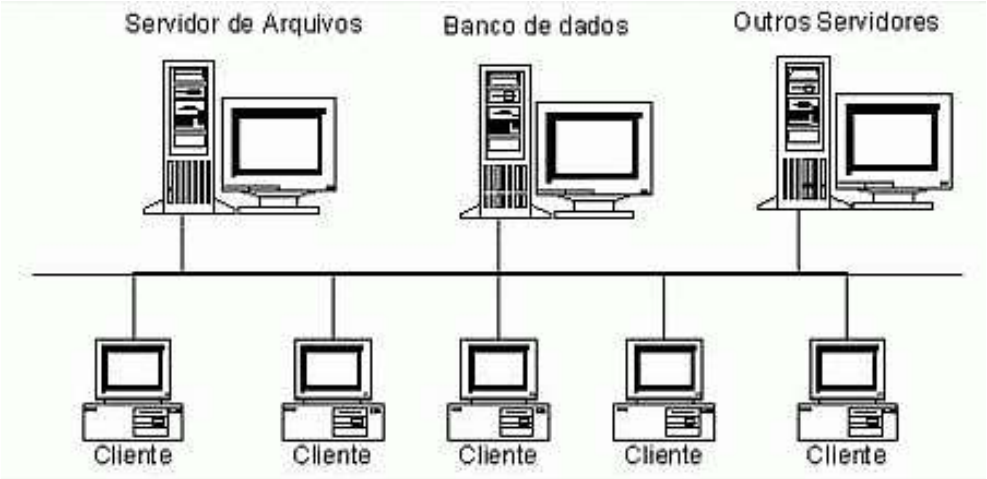
Modelo Cliente/Servidor

Cliente-servidor é um modelo computacional que separa clientes e servidores, sendo interligados entre si geralmente utilizando-se uma rede de computadores. Cada instância de um cliente pode enviar requisições de dado para algum dos servidores conectados e esperar pela resposta. Por sua vez, algum dos servidores disponíveis pode aceitar tais requisições, processá-las e retornar o resultado para o cliente. Apesar do conceito ser aplicado em diversos usos e aplicações, a arquitetura é praticamente a mesma.

O modelo Cliente/Servidor, foi criado tendo como base a descentralização dos dados e recursos de processamento, em oposição ao modelo Centralizado utilizado na época em que o Mainframe dominava absoluto. No modelo Cliente/Servidor, conforme indicado pela figura abaixo, em uma rede de computadores, existem uma ou mais máquinas que atuam como Servidores, disponibilizando recursos para as demais máquinas, as quais atuam como Clientes.

A máquina servidor é um host que está executando um ou mais programas de servidor que partilham os seus recursos com os clientes.

Um cliente não compartilha de seus recursos, mas solicita o conteúdo de um servidor ou função de serviço. Os clientes, portanto, iniciam sessões de comunicação com os servidores que esperam as solicitações de entrada.



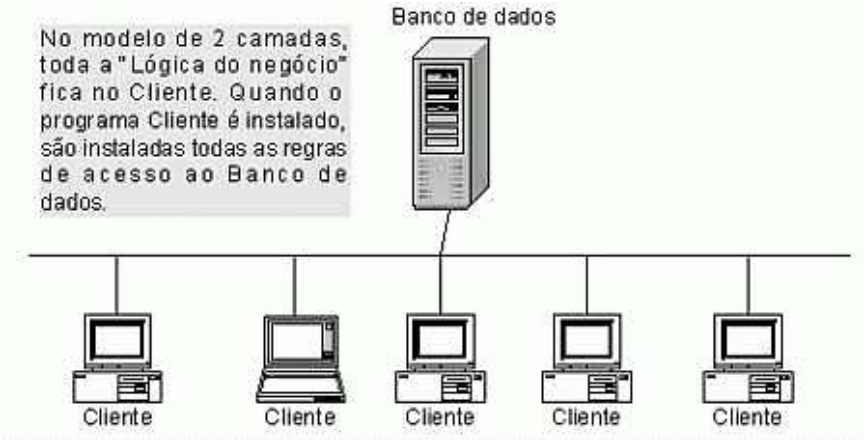
Modelo Cliente-Servidor

Conforme pode ser visto na figura, temos Servidores para Arquivos, Banco de dados e outras funções, tais como: Servidores de impressão, Servidores Web, etc.

Estas redes, tipicamente, são formadas por Servidores, os quais são equipamentos com um maior poder de processamento e armazenamento do que os clientes, os quais, na maioria dos casos, são Microcomputadores ligados em rede.

Aplicações em 2 Camadas

No início da utilização do modelo Cliente/Servidor, as aplicações foram desenvolvidas utilizando-se um modelo de desenvolvimento em duas camadas. Neste modelo, um programa, normalmente desenvolvido em um ambiente de desenvolvimento, como o Visual Basic, Delphi ou Power Builder, é instalado em cada Cliente. Este programa acessa dados em um servidor de Banco de dados, conforme ilustrado na figura abaixo:



2 Camadas

No modelo de duas camadas, temos um programa que é instalado no Cliente, programa esse que faz acesso a um Banco de dados que fica residente no Servidor de Banco de dados. Na maioria dos casos, a máquina do Cliente é um PC rodando Windows, e a aplicação Cliente é desenvolvida utilizando-se um dos ambientes conhecidos, conforme citado anteriormente.

Sendo a aplicação Cliente, um programa para Windows (na grande maioria dos casos), esta deve ser instalada em cada um dos computadores da rede, que farão uso da aplicação. É o processo de instalação normal, para qualquer aplicação Windows. No modelo de 2 camadas, a aplicação Cliente é responsável pelas seguintes funções:

Apresentação: O Código que gera a Interface visível do programa, que é utilizada pelo usuário para acessar a aplicação, faz parte da aplicação Cliente. Todos os formulários, menus e demais elementos visuais, estão contidos no código da aplicação Cliente.

Caso sejam necessárias alterações na interface do programa, faz-se necessária a geração de uma nova versão do programa, e todos os computadores que possuem a versão anterior, devem receber a nova versão, para que o usuário possa ter acesso as alterações da interface. Aí que começam a surgir os problemas no modelo de 2 camadas: Uma simples alteração de interface, é suficiente para gerar a necessidade de atualizar a aplicação, em centenas ou milhares de computadores. O gerenciamento desta tarefa, é algo extremamente complexo e de custo elevado.

Lógica do Negócio: Aqui estão as regras que definem a maneira como os dados serão acessados e processados, as quais são conhecidas como “Lógica do Negócio”. Fazem parte das Regras do Negócio, desde funções simples de validação da entrada de dados, como o cálculo do digito verificador de um CPF, até funções mais complexas, como descontos escalonados para os maiores clientes, de acordo com o volume da compra.

Questões relativas a legislação fiscal e escrita contábil, também fazem parte da Lógica do Negócio. Por exemplo, um programa para gerência de Recursos Humanos, desenvolvido para a legislação dos EUA, não pode ser utilizado, sem modificações, por uma empresa brasileira..

Alterações nas regras do negócio são bastante frequentes, ainda mais com as repetidas mudanças na legislação do nosso país. Com isso, faz-se necessária a geração de uma nova versão do programa,

cada vez que uma determinada regra muda, ou quando regras forem acrescentadas ou retiradas. Desta forma, todos os computadores que possuem a versão anterior, devem receber a nova versão, para que o usuário possa ter acesso as alterações .

Mais problemas com o modelo de 2 camadas: Qualquer alteração nas regras do negócio, é suficiente para gerar a necessidade de atualizar a aplicação, em centenas ou milhares de computadores. O gerenciamento desta tarefa, é algo extremamente complexo e de custo elevado.

A outra camada, vem a ser o Banco de dados, o qual fica armazenado em Servidor da rede. Uma aplicação desenvolvida em Visual Basic, a qual acessa um Banco de dados em um servidor Microsoft SQL Server, é um típico exemplo de uma aplicação em 2 camadas.

Com a evolução do mercado e as alterações da legislação, mudanças nas regras do negócio são bastante frequentes. Com isso o modelo de duas camadas, demonstrou-se de difícil manutenção e gerenciamento, além de apresentar um custo de propriedade muito elevado.

Isto sem contar com o problema conhecido como “DLL Hell” (Inferno das DLLs), onde diferentes aplicativos, instalam diferentes versões da mesma DLL e um conflito é gerado. É o caso típico onde a instalação de um programa, faz com que um ou mais programas, instalados anteriormente, deixem de funcionar. Em resumo, como diria um famoso comediante: “Uma verdadeira visão do Inferno”.

Inferno para o usuário, que não tem os programas funcionando como deveriam; inferno para a equipe de desenvolvimento que não tem o seu trabalho reconhecido e, normalmente, tem que trabalhar apenas “apagando incêndios”; e inferno para a Administração/Gerência da rede que não consegue gerar os resultados esperados pela Administração da empresa, apesar dos elevados valores já investidos.

Resumindo:

Sistemas em camadas surgiram para:

Melhor aproveitar os PCs da empresa

Oferecer sistemas com interfaces gráficas amigáveis

Integrar o desktop e os dados corporativos

Em outras palavras, permitiram aumentar a escalabilidade de uso de Sistemas de Informação

Os primeiros sistemas cliente-servidor eram de duas camadas

Camada cliente trata da lógica de negócio e da UI

Camada servidor trata dos dados (usando um SGBD)

Pode parecer difícil de acreditar, mas um grande número de empresas ainda tem a maioria dos seus aplicativos baseados no modelo Cliente/Servidor de 2 camadas. Em busca de soluções para os problemas do modelo de duas camadas, é que surge a proposta do modelo de 3 camadas.

Aplicações em 3 Camadas

Modelo em três camadas, derivado do modelo ‘n’ camadas, recebe esta denominação quando um sistema cliente-servidor é desenvolvido retirando-se a camada de negócio do lado docliente. O desenvolvimento é mais demorado no início comparando-se com o modelo em duas camadas pois é necessário dar suporte a uma maior quantidade de plataformas e ambientes diferentes.

Em contrapartida, o retorno vem em forma de respostas mais rápidas nas requisições, excelente performance tanto em sistemas que rodam na Internet ou em intranet e mais controle no crescimento do sistema.

Como uma evolução do modelo de 2 camadas, surge, com o crescimento da Internet, o modelo de três camadas. A idéia básica do modelo de 3 camadas, é “retirar” as Regras do Negócio do cliente e centralizá-las em um determinado ponto, o qual é chamado de Servidor de Aplicações. O acesso ao Banco de dados é feito através das

Então, ao invés de enviar os comandos para todo mundo e cada computador realizar a simulação, agora temos um computador central. Ele será responsável por receber os dados de todos os jogadores, realizar as simulações e então devolver a eles o resultado da simulação.

No modelo puro, os computadores dos jogadores seriam terminais burros, terminais onde não possuem processamento, senão as atualizações enviadas do servidor. Isso reduz o tráfego gerado: agora cada jogador só se comunica com o servidor. Logo, o único computador que precisa ser robusto é o servidor. Outra vantagem em relação ao P2P é que, como a conexão é feita somente com o servidor, o jogo não vai ser executado no passo do jogador com maior latência.

Portanto, qualquer lag que ocorrer na experiência está relacionado apenas com a conexão do usuário e o servidor. Outra vantagem é que diferente dos jogos P2P, que necessitam de um lobby para reunir os jogadores antes da partida, a arquitetura Cliente-Servidor permite facilmente conectar um jogador a qualquer momento, sendo perfeito para os MMOs.

Entretanto, o modelo tem um ponto fraco: na percepção do usuário, é possível perceber um atraso entre o comando e o movimento do avatar do jogador. Isso ocorre porque quando o usuário entra com um comando, ele é enviado para o computador. Ou seja, temos a latência de envio (vamos chamar de “le”).

Então o comando é processado, no que pode levar um tempo “ex”, e logo o servidor manda a resposta, que leva um tempo “lr”. Logo, a diferença de quando o evento de comando ocorreu e sua atualização no computador do jogador tem a duração de (le+ex+lr). Já em uma conexão P2P levará apenas o tempo dele para para ser atualizado. Entretanto, algumas medidas podem ser tomadas para evitar isso.

Predição do Estado de Jogo no Cliente

Para se ter melhor fluxo nas atualizações, podemos criar um artifício chamado de predição: dando um pouco de autonomia para o cliente para prever o resultado final do comando. Ou seja, se o jogador deu o comando para seguir em frente, o jogo fará o jogador seguir em frente imediatamente, prevendo que o mesmo ocorrerá no servidor.

Enquanto isso, o comando é processado no servidor e retorna o resultado para o cliente. Aí é que entra o pulo do gato. O cliente vai utilizar do resultado do servidor para validar sua previsão. Se a previsão for validada, o jogo simplesmente continua, entretanto se a previsão for diferente do que ocorreu no servidor, o cliente vai corrigir a situação do jogo para estar conforme o servidor.

Essa é uma situação interessante, pois quando o cliente e o servidor não concordam entre si, o servidor sempre terá a razão. Você deve estar pensando agora: e se fizéssemos o cliente ter autonomia total sobre o personagem e mandássemos apenas a atualização?

Esse realmente seria um método mais simples, mas também é mais vulnerável. Permitindo esse nível de autonomia no cliente facilitaria a criação de programas de trapaças, pois basta alterar os dados antes de ser enviado por pacote para o servidor. Com um modelo onde o servidor tem autoridade, fica muito mais difícil trapacear, pois se a alteração irregular do estado de jogo no cliente não condizer com o estado do jogo do servidor, o cliente vai ser resetado para o estado do servidor.

_____

_____