**Arquitetura cliente-servidor**

A arquitetura cliente-servidor consiste no conceito de dividir o programa em dois módulos.

É uma arquitetura onde o processamento da informação é dividido em módulos ou processos distintos. Um processo é responsável pela manutenção da informação (Servidor), enquanto que outro é responsável pela obtenção dos dados (Cliente). BATTISTI, (2001, pág. 38).

As principais características do cliente são:

* Inicia e termina as conversações com os Servidores, solicitando serviços distribuídos;
* Não se comunica com outros clientes;
* Torna a rede “transparente” ao usuário;
* Pode conter hardware simples;
* O software tem início e fim definido;

As principais características do servidor são:

* Execução contínua, aguardando ser chamado pelo cliente
* Recebe/Responde solicitações dos clientes
* Presta serviços distribuídos
* Atende a diversos clientes simultaneamente
* O servidor necessita ser uma máquina com maior poder de processamento

Sistema inovador surgido nos anos 90 e muito utilizado no meio corporativo, baseado em três componentes principais: gerenciamento de banco de dados, que funcionam como servidores; redes, que funcionam como meio de transporte de dados e, finalmente, softwares para acesso aos dados: Clientes. BATTISTI, (2001, pág. 39).

Aplicações em duas camadas eram utilizadas no princípio da implantação desse modelo. O cliente é responsável pela função de apresentação, ou seja, toda a parte gráfica, e pela lógica do negócio, que são as regras que definem a maneira como os dados serão acessados e processados. A principal desvantagem desse modelo é que, caso haja uma alteração na interface ou na lógica de negócio, é necessário que a aplicação seja atualizada em todos os clientes.

Nas aplicações em três camadas, a lógica de negócio é transferida para um servidor de aplicação. Porém, nesse tipo de aplicação, o problema com a atualização da interface persiste. O acesso do cliente ao banco de dados é feita de acordo com as regras contidas no servidor de aplicação. O servidor de banco de dados possui toda a informação necessária para que a aplicação funcione. Suas principais vantagens são:

* Atualização mais fácil das regras de negócio;
* Maior segurança e controle do acesso ao banco de dados;

**Desenvolvimento multiplataforma**

O conceito por trás do desenvolvimento multiplataforma é de criar uma aplicação que seja executável em vários sistemas operacionais, independentemente da empresa que possui o S.O., utilizando o mesmo código-fonte. Para tal, são necessárias IDEs capazes de recompilar o mesmo código-fonte para diversos tipos de executáveis.

Muitas dessas IDEs (a exemplo do Qt, da Qt-project), são softwares livres, não-pertencentes à empresas de código fechado, como Microsoft ou Apple, além de serem gratuitas. Sua lista de disponibilidade de compilações costuma incluir os principais sistemas operacionais para computadores de mesa (Desktop), assim como as versões mobile.

A linguagem de programação nessas IDEs costuma ser o C++, devido à sua popularidade, versatilidade, rapidez e por já possuir o conceito de Programação Orientada a Objetos, mas existem IDEs que utilizam C, Java, Python, entre outros. Nessas IDEs, é comum o uso de classes, interfaces e bibliotecas próprias da IDE, de modo que, na hora de compilar o código, o próprio compilador “traduza” automaticamente o comando próprio para um semelhante que o sistema operacional de destino entenda.

Existem alguns problemas inerentes ao desenvolvimento multiplataforma. É comum que a velocidade de processamento do programa seja mais lenta, além de, muitas vezes, ser difícil encontrar ajuda para um específico problema encontrado, visto que normalmente tais IDEs têm suporte via comunidade, onde várias pessoas alteram o mesmo código.

Por fim, é viável alegar que as aplicações de desenvolvimento (IDEs) voltadas para a geração de programas multiplataforma são, portanto, uma idéia válida e que deve ser incentivada nas empresas que desenvolvem softwares.