# ESTILOS ARQUITETURAIS

**CLIENTE/SERVIDOR:**

**R: Devido a diversas limitações do modelo de compartilhamento de arquivos, surgiu o estilo arquitetural Cliente/Servidor, para substituir o compartilhamento de arquivos por um servidor inteligente de banco de dados. Seu uso é bastante diferente comparado a sua última versão, ou seja, antes transferia arquivos completos para as aplicações do usuário, o servidor retorna apenas as informações solicitadas. Com a nova versão, estará nítida para a separação de interesses, camadas cliente e servidora.**

**N-CAMADAS:**

**R: A arquitetura três camadas representou um avanço significante em relação aos modelos anteriores. No entanto, arquitetos e desenvolvedores começaram a reconhecer que essa arquitetura apresentava uma limitação: havia uma dificuldade em se separar lógica de interação e lógica independente da apresentação. Perceberam que, havendo essa distinção, os sistemas seriam mais escaláveis. Então, ocorreu nova separação de interesses, então, criou-se a camada de interação, e a camada de fonte de dados passou a se chamar camada recursos.**

**REST:**

**R: REST (Representational State Transfer): Criado em 2000 por Roy Fielding, REST é um estilo de arquitetura para sistemas distribuídos hipermídia Web. Refere-se a uma coleção de princípios arquiteturais que determinam como recursos devem ser definidos e endereçados: torna extremamente simples a interface de transmissão de dados no protocolo HTTP, evitando camadas adicionais como SOAP, controle de sessão ou cookies.**

**PLUGIN:**

**R: Também chamado de plug-in, addin, add-in, addon, add-on, snap-in ou extensão, Plugin é um programa de computador que interage com uma aplicação hospedeira (um browser web, um cliente de e-mail, as IDE´s Eclipse ou Netbeans) para prover uma funcionalidade bem específica, sob demanda (on demand).**

**CHAMADA E RETORNO:**

**R: Chamada e Retorno (Call-and-Return): É caracterizada por um modelo de ativação (como os componentes são ativados e como a informação passa entre eles) que envolve uma thread de controle para executar as operações de invocação (ALBIN, 2003). Segundo Pressman (2006), há dois subtipos:**

1. **Programa principal e subrotina/subprograma: A hierarquia de controle é decomposta em programa principal e subprogramas. O programa principal delega aos subprogramas todas as atividades que lhe são solicitadas. Um dos benefícios mais celebrados desse modelo é a independência de desenvolvimento de cada módulo.**
2. **Chamada de procedimentos remotos: Usa a mesma estratégia de programa principal e subprogramas, mas distribui esses elementos em vários computadores de uma rede.**

**ARQUITETURA ORIENTADA A SERVIÇOS:**

**R: Arquitetura Orientada a Serviços (AOS ou SOA): é uma arquitetura de software em que as funcionalidades são agrupadas por processos de negócio e empacotadas como uma coleção de pequenos módulos chamados serviços. Pode-se definir serviço como sendo uma função de um negócio, sem estado, autocontida, que aceita requisições por meio de interfaces padronizadas e interoperáveis, executa uma unidade de trabalho do processo de negócio, e retorna resultados.**

**Com essa definição de serviço, uma aplicação SOA não seria muito diferente de aplicações EJB bem escritas, você poderia pensar. No entanto, SOA vai além, pois atende a três requisitos especiais:**

**1. Baixíssimo acoplamento: Serviços têm interfaces autodestruitivas e independentes de plataforma.**

**2. Altíssima interoperabilidade: É possível integrar serviços de qualquer plataforma tecnológica.**

**3. Orquestração: Um serviço/processo central controla a execução das operações. Os serviços envolvidos não "sabem" que estão envolvidos em um processo composto.**

**4. Coreografia: Cada serviço web envolvido sabe quando executar as suas operações e com quem interagir. É um esforço colaborativo que foca a troca de mensagens em processos de negócios. Todos os participantes precisam conhecer o processo de negócios, as operações a executar e as mensagens trocadas e os instantes em que isso deve ocorrer.**

**MICROSSERVIÇOS:**

**R: Os microsserviços são uma arquitetura e uma abordagem para escrever programas de software. Com eles, as aplicações são desmembradas em componentes mínimos e independentes. Diferentemente da abordagem tradicional monolítica em que toda a aplicação é criada como um único bloco, os microsserviços são componentes separados que trabalham juntos para realizar as mesmas tarefas. Cada um dos componentes ou processos é um microsserviço. Essa abordagem de desenvolvimento de software valoriza a granularidade, a leveza e a capacidade de compartilhar processos semelhantes entre várias aplicações. Trata-se de um componente indispensável para a otimização do desenvolvimento de aplicações para um modelo nativo em nuvem.**

**Fontes: DevMedia, RedHat, Frankel, Trigon Blue, Fowler.**