

## **Frameworks**

**Histórico de Frameworks**

**Frameworks**

**Histórico de Frameworks Web**

**Frameworks Web de CMS – Content Management System**

**Frameworks Web de Aplicações**

## Histórico de Frameworks

Os Sistemas de Gerenciamento de Informações - SGI são programas voltadas ao gerenciamento de informações, com grande aplicação em empresas, sendo compostos essencialmente de um banco de dados e de um programa que acessa e manipula as informações contidas neste banco de dados.

Os bancos de dados sofreram grandes evoluções nos últimos 10 anos com a consolidação dos Sistemas Gerenciados de Bancos de Dados SQL (SGBD-SQL). Isso trouxe ganhos significativos para os programadores, pois parte da lógica de programação pôde ser transferida do programa para o banco de dados, permitindo assim aumentar o desempenho e, principalmente, a confiabilidade dos dados. Os SGBD-SQL podem hoje ser encontrados nas mais diversas versões, comerciais ou gratuitas, de grande e pequeno porte, não havendo mais dúvidas quanto a sua utilidade ou funcionalidade.

Os programas por sua vez ainda são caracterizados, na grande maioria dos casos, por um trabalho quase "artesanal". Esta característica vem em parte da história de formação da maioria dos programadores que não utilizam bibliotecas, mas preferem ao invés disso desenvolver suas próprias; em parte pela necessidade de customização mencionada no parágrafo anterior; e em parte pelo grande "back-log" existente na área de desenvolvimento, ou seja, não há tempo para desenvolver nada mais "refinado" pois existe muito trabalho há fazer.

A década de 80 foi caracterizada pela grande discussão sobre qual seria a "melhor" linguagem. As comunidade discutia se C era melhor que Pascal, ou se a linguagem ADA seria, finalmente, a linguagem para resolver todos problemas ( como havia sido a promessa da PL/1 duas décadas antes ). Um ponto muito importante foi o surgimento de linguagens orientadas a objeto, que conceitualmente existiam desde o final da década de 60, mas que aparecerem para a comunidade através, inicialmente, do SMALLTALK-80. Este conceito viria anos mais tarde a revolucionar o desenvolvimento de sistemas, tendo como uma das primeiras "estrelas" a linguagem C++, criada ainda na década de 80.

A década de 90, já com a existência dos sistemas operacionais gráficos, em especial do Windows, se caracterizou pela consolidação de ambientes de desenvolvimento (IDE, do inglês Integrated Development Environment) e não mais pela discussão da linguagem. Além disso os SGBD-SQL se popularizaram juntamente com os microcomputadores, as redes e o conceito Cliente-Servidor, vendido na época como futuro substituto dos computadores de grande porte. Um conceito importante surgido nesta década foi o do Componente, em essência um objeto, disponibilizado em um determinado padrão que poderia ser usado dentro de qualquer linguagem, facilitando assim o desenvolvimento de sistemas. Padrões como ActiveX da Microsoft e VCL da Borland se tornaram cada vez mais populares, juntamente com os IDEs que

permitted sua utilização. Agora era possível desenvolver parte de um projeto em uma linguagem e integrar de forma simples em outra, tornando a escolha da linguagem algo menos importante. Falava-se, inclusive, que no futuro os profissionais de desenvolvimento seriam divididos em desenvolvedores de componentes e desenvolvedores de sistemas.

A década de 90 ainda nos trouxe a Internet que iniciou uma nova fase nas linguagens de programação e no desenvolvimento de sistemas, pois criava um novo paradigma: o desenvolvimento "web", de programas que seriam executadas em navegadores de Internet, em contraposição ao desenvolvimento "desktop" que cria programas para serem executados diretamente na máquina do usuário. Apareceu ainda a linguagem Java que com o conceito de "máquina virtual" abriu novos horizontes na programação multiplataforma, recebendo investimentos maciços de empresas como IBM e SUN. Em contrapartida a Microsoft desenvolveu o ambiente .NET, com idéias muito semelhantes a linguagem Java. Finalmente a década trouxe uma contribuição importante na área de Análise de Sistemas com a consolidação da linguagem de modelagem UML para especificação de sistemas, servindo como base para ferramentas de modelagem, inclusive para geração automática da estrutura de classes de determinada aplicação a partir de uma especificação.

Outro conceito muito interessante que se tornou popular na década de 90 foi o de sistemas distribuídos ou N-camadas de programa (N-Tier), em especial o de 3 camadas, que pregava a divisão do sistema em camadas de: dados, responsável pelo armazenamento dos dados, usualmente; regras de negócio, responsável pelo armazenamento e execução das rotinas relacionados ao processo que o sistema se propõe a controlar e interface, responsável pela interação como o usuário. Com a popularização do conceito de sistemas distribuídos estruturas de processamento remoto como COM+ e CORBA, se tornaram mais conhecidas e utilizadas, porém também mostraram toda a sua complexidade. Com a popularização da Internet e do protocolo http, surgiu o conceito de "Web Service" que permite distribuir aplicações de forma mais simples substituindo, em parte, o que as estruturas COM+ e CORBA faziam. O conceito de "Web Services" é um dos pontos abrangidos pelo Framework, pois através desta tecnologia pode-se integrar linguagens tão diferentes quanto Delphi e PHP, permitindo reaproveitar código escrito em qualquer uma delas.

Hoje a área de linguagens de programação e de desenvolvimento de sistemas vive um fase de intenso desenvolvimento. A presente década deve levar ao mundo da programação "web" os conceitos de ambientes integrados e componentes desenvolvidos nas últimas décadas para programação "desktop". Os primeiros exemplos disto são os ambientes Java, da Sun, e .NET da Microsoft, que são bem mais do que meras linguagens.

Apesar de todos estes avanços freqüentemente somos procurados por desenvolvedores ou empresas de desenvolvimento de sistemas, que tem necessidades específicas a serem atendidas em sistemas de clientes, mas que

não encontram soluções para problemas relacionados a integração de Componentes dentro de linguagens Orientadas a Objeto, lançando mão de soluções incompletas ou ineficientes. Com o avanço da Internet e o conseqüente aumento de horizonte das necessidades dos sistemas, percebemos um aumento também na demanda deste tipo de suporte.

## **Frameworks**

A necessidade de desenvolvimento de SGIs adaptados às necessidades dos clientes é uma necessidade cada vez mais flagrante em função do uso intensivo de sistemas informatizados nas empresas. A busca por uma vantagem competitiva em relação à concorrência obriga as empresas a desenvolver processos específicos que os diferenciem perante o cliente, e os sistemas, que em essência modelam estes processos dentro de um computador, devem ser igualmente customizados.

O uso de IDEs, Componentes e conceitos de Análise, Projeto e Programação Orientados a Objetos trouxeram ganhos significativos para área de desenvolvimento de sistemas. Entretanto, como pudemos perceber em mais de 15 anos de experiência, o desenvolvimento de um novo sistema repete em muito um desenvolvimento anterior, o que nos levou a buscar algo que eliminasse este retrabalho.

O conceito de Framework, como um conjunto de Componentes, Objetos e Técnicas que sirvam de base para as fases de Projeto e Implantação de um Sistema, nos parece a solução do problema. Quando um novo programa é iniciado, uma série de elementos deve ser definida, por exemplo:

- interface de acesso ao banco de dados;
- padrões de interface com o usuário ( componentes );
- modelo de segurança de acesso;
- padrões de interface com outros sistemas ( exportação/importação de dados );

Estes elementos são essencialmente genéricos, ou seja, não são relacionadas a função fim do sistema em desenvolvimento, que pode ser, por exemplo, um Controle de Estoque, mas sim a forma de implantação do sistema. Assim são algo que o usuário final nem tomará conhecimento. Cada parte do sistema do sistema já pode fazer uso dos milhares de componentes disponíveis no mercado, cada um com suas habilidades específicas, porém o desenvolvedor ainda precisa juntar estes vários componentes e transformá-los em um conjunto coerente, que sirva de base para o programa.

Pode ser feita uma analogia interessante e esclarecedora com a Indústria Automobilística. As primeiras indústrias produziam praticamente todas as peças do carro. A seguir veio o conceito de auto-peças, permitindo que a

indústria se concentrasse no essencial, ou seja, fazer um automóvel e não se preocupar em como fazer uma lâmpada ou então um pneu. O conceito mais atual é o de integração de fornecedores na mesma planta, ou seja, ao invés de auto-peças já são fornecidos módulos prontos (Ex: fábrica da Daimler-Chrysler, em Juiz de Fora - MG). Atualmente a indústria de sistemas já utiliza o conceito de componentes ( auto-peças ), mas a cada novo programa recria uma nova estrutura ( motor, chassis, etc ) bastante similar entre os diversos sistemas, porém o que cliente efetivamente percebe é o projeto externo do carro, ou seja, seu design, acabamento, preço, etc.

O Framework busca fornecer esta estrutura, permitindo ao desenvolvedor se concentrar nos aspectos específicos da aplicação, que realmente interessam ao usuário final, que, em última instância, está pagando pelo sistema. Um Framework pode ser visto como um conjunto de camadas de software que se complementam, mas não se entrelaçam (ver Figura 1). A primeira camada é a **Camada de Dados**, responsável por armazenar e acessar os dados, normalmente representada por um SGBD-SQL e uma API de acesso a dados fornecida pelo Sistema Operacional ou pela linguagem que está sendo utilizada. A seguir vem a **Camada de Objetos**, que contém os códigos que executam o processo que o sistema em questão busca automatizar. Esta camada pode ser dividida em 2 sub-camadas: a Camada de Objetos de Sistema, que contém os objetos responsáveis por controlar o próprio sistema como, por exemplo, segurança ou relatórios; e a **Camada de Objetos de Negócio**, que contém os objetos criados especificamente para modelar o processo automatizado pelo sistema. Finalmente temos a Camada de Interface, que faz a interação com o usuário, ou mesmo com outro programa, como ocorre com "Web Services".

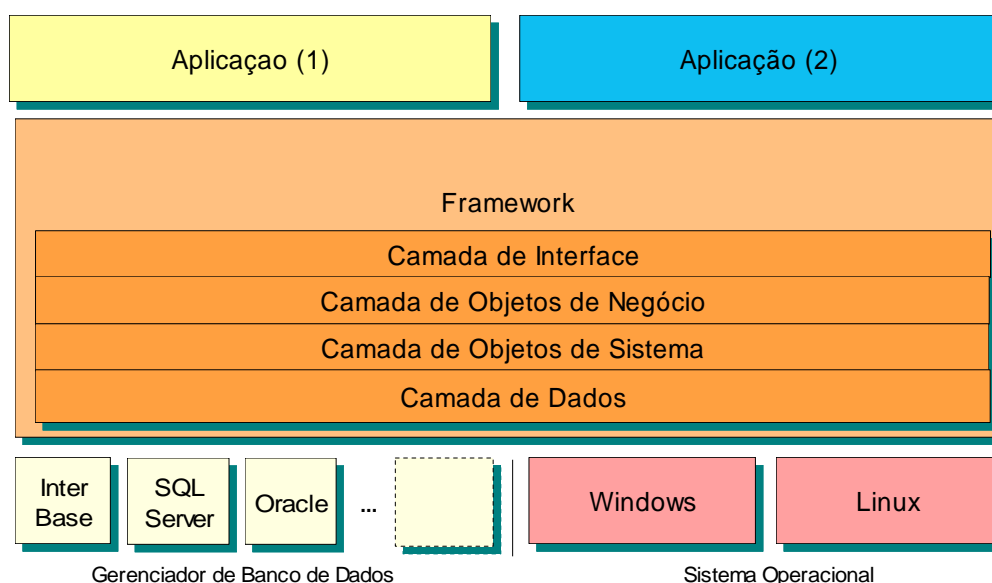


Figura 1– Modelo de Camadas do Framework

Uma consequência interessante da criação de um Framework de SGIs é a possibilidade de criar extensões específicas ( Camada de Objetos de Negócio ) para, por exemplo, Sistemas de Gestão Empresarial, onde ao invés de tratar aspectos estruturais do sistema, seriam criados objetos para tratar, por exemplo, de Controle de Estoque. Isto abre perspectivas interessantes sobre novos modelos de negócio para as empresas de desenvolvimento.

Existe ainda o conceito de **Framework de Produto** e de **Framework de Produção**. O primeiro representa os Componentes, Objetos e Técnicas de Programação descritas acima. O segundo representa o conjunto de Componentes, Técnicas e Ferramentas que permitem colocar um sistema em produção, abrangendo:

- controle de versões (Ex: CVS);
- criação de help de programa;
- criação de instalador de programa;
- proteção de cópias;

O segundo é o objeto do presente projeto, pois, o Framework de Produção é, usualmente, composto de pacotes amplamente disponíveis no mercado quer na forma de licenças comerciais quer como open-source, escolhidos em função de necessidades específicas do desenvolvedor.

## Histórico de Frameworks Web

O Ambiente Web é muito rico em Frameworks de desenvolvimento motivado pela complexidade de desenvolvimento de grandes aplicações, pela disponibilidade de linguagem interpretadas e de código-fonte aberto o que facilita a distribuição e soluções e pela grande "efervescência" no desenvolvimento deste tipo de aplicações nos últimos anos.

Um grande exemplo de Frameworks para desenvolvimento Web são os softwares de CMS – Content Management Systems, ou, Sistema de Gerenciamento de Conteúdo, voltadas a criação de sites corporativos. São formados em essência por um conjunto de código e banco de dados que implementa os módulos mais comuns existentes em grandes portais corporativos como, por exemplo: controle de acesso, notícias, agenda, grupos de discussão, portal de conteúdo, etc.

Um grande exemplo deste tipo de sistema é o PHPNuke, criado originalmente na Itália, em linguagem PHP, do qual derivaram algumas centenas de outros sistemas em diversas linguagens. A maioria destes projetos são de código-fonte aberto mantidos por grupos de desenvolvedores em grandes sites de open-source como o SourceForge ( <http://sourceforge.net> ).

## Frameworks Web de CMS – Content Management System

Um grande problema com projetos open-source é garantir sua continuidade e qualidade no código e documentação. Assim existem vários sites que mostram características dos principais Frameworks de CMS e de outros tipos de aplicação, tentando ajudar o usuário a escolher alguma aplicação mais adequada a solução do seu problema.

### **The CMS Matrix - cmsmatrix.org - The Content Management Comparison Tool**

<http://www.cmsmatrix.org>

Site que compara dezenas de frameworks CMS na forma de uma matriz, onde podemos escolher vários pacotes e comparar suas características

### **OpenSourceCMS**

<http://www.opensourcecms.com>

Site com dezenas de frameworks CMS

### **JOOMLA**

<http://www.joomla.org>

Um framework baseado no CMS MAMBO e escrito em linguagem PHP

**XOOPS**<http://www.xoops.org>

Um framework orientado a objetos, baseado no CMS PHP-Nuke e escrito em linguagem PHP.

**DotNetNuke**<http://www.dotnetnuke.com>

Um framework para ambiente .NET. e escrito em linguagem VB.NET

**Zope**<http://www.zope.org>

Um frameworks escrito em linguagem Phyton que funciona como servidor para criar sistemas de gerenciamento de conteúdo, Intranets, portais e aplicações customizadas.

**Frameworks Web de Aplicações**

Além dos Frameworks voltados a criação de aplicações específicas, existem aqueles que fornecem apenas a infra-estruturas necessária a criar uma aplicação Web, como, por exemplo, em Sistema de Gerenciamento de Informações, ao invés de definir todo um ambiente de aplicação. Como o desenvolvimento de aplicações Web é bem mais complexo que o desenvolvimento de aplicações Desktop dado a grande quantidade de tecnologias envolvidas, este tipo de Framework é extremamente interessante.

Existe todo tipo implementação de Frameworks de Aplicações, mas usualmente encontramos as seguintes características:

- implementação do modelo MVC
- implementação de um modelo da camada de objetos de negócio
- implementação de uma camada de acesso a dados, ou utilização de alguma já existente
- temas de telas
- segurança de acesso
- componentes gráficos de tela, para os quais existem padrões apenas em ambiente .NET e aos poucos em Java
- implementação de camada de persistência de acesso a dados

**Php Mvc Frameworks - Web Application Component Toolkit**[http://www.phpwact.org/php/mvc\\_frameworks](http://www.phpwact.org/php/mvc_frameworks)

Site com dezenas de Frameworks de Aplicação em PHP.

**Ajax Patterns**<http://ajaxpatterns.org>

Site com dezenas de Frameworks para utilização da tecnologia AJAX

### **MIOLO**

<http://www.miole.org.br>

<http://cursosiga.ufjf.br/index.php/tutorial>

Framework em PHP criado pela UNISINOS ( versão 1.0 ) e UFJF ( versão 2.0 ) que tem todas a características interessantes de um Framework de Aplicações.

### **PRADO PHP Framework**

<http://www.pradosoft.com>

Framework em PHP com ênfase em componentes