Construindo Aplicações Web 2.0 que Funcionam

Eduardo Santos

eduardo.edusantos@gmail.com
eduardo.santos@planejamento.gov.br
www.softwarepublico.gov.br

Histórico

Por que existe a Internet?

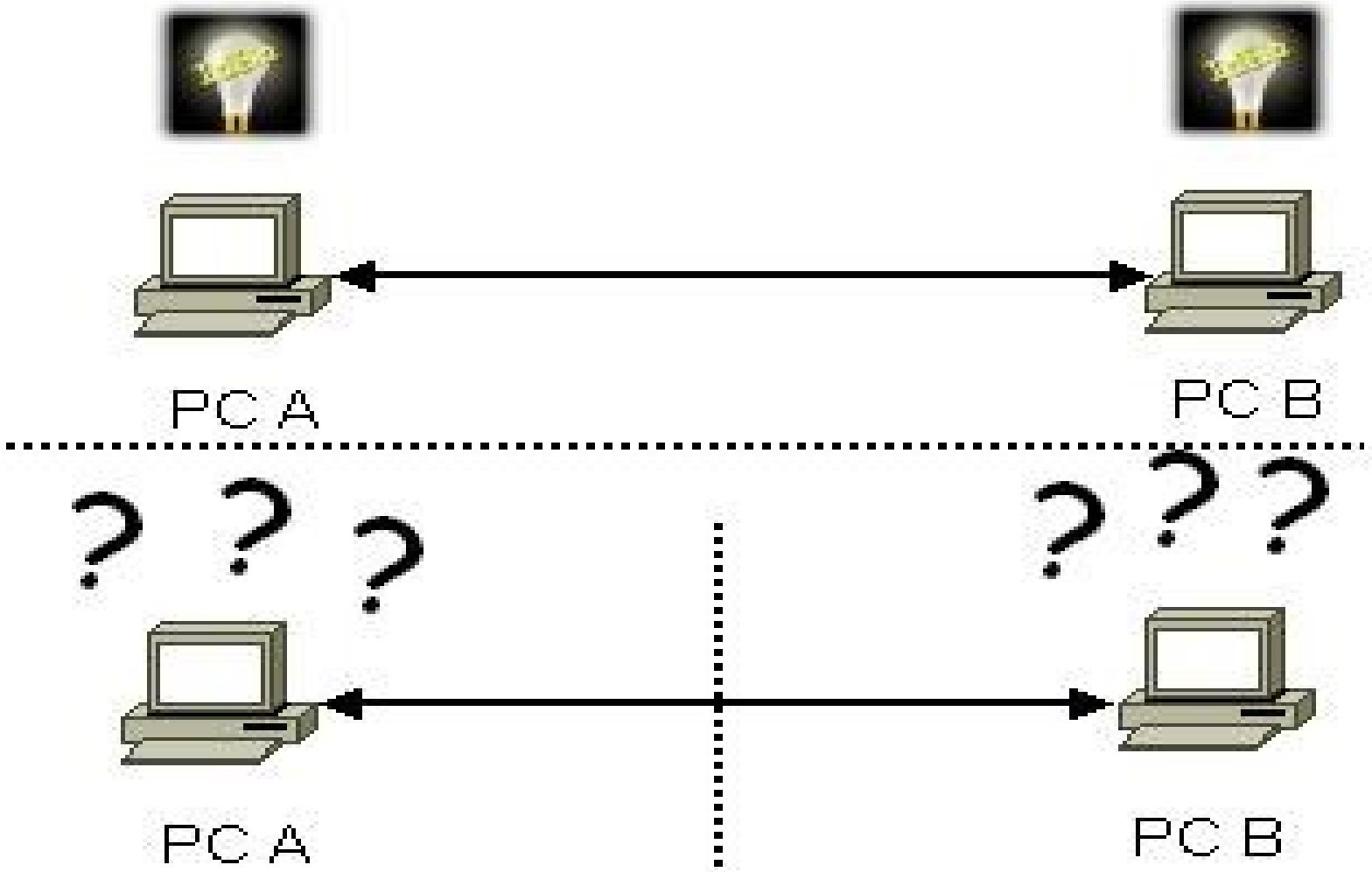
- Por que existe a internet? Por que conectar todas as pessoas do mundo?
- No início existiam duas categorias de sites:
  - Publicar conteúdo (*Web Publishing*): HTML estático;
  - Aplicações para a rede (*Web Based Applications*)

Web Publishing

- Podem ser chamadas de revistas eletrônicas
  - Ex.: Catálogo de produtos de uma companhia
- A tecnologia por trás é apenas um detalhe
- Principal pergunta: por que alguém acessaria o seu site?

Tecnologia de Comunicação

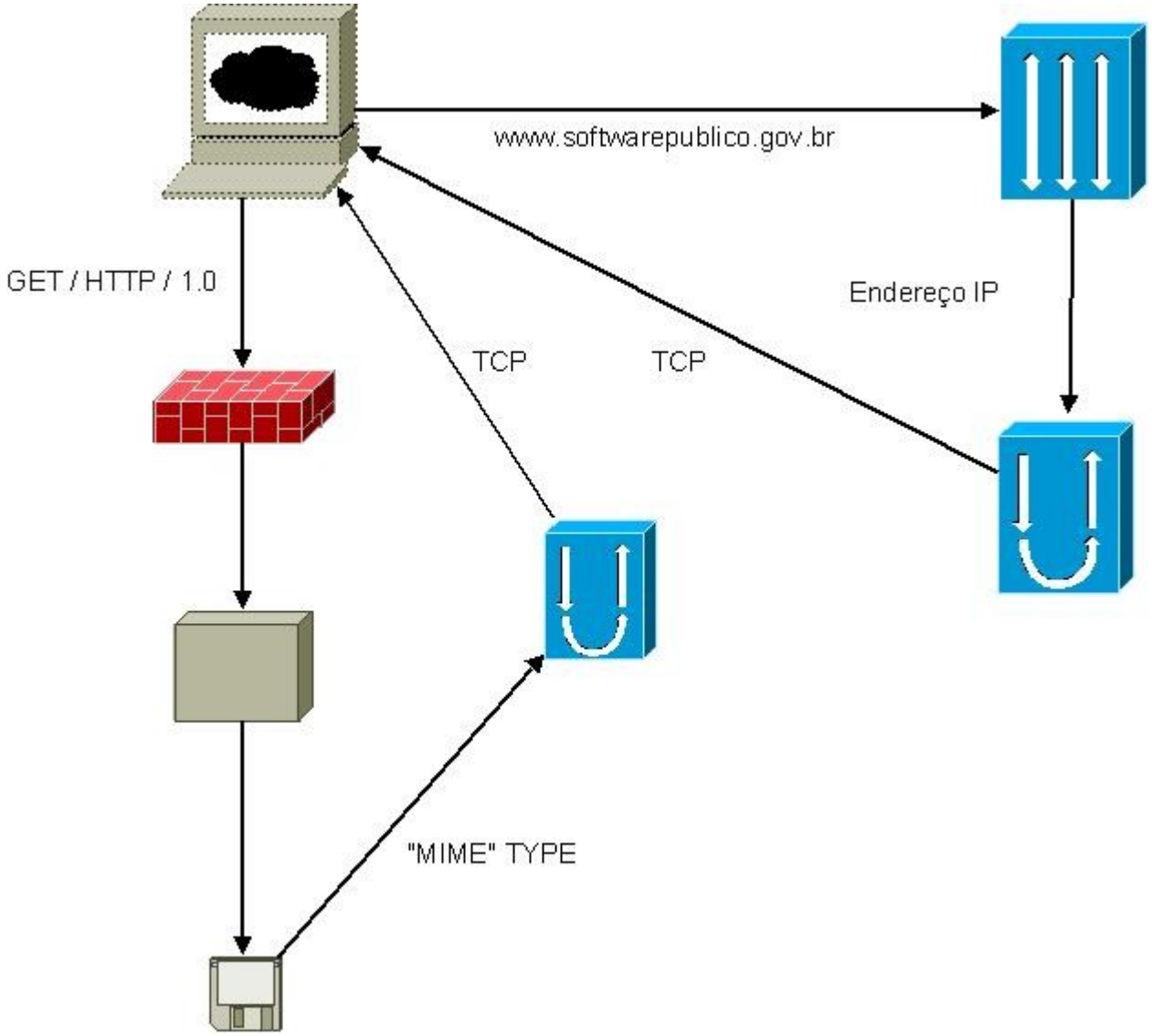
Estrutura de Comunicação



Protocolo HTTP

- *Stateless* (não guarda “passado”): cada chamada gera uma requisição
- Anônimo: não sabe exatamente “quem” está conectando.

Protocolo HTTP



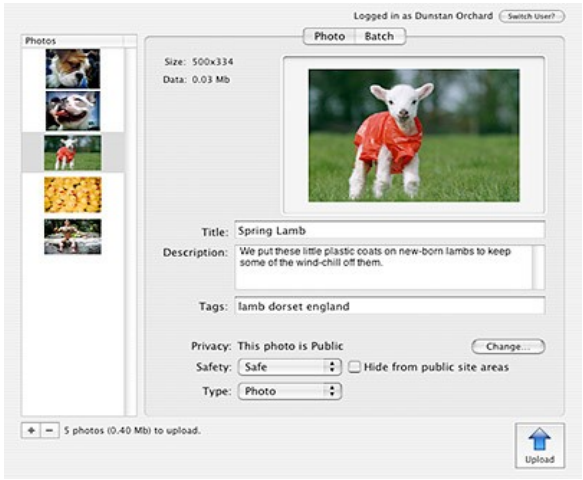
Protocolo HTTP

- Quando a conexão terminar, a comunicação acaba
- Nenhuma informação é armazenada

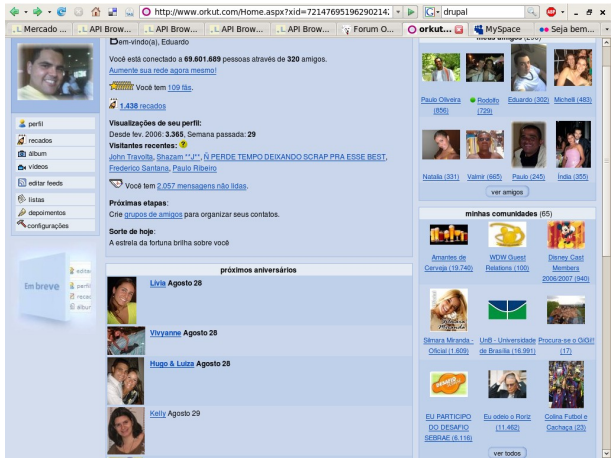
Web 2.0

A Nova Internet

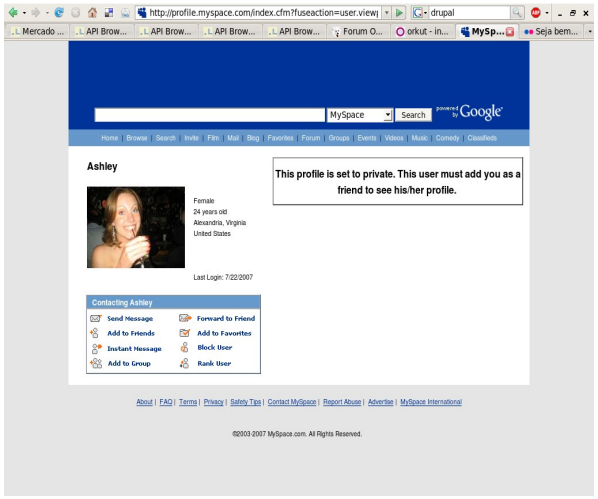
- *"Web 2.0 é uma série de aplicações que propiciam e potencializam a formação de redes sociais digitais" - Abel Reis*



Flickr



Orkut

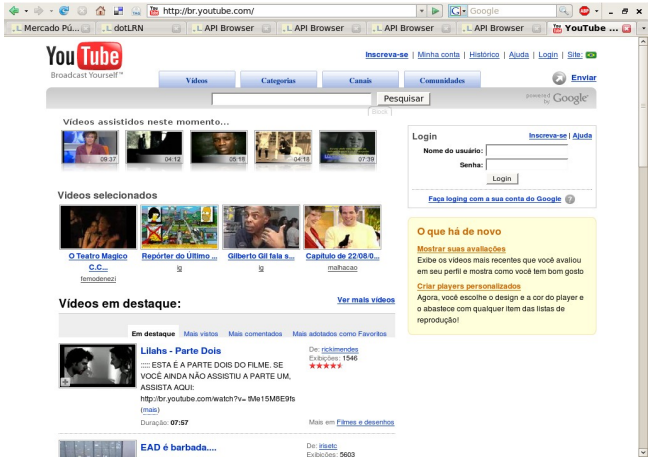


MySpace

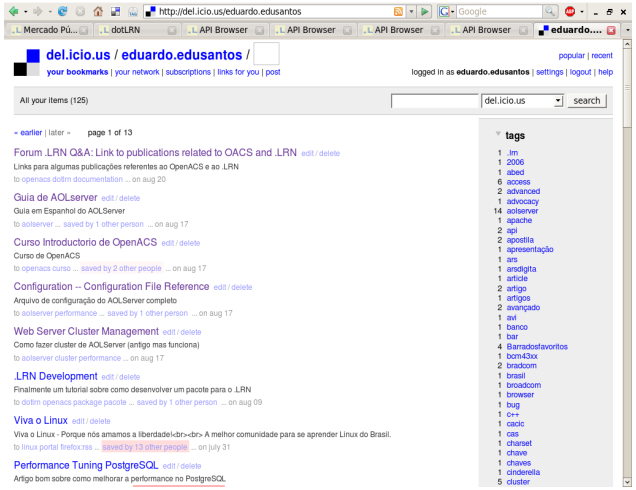
Exemplos Web 2.0

Blogs

Wikipedia



You Tube



del.icio.us

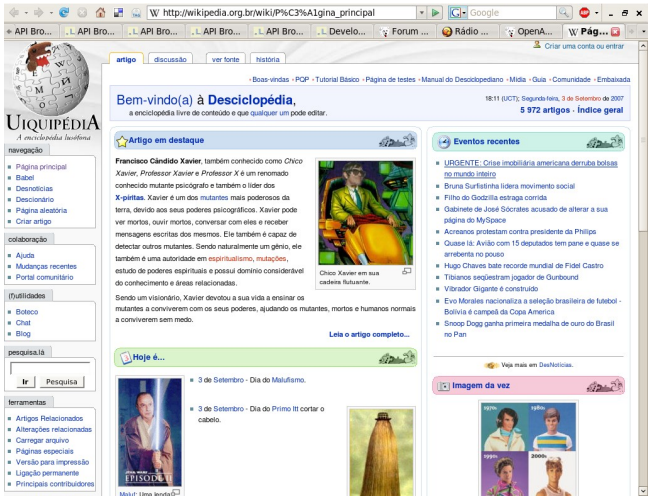
Exemplos Web 2.0

Blogs

- Primeira Revolução:
  - Leva em conta a opinião dos usuários
  - Contato direto com a opinião do consumidor

Exemplos Web 2.0

Blogs

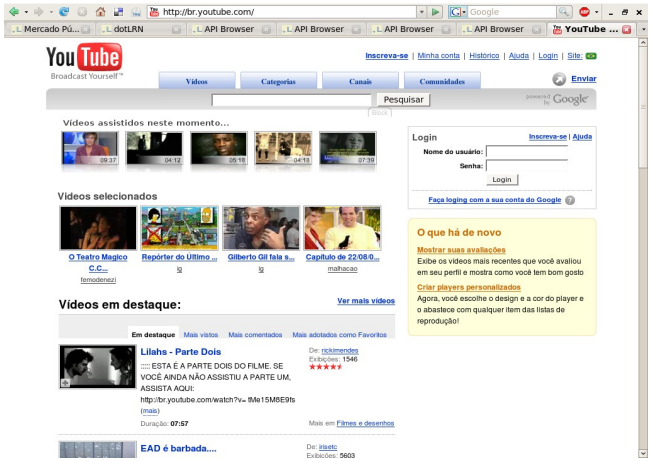


Wikipedia

- Primeira Revolução:
  - Leva em conta a opinião dos usuários
  - Contato direto com a opinião do consumidor
- Segunda Revolução:
  - Leva em conta o conteúdo produzido pelo usuário
  - Capacidade de acúmulo de informações tende ao infinito

Exemplos Web 2.0

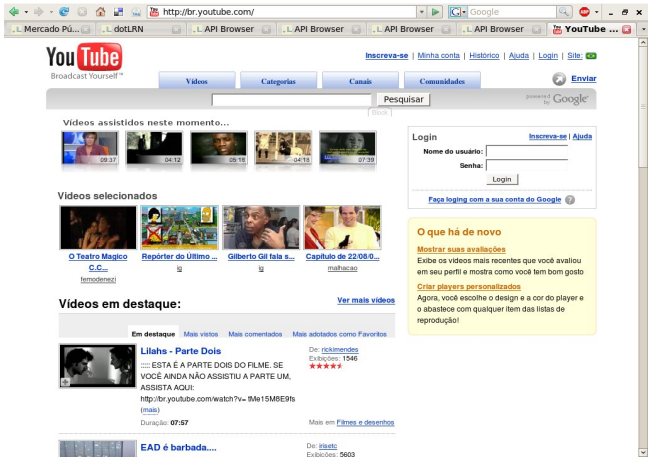
- Terceira Revolução:
  - Conteúdo multimídia produzido pelo usuário
  - Produção não segue nenhum padrão de mercado ou modelo



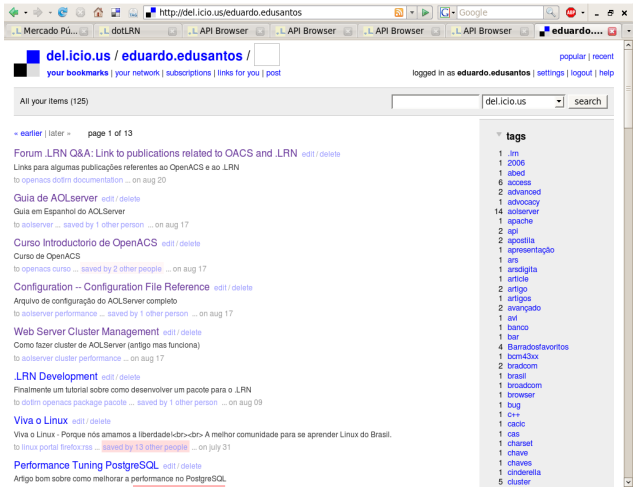
You Tube

Exemplos Web 2.0

- Terceira Revolução:
  - Conteúdo multimídia produzido pelo usuário
  - Produção não segue nenhum padrão de mercado ou modelo
- Quarta Revolução:
  - Leva em conta os sites que o usuário visita
  - Organização de conteúdo de acordo com a preferência



You Tube



del.icio.us

Qual a revolução trazida pelo Google?

Qual a revolução trazida pelo Google?

Modelo de negócios bastante lucrativo, com foco no conteúdo produzido pelo usuário

Web 2.0

Nesse contexto, como desenvolver aplicações para a internet?

Engenharia Básica

Guardar informação do usuário

- Sabemos que o protocolo HTTP não guarda histórico
- Para que haja interação, é necessário armazenar informações
- Questão: como armazenar essa informação?

Guardar informação do usuário

- Idéia: passar uma informação extra em cada URL visitada.

http://www.amazon.com/exec/obidos/ASIN/1588750019/103-9609966-70894

Armazenamento na URL

- Idéia: passar uma informação extra em cada URL visitada.

http://www.amazon.com/exec/obidos/ASIN/1588750019/103-9609966-70894

- Problemas:
  - O que acontece com endereços muito grandes?
  - Como organizar?
  - URLs maiores que 255 bytes são problema, principalmente no Internet Explorer.

Cookies

"Cookies são um mecanismo geral pelo qual as conexões do lado do servidor (como scripts CGI) podem utilizar tanto para armazenar quando para recuperar informações no lado do cliente na conexão."

Philip Greenspun

Cookies

- Funcionamento:

  Set-Cookie: cart_contents=1588750019; path=/

- Se o browser não for fechado, a seguinte informação será fornecida junto com o cabeçalho:

  Cookie: cart_contents=1588750019

Cookies

- Problemas
  - O máximo que o browser pode armazenar são 20 cookies, cada um com no máximo 4 KB.
  - A cada página carregada, a informação é enviada novamente.
  - O que acontece se o usuário acessa o mesmo site de outra máquina?
  - Problemas de privacidade.

Armazenamento no Servidor

- Qual a melhor maneira de armazenar a informação?
  - Planilhas eletrônicas?
  - Arquivos de texto?
    - O que acontece se vários usuários tentam abrir um arquivo ao mesmo tempo?
    - E se eles tentarem gravar o arquivo?

Armazenamento no Servidor

- *Database Management System (DBMS)*
  - Resolve o problema de concorrência de transações.
- Como avaliar um *DBMS*?

ACID Test

- *Atomicity*
  - Os resultados da execução de uma transação ou são todos gravados (*commit*) ou são todos desfeitos (*roll back*).
- *Consistency*
  - Um banco de dados só pode passar de um estado válido para outro também válido. Uma transação só é legal se obedece as restrições de integridade do usuário e, se uma restrição não pode ser satisfeita, a transação é desfeita.

ACID Test

- *Isolation*
  - Os resultados de uma transação são invisíveis às outras transações até que a primeira esteja completa.
- *Durability*
  - Uma vez gravados (*commited*), os resultados das transações são permanentes e sobrevivem a falhas de hardware e software no sistema.

Por que RDBMS?

- Linguagem declarativa SQL.
- Isolamento de partes importantes contra erros do programador.
  - Não permite alterações arbitrárias no *data set* por outros programas;
  - Tudo se resume a INSERT, UPDATE e SELECT.
- Performance

Quatro Passos para Você

1. Faça a modelagem de dados. O que vai armazenar e como representar?
2. Desenvolva uma série de scripts para criar o banco (SQL).
3. Desenhe o fluxo de páginas. Como será a interação do usuário?
4. Implemente as páginas individualmente. Como será o layout de cada uma?

Escolhendo um ambiente de desenvolvimento

Escolhendo um *RDBMS*

- Microsoft SQL Server
  - Passo!

Escolhendo um *RDBMS*

- Oracle
  - Banco de dados que melhor lida com transações concorrentes;
  - Performance imbatível.
- PostgreSQL
  - Possui um sistema bastante parecido para lidar com transações concorrentes;
  - Já está bem avançado na parte de tratamento de queries.

Escolhendo um *RDBMS*

- MySQL
  - Fácil administração;
  - Bastante rápidos para operações de SELECT.
- Seu banco
  - A política de banco de dados deve obedecer à preferência de sua instituição.

Escolhendo a linguagem

- Na maior parte do tempo, sua aplicação estará somente colocando resultados das consultas na tela.
- As procedurais são preferíveis por serem mais simples.
- Consultas com maior capacidade de abstração podem ser tratadas dentro do banco de dados.

Escolhendo o ambiente

- O primeiro critério deve ser:

  uma URL = um arquivo

- Ambientes em que isso acontece:
  - Perl CGI;
  - Microsoft Active Server Pages (ASP);
  - Java Server Pages (JSP);
  - AOLServer ADP templates and .tcl scripts;
  - PHP?

Escolhendo o ambiente

- Modularidade e reusabilidade de código.
- Filtros.
- URLs abstratas:
  - É importante para o usuário mostrar a extensão do arquivo?
  - O que acontece com as marcações do usuário se mudo o ambiente?
  - Como tratar a execução dos arquivos?

Escolhendo o ambiente

- *Request processor*:
  - Existe algum arquivo .jsp? Se tiver, execute.
  - A requisição vem de um celular. Se vier, existe algum arquivo .mobile no sistema?
  - Existem algum arquivo HTML?
  - Existem imagens?

Modularidade

- Estudos dizem que 80% do trabalho são gastos refazendo o que já foi feito.
- Qual a melhor maneira de reutilizar o código?
- A idéia é que cada aplicação funcione como um serviço diferente, mas mantendo um padrão com o sistema.

Modularidade

- Agrupamento de código
  - Manter tabelas separadas mas com algum nível de relacionamento entre cada aplicação;
  - Construir procedimentos armazenados para o banco de dados;
  - Criar bibliotecas com procedimentos que sejam compartilhados por mais de uma página;
  - Construir as páginas individuais;
  - Escrever documentação para o seu módulo.

Modularidade

- Requisitos
  - Estrutura de arquivos comum a todos os módulos;
  - Arquivos de configuração individual para cada módulo;
  - Facilidade em integrar o módulo com a estrutura de dados da aplicação.

Modularidade

- Interoperabilidade (utilização de APIs);
- Parâmetros de configuração;
- Interface de administração de módulos.

Frameworks Disponíveis

- OpenACS (TCL)
- Plone (PYTHON)
- Zend (PHP)
- Ruby on Rails (RUBY)
- Hibernate / Struts (JAVA)

Contato

Eduardo Santos

eduardo.edusantos@gmail.com
eduardo.santos@planejamento.gov.br

www.softwarepublico.gov.br