

|  |    |
|--|----|
| Pré-requisitos do Tutorial .....                             | 2  |
| Objetivos do Tutorial .....                                  | 2  |
| Fundamentos do Stress-test .....                             | 3  |
| Olá Mundo Stress-test .....                                  | 5  |
| Monitoração.....   | 5  |
| Arquitetura .....  | 5  |
| JMeter .....   | 6  |
| Metodologia para Teste de Capacidade.....                    | 9  |
| Formulário de Análise de Requisitos não-funcionais .....     | 10 |
| Perguntas dissertativas.....                                 | 10 |
| Distribuição de utilização da solução por use-case .....     | 11 |
| Cálculo de Volume de Dados por Entidade .....                | 11 |
| Identificação de Cenários Atípicos .....                     | 12 |
| Tabela de Mapa de Riscos .....                               | 13 |
| Melhores práticas na captura de requisitos para testes ..... | 14 |
| Planejamento de Testes de Capacidade .....                   | 15 |
| Exemplo de Proposta de Planejamento de Testes .....          | 16 |
| Ferramenta JMeter .....                                      | 18 |
| Download e instalação .....                                  | 18 |
| Inicializando o JMeter .....                                 | 19 |
| Criando um Plano de Testes.....                              | 20 |
| Elementos JMeter .....                                       | 21 |
| Utilizando Elementos Básicos .....                           | 22 |
| Elemento Test Plan.....                                      | 23 |
| Config Element.....  | 25 |
| Thread Group .....   | 27 |
| Sampler .....  | 29 |
| Listeners.....   | 32 |
| Logic Controller .....                                       | 35 |
| Elementos Avançados .....                                    | 38 |
| Assertions.....  | 38 |
| Pre Processors.....  | 41 |
| Post Processors .....  | 44 |
| Timers .....   | 45 |
| Monitoração de Ambiente de Testes .....                      | 47 |
| Protocolo SNMP .....   | 48 |
| Agente SNMP .....  | 50 |

|  |    |
|--|----|
| Gerente SNMP .....                                       | 50 |
| Management Information Base .....                        | 50 |
| Operações SNMP .....                                     | 52 |
| Segurança e características técnicas do SNMP .....       | 52 |
| Ferramentas Comerciais de Monitoração .....              | 53 |
| Ferramentas não-comerciais de Monitoração .....          | 54 |
| Plano de Capacidade e Relatório de Conclusões .....      | 57 |
| Análise de Requisitos não-funcionais.....                | 58 |
| Perguntas .....  | 58 |
| Distribuição de utilização da solução por use-case ..... | 59 |
| Cálculo de Volume de Dados por Entidade .....            | 59 |
| Documentação do Ambiente Físico.....                     | 60 |
| Documentação do Ambiente Lógico.....                     | 61 |
| Detalhes do Plano de Stress-test .....                   | 62 |
| Informações Coletadas na Monitoração do Ambiente .....   | 65 |
| Relatório de Conclusões.....                             | 66 |
| Plano de Capacidade e Recomendações .....                | 68 |

### Pré-requisitos do Tutorial

- Lógica de programação;
- Orientação a objetos;
- Conhecer plenamente a linguagem Java;
- Conhecimento básico de XML;
- Banco de dados relacional;
- UML;
- Computação distribuída;
- TCP/IP
- Sockets, RMI e CORBA (nível básico)

### Objetivos do Tutorial

Aprender técnicas e metodologia para captura de requisitos não-funcionais além de abordar tecnicamente a ferramenta Apache JMeter.

## Fundamentos do Stress-test

O principal objetivo de executarmos testes de stress em soluções é para assegurar que a arquitetura desenvolvida para atender a solução realmente consegue responder a quantidade de usuários previstos para acessar o aplicativo.

Para executar um teste de stress temos diversas tarefas para cumprir, devemos seguir um procedimento ou metodologia adequada para o cenário em questão. Atualmente não contamos com muitos padrões na indústria. O que temos são tecnologias isoladas que se destacam como SNMP para monitoração, JMeter como ferramenta open-source de stress, entre outros.

Podemos dividir o teste de stress em cinco partes:

### 1. Metodologia

Prevê forma de captar dados e requisitos não-funcionais como número máximo de usuários simultâneos, curva de crescimento para próximos dois anos, distribuição dos acessos durante o dia, distribuição dos acessos durante a semana entre outros servem como dados de entrada para conhecer os resultados esperados pelo aplicativo.

Muitos destes dados são capturados antes mesmo de iniciar o desenvolvimento da solução e são vitais para a escolha da arquitetura. No momento do teste de stress devemos confirmar se as decisões e escolha de componentes atingiu os objetivos não-funcionais.

Sua equipe deve prever uma atenção especial para estes dados na metodologia utilizada para captura de requisitos.

### 2. Planejamento

Escolher processos para o teste, preencher a base de dados com amostras bem próximas dos dados reais, projetar a quantidade de dados e crescimento de base de dados, consumo de banda, entre outros podem ser cruciais no sucesso final de um projeto.

Conhecer de perto as necessidades de tráfego e armazenamento de dados, ajuda nas previsões e planejamento de testes. Tecnicamente montamos scripts em endpoints de stress disparamos este script de forma cíclica até atingir o resultado de “bombardeio” esperado.

### 3. Monitoração

De nada vale um teste de stress se não conhecermos o quanto determinada quantidade de usuários está consumindo do ambiente. Os principais dados que devemos observar nos componentes participantes da solução são:

- Consumo de memória dos servidores;
- Consumo de CPU(s) dos servidores;
- I/O de rede entre container e banco de dados;
- I/O de rede entre web e container de EJBs
- I/O de disco em todos servidores;
- Swap de memória;

### 4. Plano de capacidade

Com a coleta dos dados na etapa de monitoração Vs. número de acessos projetados, podemos traçar um plano de capacidade e entender a curva de escalabilidade da solução assim como planejar novas aquisições e upgrades a medida que o software vai sendo utilizado.

### 5. Conclusão

O relatório de conclusão vai documentar para o cliente da solução a real capacidade do software desenvolvido. No relatório você deve anexar principais amostras de dados de monitoração.

Vamos estudar detalhadamente cada passo do stress-test e no término deste módulo você estará apto a desenvolver um aplicativo, desde a arquitetura até o stress-test.

## **Olá Mundo Stress-test**

Para entendermos os principais conceitos de monitoração e planejamento de testes vamos desenvolver um teste rápido e simples utilizando monitoração básica e a ferramenta JMeter.

### **Monitoração**

Para monitorar o consumo do servidor Linux, utilizaremos o seguinte comando:

```
vmstat 1 > resultado.txt
```

vmstat = comando que apresenta consumo de CPU, memória e I/O

1 = atualização de 1 em 1 segundo

> resultado.txt = todo redirecionamento para este arquivo

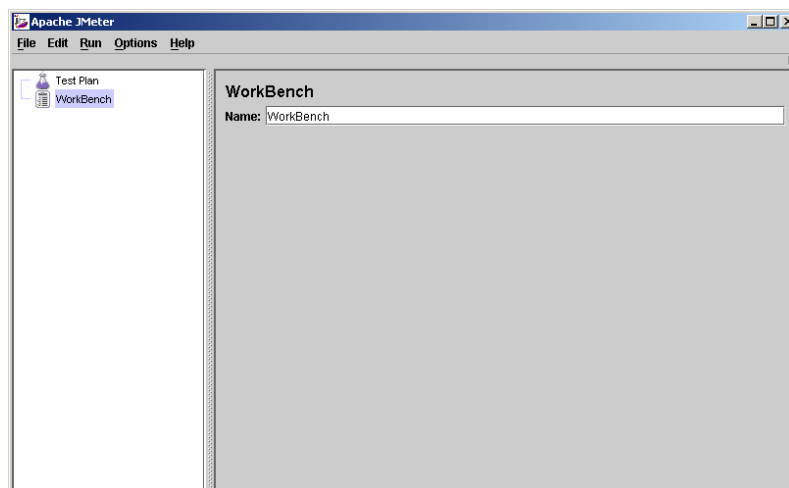
### **Arquitetura**

Vamos trabalhar com a arquitetura do laboratório 4 do módulo 1 (AA1 – lab04-sol) e vamos utilizar a arquitetura do laboratório 13 do módulo 3 (AA3 – lab13).

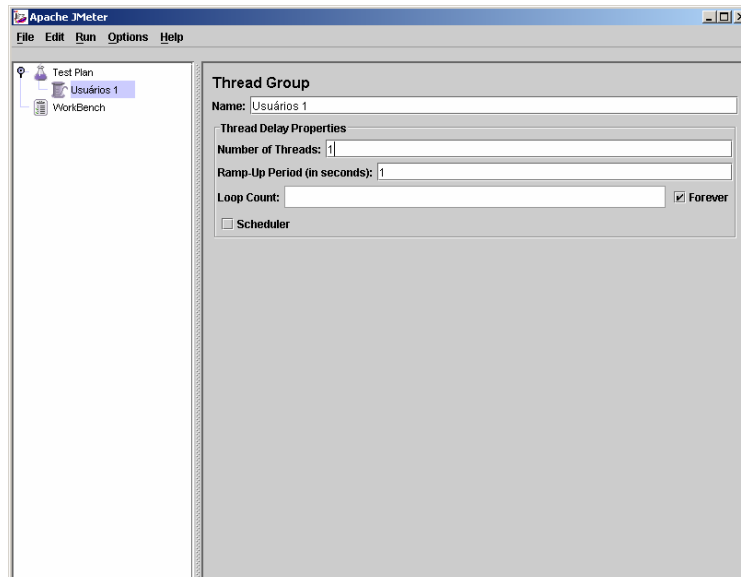
A princípio vamos manter todos os containers na mesma máquina: jboss, mysql e ferramenta de stress. A medida que você for progredindo desacople a ferramenta de stress (JMeter) da máquina do usuário.

## JMeter

1. Faça instalação do JMeter descompactando seu arquivo ZIP (informe-se com o instrutor sobre a URL)
2. Execute o script jmeter no diretório bin;

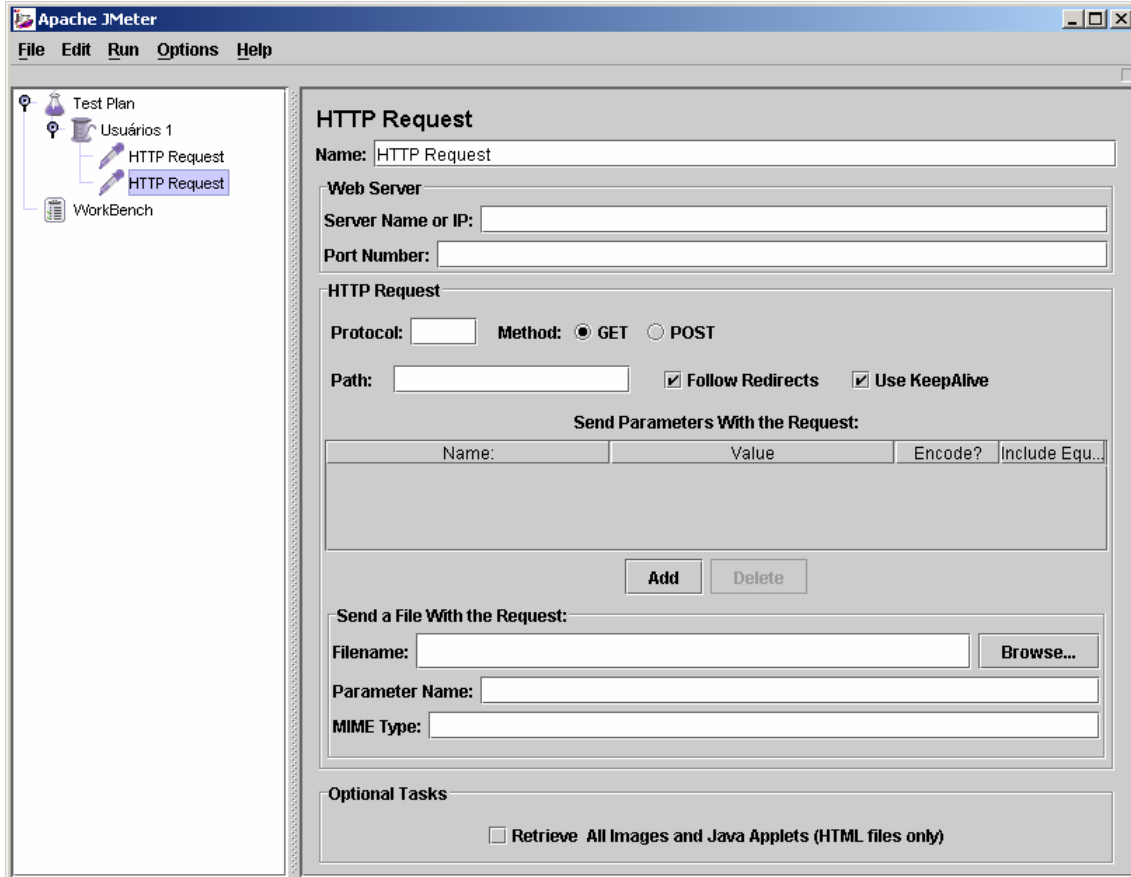


3. Clique o botão direito em Test Plan -> Add -> Thread Group



4. Configure Name para Usuários 1, Número de Threads para 10, Ramp-up Period para 3 e Loop Count para 10.

5. Clique o botão direito na Thread Group recém nomeada para Usuários 1 e selecione: Add -> Sampler -> HTTP Request



The screenshot shows the Apache JMeter interface with the 'HTTP Request' configuration dialog open. The left sidebar shows a test plan with a thread group named 'Usuários 1' containing two 'HTTP Request' samplers. The main panel is titled 'HTTP Request' and contains the following fields and options:

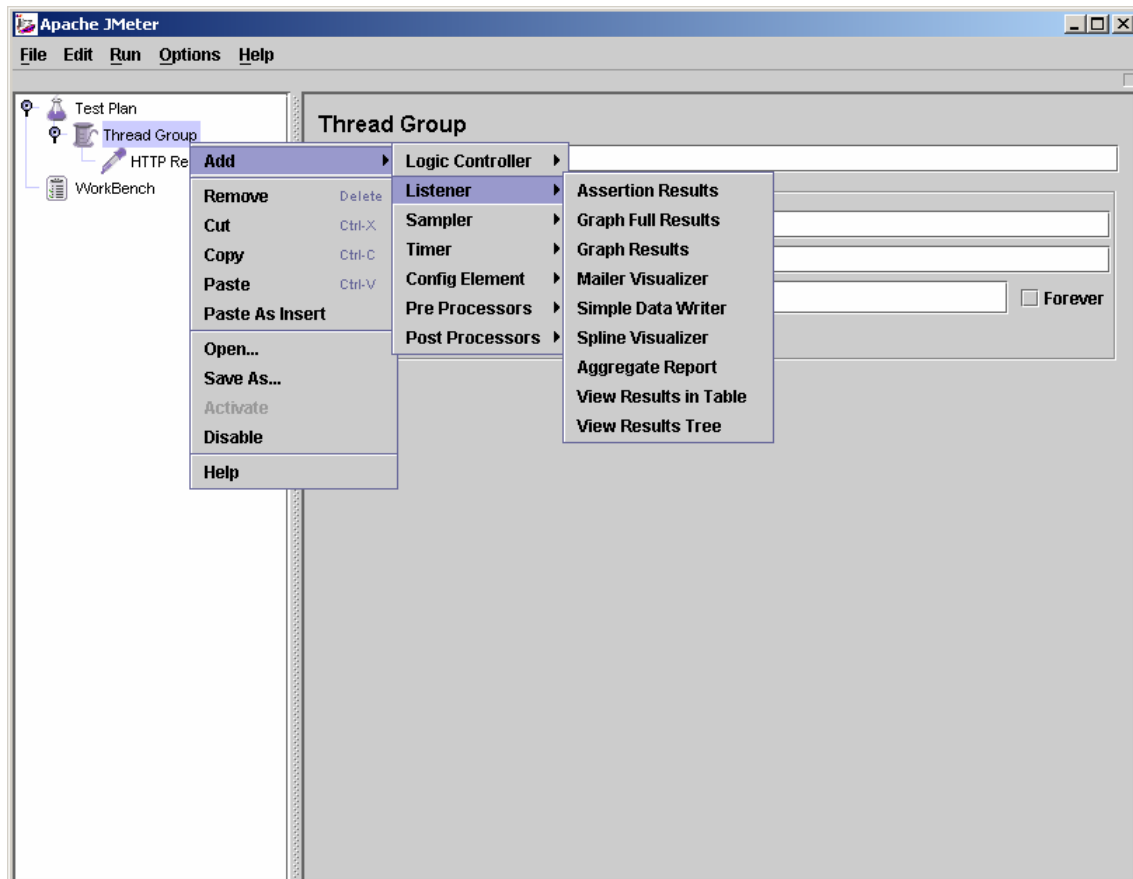
- Name:** HTTP Request
- Web Server:**
  - Server Name or IP:** [Empty field]
  - Port Number:** [Empty field]
- HTTP Request:**
  - Protocol:** [Empty field]
  - Method:** ☒ GET ☐ POST
  - Path:** [Empty field] ☒ Follow Redirects ☒ Use KeepAlive
- Send Parameters With the Request:**

| Name: | Value | Encode? | Include Equ... |
|-------|-------|---------|----------------|
|       |       |         |                |

Add Delete
- Send a File With the Request:**
  - Filename:** [Empty field] Browse...
  - Parameter Name:** [Empty field]
  - MIME Type:** [Empty field]
- Optional Tasks:**
  - ☐ Retrieve All Images and Java Applets (HTML files only)

Preencha os dados colocando uma URL válida que faça o acesso ao command que lista os cursos.

6. Clique novamente o botão direito na Thread Group Usuários 1 e selecione Add -> Listener -> Graph Full Results, Graph Results, Simple Data Writer e mais os que você desejar. Estes componentes são visualizadores dos resultados.

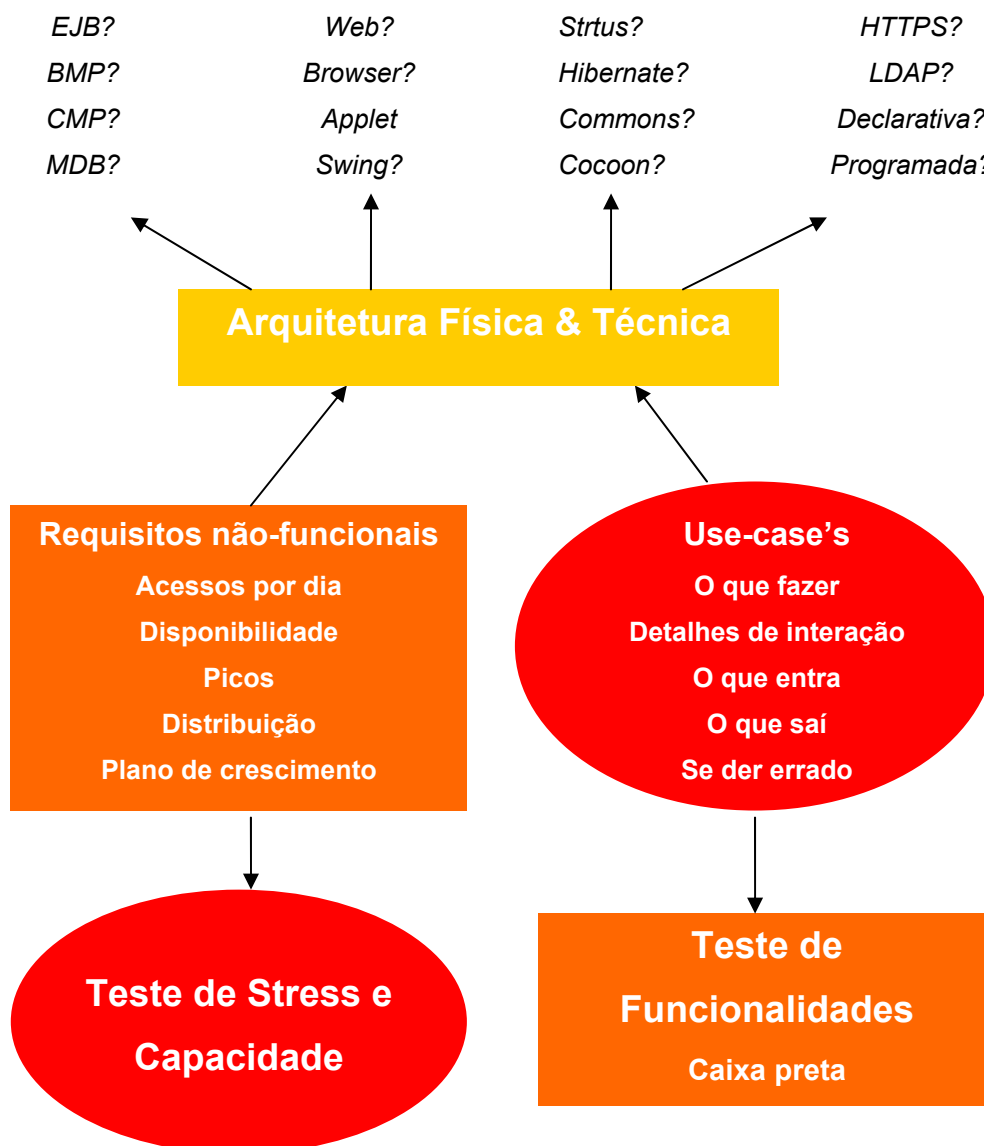


7. Acione o comando de monitoração vmstat e inicie os testes. Assim que finalizar pare o comando de monitoração com control + c, analise o consumo de CPU versus o resultado do JMeter.



## Metodologia para Teste de Capacidade

Devemos ficar atentos durante toda a fase de execução do projeto em aspectos que poderão afetar as etapas de teste de capacidade. Existem diversos questionamentos que devemos fazer na fase de captura de requisitos não-funcionais para podemos projetar a arquitetura, validá-la com teste de capacidade e também traçarmos um plano de crescimento.



## Formulário de Análise de Requisitos não-funcionais

**Nome do Projeto:** Estudo de Caso Academia do Arquiteto

**Data coleta dos dados:** 1/1/2000

### Perguntas dissertativas

- *Qual é a demanda prevista para usar a solução?*  
R. 200 acessos por dia em média.
- *Existe possibilidade de picos? Se sim, qual o pico previsto?*  
R. Sim. Podemos chegar a um pico de 100 usuários simultâneos.
- *Qual é o tempo de resposta desejado?*  
R. O nível ideal de trabalho é que o usuário não espere mais do que 2 segundos por cada resposta.
- *Os acessos durante o dia vão se concentrar mais em um horário específico?*  
R. Sim, 80% devem ocorrer entre as 11:00 e 21:00.
- *Existe um nivelamento no acesso durante a semana ou existe um período de maior concentração?*  
R. Segundas e terças, acesso baixo, 100 acessos por dia.  
Quartas, quintas, médio acesso com 150 acessos por dia.  
Sextas e sábados, pico de 300 acessos por dia.  
Domingo, 200 por dia.
- *Quanto seu negócio tende a crescer no próximo ano e no ano seguinte dentro do cenário otimista do seu business plan?*  
R. Pretendemos crescer 15% neste ano e 30% no próximo.
- *O negócio que a solução atende pode apresentar variações extremas e situações atípicas com que grau de frequência?*  
R. 5%.

### Distribuição de utilização da solução por use-case

Nesta tabela o solicitante deve informar a previsão de uso das funcionalidades para que você consiga planejar scripts que simulam a realidade. De nada vale testar nosso estudo de caso com a entidade Curso se considerarmos os valores relacionados abaixo:

| Use-case   | % de acesso |
|--|-------------|
| Secretaria mantém cursos<br>(adiciona, exclui, inclui e pesquisa)  | 5%          |
| Secretaria mantém turmas<br>(criação de turmas, exclusão, matrícula de alunos e modificação de matrículas) | 35%         |
| Secretaria mantém memberships  | 50%         |
| Alunos se cadastram no sistema via Web   | 5%          |
| Alunos efetuam matrículas via Web  | 5%          |

### Cálculo de Volume de Dados por Entidade

Nesta etapa o solicitante / cliente deverá informar o volume esperado de dados por entidade para no mínimo os próximos dois anos. Procure facilitar o trabalho entregando no formulário a tabela já preenchida com as entidades identificadas na etapa de captura.

Este dado vai indicar também a integridade das informações respondidas nas perguntas dissertativas.

| Entidade   | Atual | Ano 1 | Ano 2 |
|------------|-------|-------|-------|
| Curso      | 25    | 35    | 50    |
| Membership | 4000  | 6000  | 8000  |
| Turma      | 40    | 100   | 300   |
| Matricula  | 600   | 1800  | 4000  |

## Identificação de Cenários Atípicos

Devemos considerar também que uma solução nunca é utilizada de forma linear, ou seja, quase todo negócio apresenta momentos onde a maior demanda por um grupo de entidades e use-cases aumenta brutalmente enquanto outras são menos requisitadas.

Devemos tentar levantar cuidadosamente situações atípicas que acontecem no domínio de negócio onde a solução está atuando:

- Em um software de gestão empresarial, as entidades de recursos humanos serão mais acessadas no final e início do mês;
- Um site de e-commerce pode oferecer uma promoção como nunca fez anteriormente causando um pico não previsto de 10 vezes mais usuários que o previsto;
- Em uma escola as entidades de matrícula e operação de inclusão são mais acessadas, causando um aumento nos outputs e redução nos inputs do sistema;
- Uma montadora comete um erro técnico e vende 200.000 mil carros com defeito e precisa elaborar uma ação de recall que vai demandar muito mais do seu sistema de gestão do que antes;

Procure sempre identificar os fatores de risco nos use-cases do projeto. Faça uso de métricas sugeridas por metodologias para aferir o mapa de riscos do cenário.

Como resultado do seu Teste de Capacidade você pode indicar a compatibilidade da solução com tais situações e recomendar arquiteturas alternativas de fácil adaptação para o projeto.

Isso reforça a importância de um processo de captura de requisitos ideal. O bom analista não só recolhe informações relevantes como antecipa problemas através da otimização dos seus processos de captura de requisitos.

### Tabela de Mapa de Riscos

A entrega para o cliente deve ser uma especificação relacionando situações que podemos chamar de cenários. Futuramente utilizaremos os cenários para especificar planos específicos e estudarmos arquiteturas alternativas e necessidade de flexibilização da solução.

| Risco identificado | Documentação  |
|--------------------|---|
| Época de Matrícula | A empresa fornece mini-cursos gratuitos e repentinamente pode ter picos atípicos com lançamentos e anúncios na mídia. |

## Melhores práticas na captura de requisitos para testes

Na prática costumamos lidar com dois tipos de cenários que atuam em pontos extremos e raramente vemos o terceiro cenário:

- *Cenário 1:* não existe nenhuma ou quase nenhuma preocupação quanto ao desempenho e performance do sistema pois não existe nenhum “gargalo” aparente.
- *Cenário 2:* existe uma chocante necessidade de uso em escala extrema e toda a concentração da equipe fica voltada para a capacidade de processamento, prejudicando o andamento das funcionalidades de negócio.
- *Cenário 3:* existe um equilíbrio entre produzir o software e garantir que suas funcionalidades atendam a demanda esperada. Somente a metodologia, experiência e conhecimento de processos conseguem garantir tal equilíbrio.

Devemos sempre procurar um ponto de equilíbrio e sempre validar novas situações e novas arquiteturas que planejamentos ainda que informalmente. Desenvolver uma prova conceitual e promover um laboratório no seu computador pessoal pode passar uma noção e *feeling* do comportamento da sua idéia no ambiente real.

Preocupe-se com as informações sobre a demanda de forma mais abrangente que seu solicitante.

Não se responsabilize totalmente pelo teste em caso de ausência de informações vitais para arquitetura e teste de capacidade.

Sempre que possível, desenvolva provas de conceito com mais de uma arquitetura e execute teste de capacidade com peças conceituais.

Anexe no resultado final toda a especificação coletada com seu cliente, configuração de ambiente de teste, versão do código utilizado e resultados de consumo de hardware.

## Planejamento de Testes de Capacidade

Alimentamos a fase de planejamento de testes com dados capturados na etapa que apresentamos anteriormente. As respostas para os questionários propostos fornecerão uma boa base para o planejamento do teste.

O planejamento dos testes é o resultado da análise dos dados coletados. Vejamos algumas informações relevantes para o teste, que capturamos com os questionários:

- A arquitetura deve atender 100 acessos simultâneos no startup do projeto;
- Devemos projetar a arquitetura para escalar para 115 acessos simultâneos no primeiro ano, e no segundo para 150 acessos simultâneos:

Escala em 2 anos =  $((\text{USUARIOS\_ATUAIS} + \text{CRESC\_ANO1}) + \text{CRESC\_ANO2}\%)$

Escala em 2 anos =  $((100 + 15\%) + 30\%) = 150$  usuários.

- Devemos chegar em uma conclusão de hardware e arquitetura necessária para a demanda atual e devemos também recomendar como aumentar a capacidade dos servidores para a demanda nos próximos dois anos.
- Nosso principal foco são as turmas e matrículas;

O planejamento gera um documento que representa um delivery para seu cliente. Neste documento você deve sintetizar suas conclusões e deve propor unidades de trabalho que representem as atividades dos usuários.

### Exemplo de Proposta de Planejamento de Testes

#### Cenário de Uso Comum

| Incidência Global                         | Incidência local | Ação                                | Actor      | Etapas   |
|---|------------------|-------------------------------------|------------|--|
| <b>Use-case: Secretaria mantém cursos</b> |                  |                                     |            |  |
| 35%                                       | 35%              | Cadastrar novo membership           | Secretaria | Request de form<br>Submit de form  |
|   | 35%              | Alterar dados de membership         | Secretaria | Load dos memberships<br>Edit membership<br>Submit de form                              |
|   | 5%               | Exclusão de memberships             | Secretaria | Load<br>Edit<br>Delete   |
|   | 25%              | Visualização de dados               | Secretaria | Load<br>Edit   |
| <b>Use-case: Secretaria mantém turmas</b> |                  |                                     |            |  |
| 50%                                       | 5%               | Matricular membership               | Secretaria | Load das turmas<br>Edit turma<br>Adicionar matricula<br>Submit matricula<br>Load turma |
|   | 5%               | Excluir matricula                   | Secretaria | Load das turmas<br>Edit turma<br>Excluir turma<br>Load das Turmas                      |
|   | 5%               | Alterar dados da matrícula          | Secretaria | Load das turmas<br>Edit Turma<br>Submit Turma<br>Load das turmas                       |
|   | 5%               | Criação de turmas                   | Secretaria | Request de form<br>Submit de form  |
|   | 80%              | Visualização de turmas e matrículas | Secretaria | Load das turmas<br>Edit turma  |

\*Vamos considerar a operação de login como aleatória, pois não temos 1 login por operação.



*Cenário: “Época de Matrículas”*

| Incidência Global                         | Incidência local | Ação                                | Actor      | Etapas   |
|---|------------------|-------------------------------------|------------|--|
| <b>Use-case: Secretaria mantém cursos</b> |                  |                                     |            |  |
| 35%                                       | 60%              | Cadastrar novo membership           | Secretaria | Request de form<br>Submit de form  |
|   | 25%              | Alterar dados de membership         | Secretaria | Load dos memberships<br>Edit membership<br>Submit de form                              |
|   | 5%               | Exclusão de memberships             | Secretaria | Load<br>Edit<br>Delete   |
|   | 10%              | Visualização de dados               | Secretaria | Load<br>Edit   |
| <b>Use-case: Secretaria mantém turmas</b> |                  |                                     |            |  |
| 50%                                       | 50%              | Matricular membership               | Secretaria | Load das turmas<br>Edit turma<br>Adicionar matricula<br>Submit matricula<br>Load turma |
|   | 5%               | Excluir matricula                   | Secretaria | Load das turmas<br>Edit turma<br>Excluir turma<br>Load das Turmas                      |
|   | 15%              | Alterar dados da matrícula          | Secretaria | Load das turmas<br>Edit Turma<br>Submit Turma<br>Load das turmas                       |
|   | 5%               | Criação de turmas                   | Secretaria | Request de form<br>Submit de form  |
|   | 25%              | Visualização de turmas e matrículas | Secretaria | Load das turmas<br>Edit turma  |

\*Vamos considerar a operação de login como aleatória, pois não temos 1 login por operação.

## Ferramenta JMeter

JMeter é um software open-source mantido pelo grupo Jakarta Apache que tem a capacidade de executar plano de testes configurados através da sua ferramenta gráfica.

Podemos utilizar o JMeter para teste de performance de aplicativos para simular uma demanda. Também é possível adaptá-lo para trabalharmos com testes de caixa preta.

Algumas características que tornam o JMeter uma ferramenta gratuita de alto valor agregado:

- Pode executar testes através Samplers para HTTP, FTP, SOAP, JDBC, LDAP ou Java;
- 100% escrito em Java, provendo portabilidade entre plataformas;
- Interface gráfica elaborada com Java Swing;
- Ferramenta Multithreading de teste permitindo que uma só máquina simules muitas requisições simultaneamente;
- Log de resultados para análise off-line e cachê para análise on-line;
- Alta extensibilidade:
  - Permite o desenvolvimento de novos Samplers;
  - Permite o desenvolvimento de novos plug-ins de análise;
- Estatísticas e gráficos on-line;

## Download e instalação

O download pode ser feito através do site do grupo Apache:

<http://jakarta.apache.org>

Faça do download de uma versão binário no format ZIP.

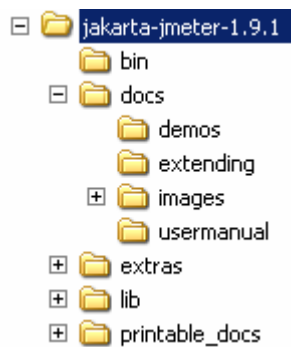
Trabalhamos atualmente com a versão 1.9.1 na Academia do Arquiteto. Caso esta versão tenha sido atualizada o instrutor notificará.

Descompacte o arquivo ZIP no diretório de preferência.

A instalação está concluída, vale lembrar que a ferramenta é totalmente dependente do ambiente Java 2 1.4 Standard Edition.

## Inicializando o JMeter

A seguinte estrutura de diretórios será criada no local de descompactação do arquivo de instalação do JMeter:



Navegue para o diretório bin e no ambiente Unix digite:

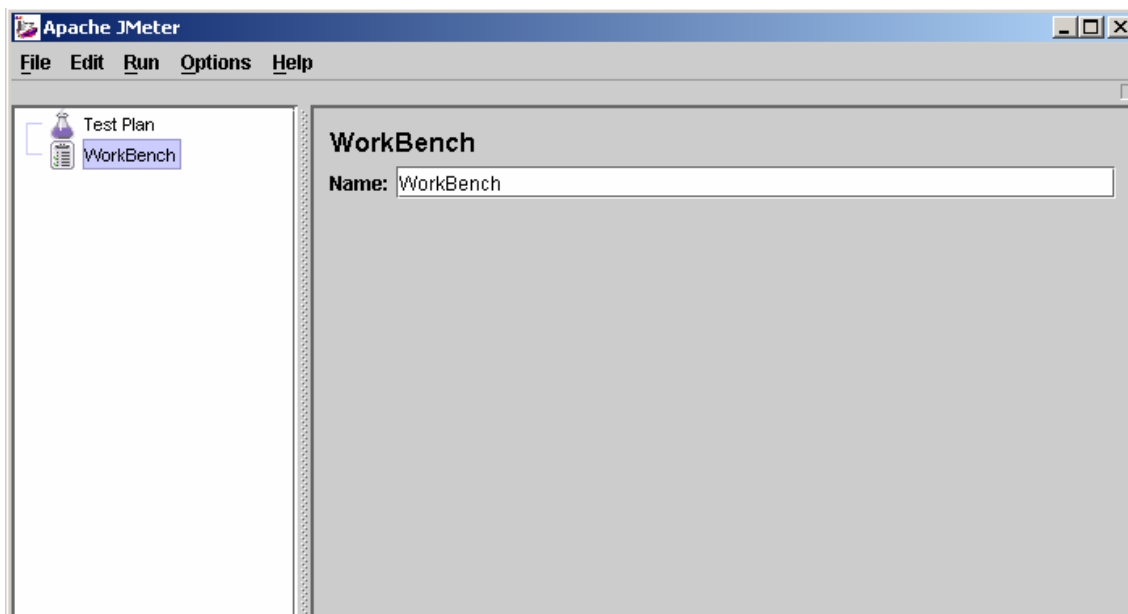
```
→ java -jar ApacheJMeter.jar
```

No ambiente Windows digite:

```
→ jmeter.bat
```

## Criando um Plano de Testes

Ao inicializarmos o JMeter a seguinte interface será apresentada:



Temos, portanto no lado esquerdo uma árvore de elementos (JTree) uma representação dos elementos que compõem nosso plano de testes.

Dois itens principais são apresentados na raiz da nossa árvore, por padrão, com os seguintes nomes:

1. **Test Plan:** agrupa itens que representam a simulação de múltiplos usuários no plano de testes (samplers), além de configuradores e controladores de lógica de execução do teste.
2. **WorkBench:** área de trabalho para armazenamento temporário de elementos. Os itens associados à este elemento não são considerados como parte do plano de testes.

Os elementos dentro da árvore são adicionados de forma ordenada e hierárquica. Determinados elementos são sensíveis à hierarquia e / ou a ordem em que eles se encontram na árvore.

## Elementos JMeter

Um plano de testes é composto por diversos elementos que representarão os usuários acessando a solução e também configurações, visualizadores de resultados entre outros.

| Tipo de Elemento              | Descrição   |
|-------------------------------|---|
| <b>Test Plan</b>              | Representa seu plano e todos seus elementos.  |
| <b>Workbench</b>              | Área temporária de trabalho que apóia o desenvolvimento do plano de testes.   |
| <b>Thread Groups</b>          | Representa um grupo de usuário executando determinada(s) solicitação(ões).  |
| <b>Samplers</b>               | Representa uma solicitação. O JMeter suporta solicitações para HTTP, FTP, SOAP, JDBC, LDAP e Java.<br>Elementos do tipo Sampler são adicionados em um Thread Group.   |
| <b>Logic Controllers</b>      | Representam elementos que ajudam a controlar a execução das requisições através de repetidores, módulos, randomização entre outros.   |
| <b>Listener</b>               | Elementos que visualizam resultados que podem ser representados por gráficos, tabelas, entre outros.  |
| <b>Configuration Elements</b> | Para configuração padrão de dados. Com eles conseguimos, por exemplo, configurar o mesmo servidor HTTP para uma determinada solicitação.  |
| <b>Assertions</b>             | Elementos que possibilitam diversas verificações nas respostas obtidas.   |
| <b>Pre Processors</b>         | Elementos que podem produzir dados para enviar como parte de uma solicitação. Por exemplo, em teste de uma rotina de inclusão de dados no sistema, devemos alternar alguns dados que são exclusivos por definição. Temos pré-processadores que são capazes de gerar: nome_1, nome_2, nome_3, etc. |
| <b>Post Processors</b>        | Processadores de resultados de requisições programadas. Pode extrair uma determinada parte da resposta do servidor utilizando expressões regulares.   |
| <b>Timer</b>                  | Elementos que permitem um controle avançado no intervalo de execução das requisições.   |

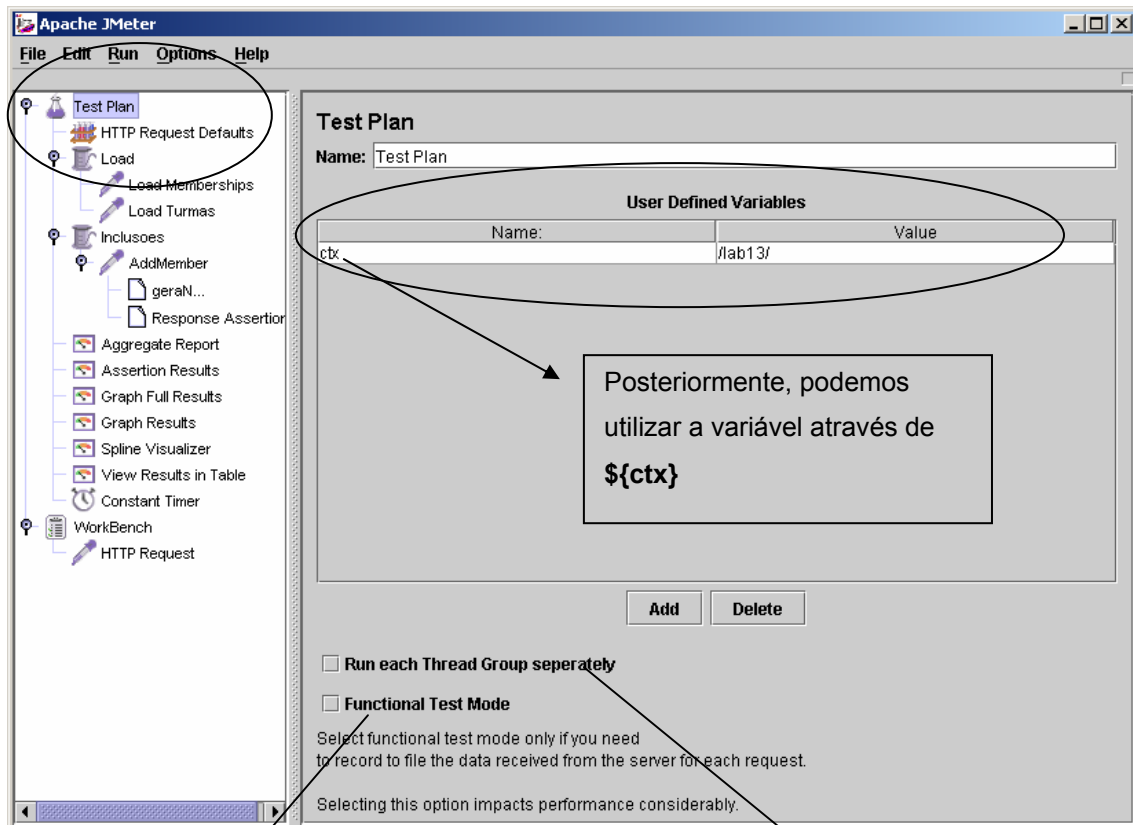
## Utilizando Elementos Básicos

Os elementos apresentados na tabela podem devem ser colocados no seu plano de testes dentro de uma determinada hierarquia obrigatória. A tabela a seguir, documenta quais elementos podem ser adicionados em quais elementos.

| Elemento               | Elementos que podemos adicionar como filho / child  |
|------------------------|---|
| Test Plan              | Thread Group, Listeners, Config Element, Assertions, Pre Processor, Post Processor, Timer |
| Workbench              | Logic Controller, Sampler, Config Element, Nom-Test Elements                              |
| Thread Group           | Logic Controller, Listener, Sampler, Timer, Config Element, Pre Processor, Post Processor |
| Sampler                | Config Element, Assertion, Timer, Pre Processor, Post Processor                           |
| Logic Controller       | Logic Controller, Sampler, Config Element, Timer, Listener, Pre Processor, Post Processor |
|                        |   |
| Listener               | Nenhum  |
| Configuration Elements | Nenhum  |
| Assertions             | Nenhum  |
| Pre Processors         | Nenhum  |
| Post Processors        | Nenhum  |
| Timer                  | Nenhum  |

## Elemento Test Plan

Representa seu plano e todos os elementos que o compõe. Podemos configurar variáveis globais, chamadas de User Defined Variables.



Execução de teste funcional (caixa preta). Fará que o JMeter armazene o resultado das requisições enviadas para o servidor.

Se selecionado, executa cada grupo de usuários (Thread Group) em sequência, do contrário, executará as Thread Groups de forma paralela.

## Workbench

Área temporária de trabalho, você pode mover temporariamente elementos para esta área, copiar e colar etc. Não é salvo juntamente com o Test Plan, você deve salvar manualmente clicando o botão direito do mouse no item e escolhendo Save As...

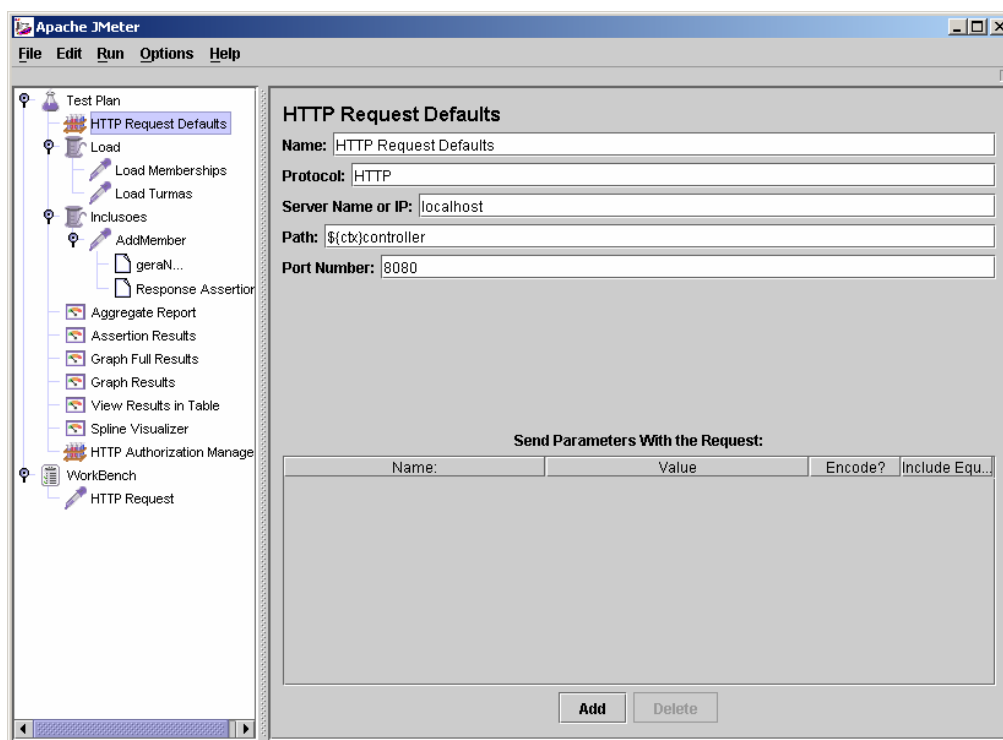
O Workbench não tem efeito sobre seu plano de testes, ou seja, exerce meramente a função de apoio para o desenvolvimento do plano de testes.



## Config Element

Como seu próprio nome afirma, são elementos de configuração do nosso plano de testes. Com eles podemos tornar nosso plano mais flexível e com informações componentizadas.

Os Config Elements mais utilizados são os que definem padrões de configuração para tarefas. Por exemplo, quando estressamos um aplicativo Web, temos a necessidade de acionar diversas vezes um mesmo controller passando somente parâmetros diferentes. Neste caso, com um elemento HTTP Requests Default, estabelecemos o nome de servidor padrão, porta, protocolo e URL e nas tarefas definimos somente os parâmetros de acionamento do Controller.



Notem que atribuímos como protocolo padrão HTTP, Server Name como localhost, Path atribuímos `${ctx}controller` e Port Number 8080. Todas as requisições HTTP (elemento Sampler -> HTTP Request) terão como padrão esses valores podendo sobrepo-lô se necessário.

Um aspecto interessante nesta configuração é o uso da User Defined Variable `${ctx}` apresentada no elemento Test Plan. Caso mude o contexto do aplicativo, é bastante simples mudá-lo para todas as requisições, uso altamente recomendado.

Temos diversos tipos de Config Element, conforme apresentamos na tabela a seguir:

| Config Element                         | Descrição  |
|--|--|
| Login Config Element                   | Padrão de login para atividades que utilizão de sistemas de login.                       |
| Simple Config Element                  | Permite que desenvolvedores adicionem novos componentes e funcionalidades para o JMeter. |
| FTP Request Defaults                   | Padrão de acesso a servidores FTP.   |
| HTTP Request Defaults                  | Padrão de acesso a servidores HTTP.  |
| HTTP Authorization Manager             | Para acesso a páginas protegidas que requerem login via HTTP.                            |
| HTTP Cookie Manager                    | Permite que você configure Cookies para enviar nas requisições HTTP.                     |
| HTTP Header Manager                    | Permite a customização do Header HTTP utilizado para teste.                              |
| Java Request Defaults                  | Padrões para testes em classes Java.   |
| JDBC Database Login Defaults           | Padrões para login via JDBC  |
| JDBC Database Connection Pool Defaults | Padrões do pool utilizado para testes com JDBC.  |
| JDBC SQL Query Defaults                | Padrões de consultas SQL.  |
| LDAP Request Defaults                  | Padrões para testes em servidores LDAP.  |

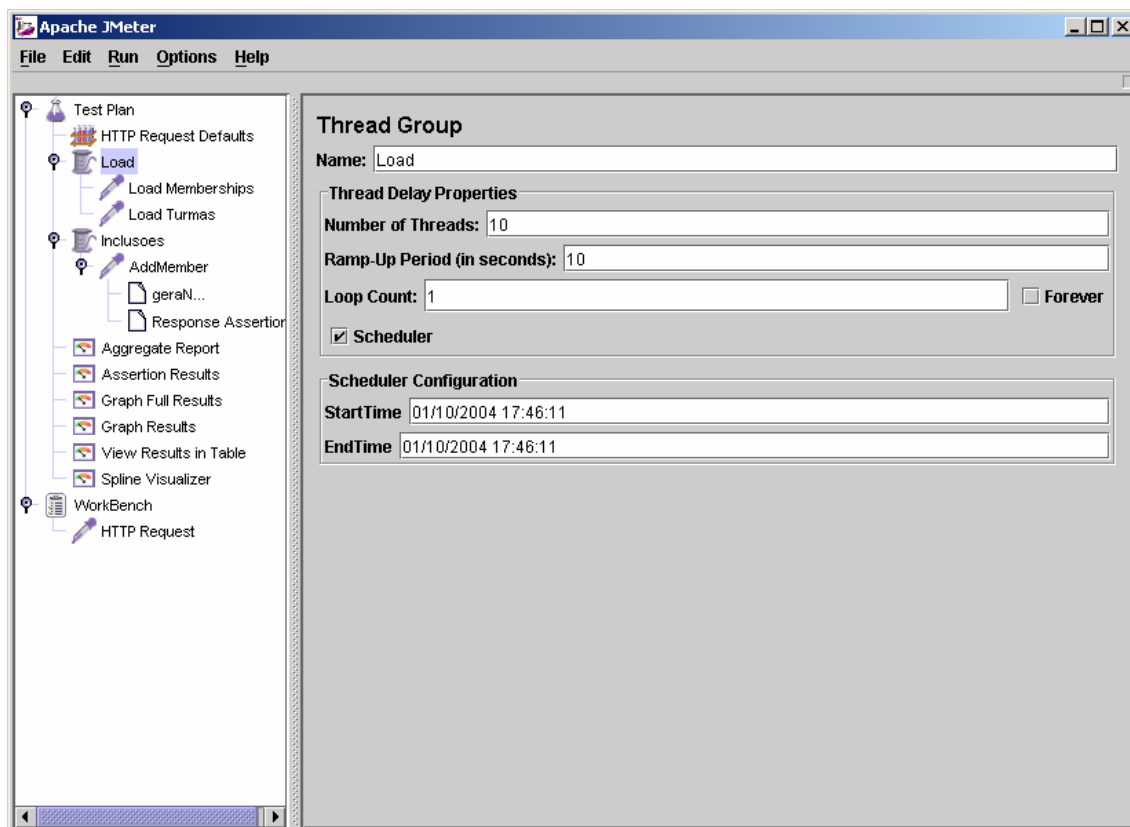
Os elementos de configuração são sensíveis à ordem e hierarquia em que você os adiciona em seu Test Plan. Fique atento neste detalhe.

Caso trabalhe com processo de autenticação HTTP você vai precisar de Config Elements do tipo Authorization Manager.



## Thread Group

Um elemento Thread Group representa uma determinada demanda ou um conjunto de usuários executando uma mesma atividade. Por esse motivo é um dos elementos mais importantes no seu plano e deve ser utilizado e configurado com bastante cautela para que você crie um plano de testes que de fato simule a realidade.



Configuramos em um Thread Group:

**Number of Thread:** a quantidade de usuários simultâneos;

**Ramp-up Period (in seconds):** intervalo entre os lançamentos de requisições. O valor digitado será dividido pelo número de requisições, e o resultado será o intervalo real entre cada requisição. No exemplo acima, temos 10 threads e Ramp-up 10, 10 dividido por 10 resulta em 1, portanto teremos um disparo de atividade nesta thread a cada segundo.

**Loop Count:** quantidade de vezes que queremos executar as threads de teste. Este número multiplicado pela quantidade de threads resulta no total de requisições que serão enviadas.

**Forever:** se ligado, ignora o valor configurado em Loop Count e executa as tarefas até que você cancele a execução do plano de testes.

**Scheduler:** permite que você agende o disparo da Thread Group em um determinado horário.

Após configurar a Thread Group com os valores convenientes para seu plano de testes, você poderá adicionar uma ou mais atividades de requisições para servidores, no JMeter representadas por elementos do tipo Sampler.

Estudaremos Samplers a seguir, portanto, note que no exemplo da imagem acima, temos uma Thread Group chamada **Load** com dois Samplers HTTP: **Load Memberships** e **Load Turmas**. Esta Thread Group estará simulando um usuário clicando na opção “Visualizar Memberships” e outro clicando em “Visualizar Turmas” no mesmo instante.

Como esta thread group foi configurada para 10 amostras de 10 threads com intervalo de um segundo entre as threads, teremos de segundo em segundo uma solicitação para Visualizar Memberships e outra para Visualizar Turmas.

Você pode adicionar uma ou mais Thread Group em um plano de testes. Vale lembrar que por padrão as Thread Group são disparadas em paralelo, ou seja, simultaneamente. Caso queira alterar este comportamento, habilite o parâmetro **Run Each Thread Group Separately** no Test Plan.

## Sampler

Samplers são elementos que farão a requisição física para um determinado servidor. Temos 7 diferentes Samplers inclusos na versão utilizada do JMeter. Samplers sempre são adicionados as Thread Groups.

Podemos também desenvolver nossos próprios Samplers. Essencialmente o que temos que fazer é implementar uma interface em uma classe, empacotá-la em um jar, e informar o JMeter da existência deste jar.

Os seguintes Samplers estão disponíveis por padrão:

| Nome  | Descrição  |
|---|--|
| <b>FTP Request</b>                            | Para efetuar downloads via FTP como parte do plano de testes.  |
| <b>HTTP Request</b>                           | Utilizado para simular requisições HTTP, fazendo com o que o JMeter atue como um browser. O Sampler mais utilizado.  |
| <b>SOAP/XML-RPC Request</b>                   | Para requisições simples via SOAP. Configuramos uma URL e um XML para enviar para o servidor.  |
| <b>WebService (SOAP) Request (Alpha Code)</b> | Permite o envio de requisições Webservice mais elaboradas. Capacidade de leitura de arquivos WSDL.   |
| <b>Java Request</b>                           | Para execução de testes em classes customizadas. É um ponto de extensão do framework JMeter.   |
| <b>JDBC Request</b>                           | Para testes de carga em banco de dados via JDBC. Podemos utilizar também para tarefas simples como criar tabelas, excluir todos os dados antes de iniciar o teste, entre outros. |
| <b>LDAP Request</b>                           | Para testes em servidores LDAP.  |

Vamos exemplificar o uso de Sampler através do HTTP Sampler pelo fato de ser o mais utilizado atualmente.

## Exemplo de HTTP Sampler

Informações fornecidas no Config Element HTTP Request Defaults. O preenchimento destes dados causaria o efeito de sobreposição das informações padrões.

**Apache JMeter**

File Edit Run Options Help

Test Plan

- HTTP Request Defaults
- Load
- Load Memberships
- Load Turmas
- Inclusões
- AddMember
- geraN...
- Response Assertion
- Aggregate Report
- Assertion Results
- Graph Full Results
- Graph Results
- View Results in Table
- Spline Visualizer
- WorkBench
- HTTP Request

**HTTP Request**

Name: Load Memberships

**Web Server**

Server Name or IP:

Port Number:

**HTTP Request**

Protocol: Method: ☒ GET ☐ POST

Path: ☐ Follow Redirects ☒ Use KeepAlive

**Send Parameters With the Request:**

| Name    | Value           | Encode?                  | Include Equ.                        |
|---------|-----------------|--------------------------|-------------------------------------|
| command | membership.Load | <input type="checkbox"/> | <input checked="" type="checkbox"/> |

Add Delete

**Send a File With the Request:**

Filename:  Browse...

Parameter Name:

MIME Type:

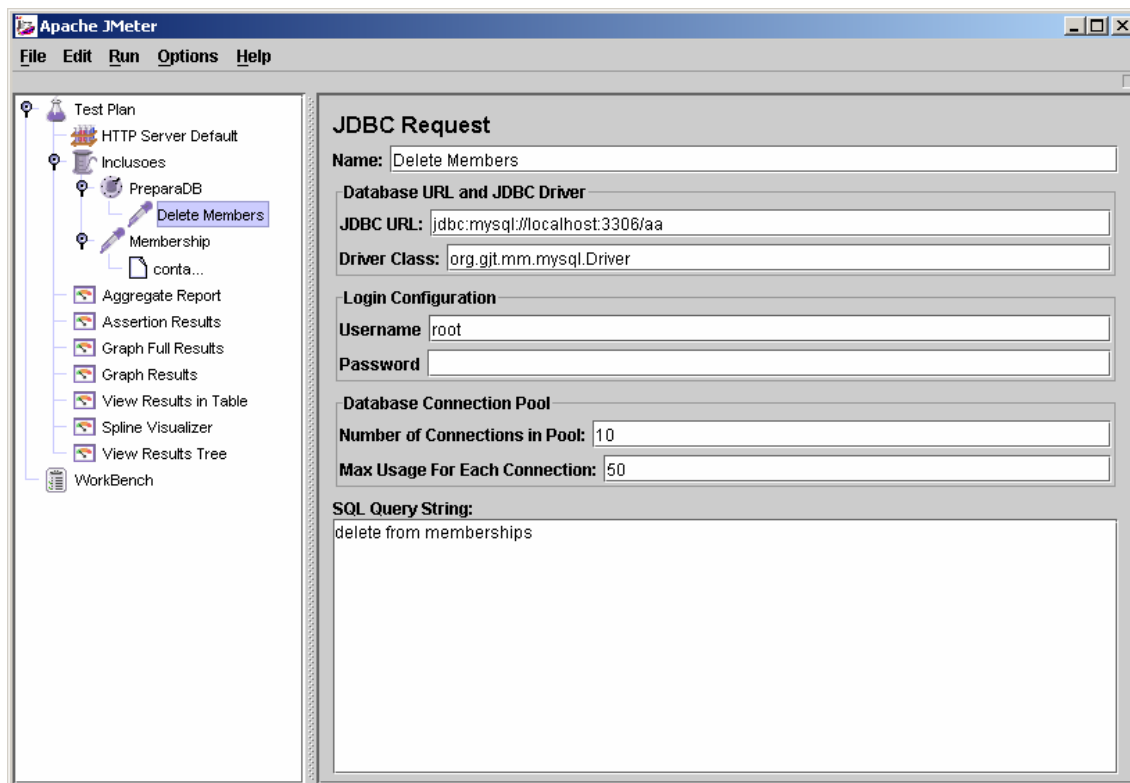
**Optional Tasks**

☐ Retrieve All Images and Java Applets (HTML files only)

Opções para simular um upload de arquivo.

Parâmetros a serem enviados para o Controller.

## Exemplo de JDBC Sampler



O Sampler JDBC Request é bastante simples de ser utilizado, basta configurar a URL JDBC, classe do driver e informações de login (que podem ser configuradas externamente em um Config Element Login).

Para que seu driver JDBC seja carregado, você vai precisar que ele esteja no diretório lib/ext do seu JMeter.

Um outro Sampler útil nos dias de hoje é o LDAP Request, nele podemos simular pesquisar, adições, deleções e modificações no servidor de diretórios em questão.

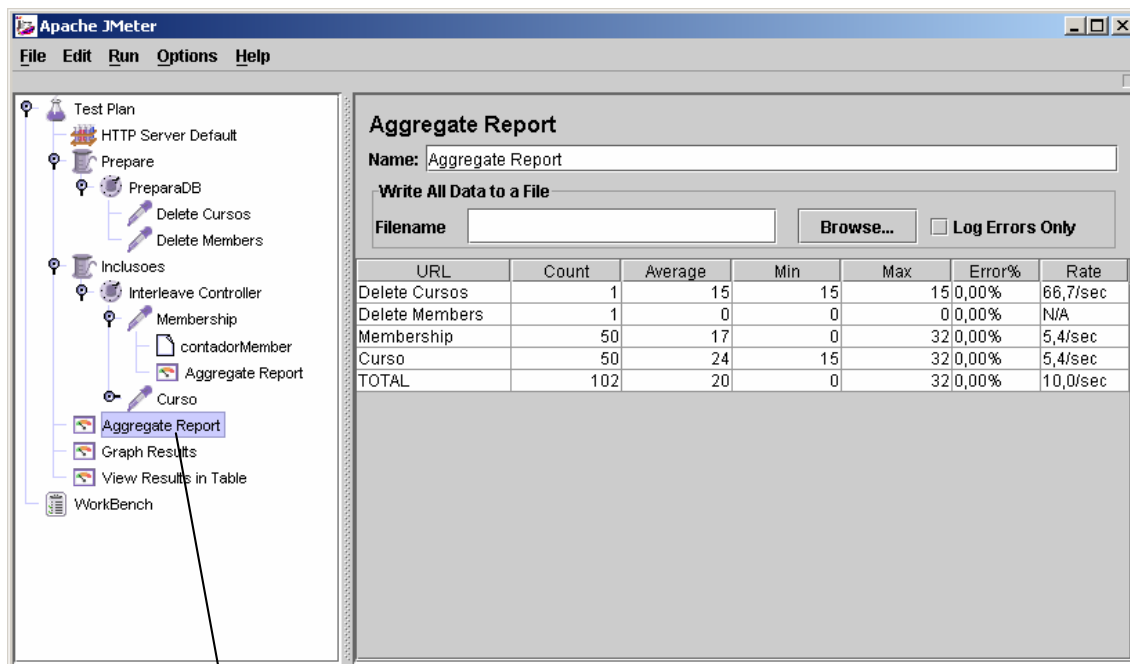
Um tipo de Sampler que você poderá questionar, seria algum para chamar EJBs que não temos por padrão. Talvez a concepção do projeto tenha o foco em testes na camada do client, neste ponto de vista, devemos testar o client do EJB e não ele próprio.

De um outro lado, visualizando o JMeter como um framework de testes de stress e também de caixa preta, tal funcionalidade seria bastante útil.

## Listeners

São elementos que capturam os resultados gerados pelo plano de testes e apresenta-os em um determinado formato.

Listeners podem ser diretamente vinculados a um Test Plan, neste caso teremos um mesmo listener para todos os Samplers.



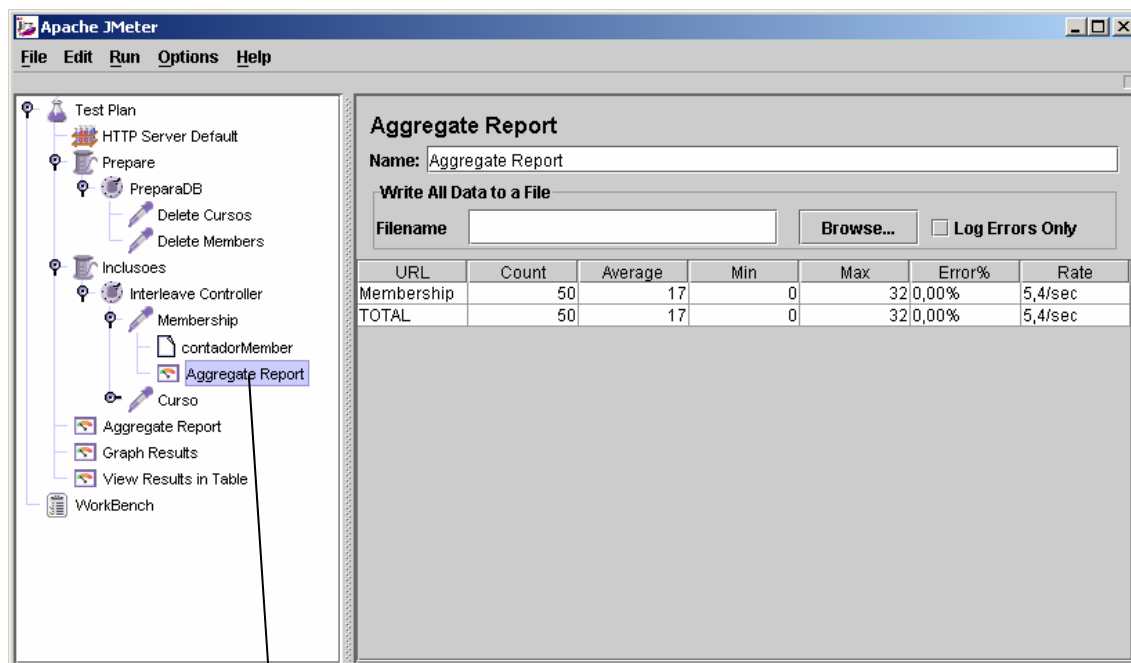
The screenshot shows the Apache JMeter GUI. On the left, the Test Plan tree includes: HTTP Server Default, Prepare, PrepareDB, Delete Cursos, Delete Members, Inclusions, Interleave Controller, Membership, contadorMember, Aggregate Report, Curso, another Aggregate Report (highlighted with a red box), Graph Results, View Results in Table, and WorkBench. An arrow points from this highlighted 'Aggregate Report' listener to a text box below. The main panel on the right displays the 'Aggregate Report' configuration with 'Name: Aggregate Report' and 'Write All Data to a File' checked. Below this is a table of results:

| URL            | Count      | Average   | Min      | Max       | Error%       | Rate            |
|----------------|------------|-----------|----------|-----------|--------------|-----------------|
| Delete Cursos  | 1          | 15        | 15       | 15        | 0,00%        | 66,7/sec        |
| Delete Members | 1          | 0         | 0        | 0         | 0,00%        | N/A             |
| Membership     | 50         | 17        | 0        | 32        | 0,00%        | 5,4/sec         |
| Curso          | 50         | 24        | 15       | 32        | 0,00%        | 5,4/sec         |
| <b>TOTAL</b>   | <b>102</b> | <b>20</b> | <b>0</b> | <b>32</b> | <b>0,00%</b> | <b>10,0/sec</b> |

Um listener adicionado ao Test Plan, captura e apresenta resultados de execução de todos Samplers.



No próximo exemplo, temos um listener vinculado a um Sampler específico, neste caso ele vai apresentar somente os resultados daquele Sampler:



Perceba a diferença no quadro ao lado.  
Agora só temos totalizações da  
execução do Sampler Membership.

Os seguintes Listeners já estão incluídos JMeter:

| Listener                     | Descrição  |
|------------------------------|--|
| <b>Assertion Results</b>     | Quando utilizamos assertions (verificações nas respostas dos samplers), este listener apresenta se determinada amostra está de acordo com a Assertion ou não.  |
| <b>Graph Full Results</b>    | Não funciona corretamente na versão 1.9 do JMeter, teoricamente deveria apresentar um gráfico de linha completo, com todas as respostas dos Samplers.  |
| <b>Graph Results</b>         | Apresenta um gráfico simples e útil. Com média, mediano, desvio padrão, mínimo e máximo do tempo de resposta das requisições.  |
| <b>Mailer Visualizer</b>     | Não disponível.  |
| <b>Simple Data Writer</b>    | Listener que tem a capacidade de armazenar os dados de resposta em um arquivo XML.   |
| <b>Spline Visualizer</b>     | Gráfico que apresenta uma linha continua com todos os resultados de tempo de resposta em milisegundos dos testes efetuados. Composto por 10 pontos, cada ponto contém a média 10% das amostras. Bastante útil para analisar impacto de performance e estabilidade. |
| <b>Aggregate Report</b>      | Mostra totalizações diversas do resultado.   |
| <b>View Results in Table</b> | Resultado individual de cada amostra, indicando seu tempo de resposta e seu obteve sucesso ou não.   |
| <b>View Results Tree</b>     | Apresenta cada requisição e resposta retornada pelo servidor. Excelente ferramenta para testes de caixa preta.   |

Todos os Listeners podem gravar seus resultados em um arquivo XML.

## Logic Controller

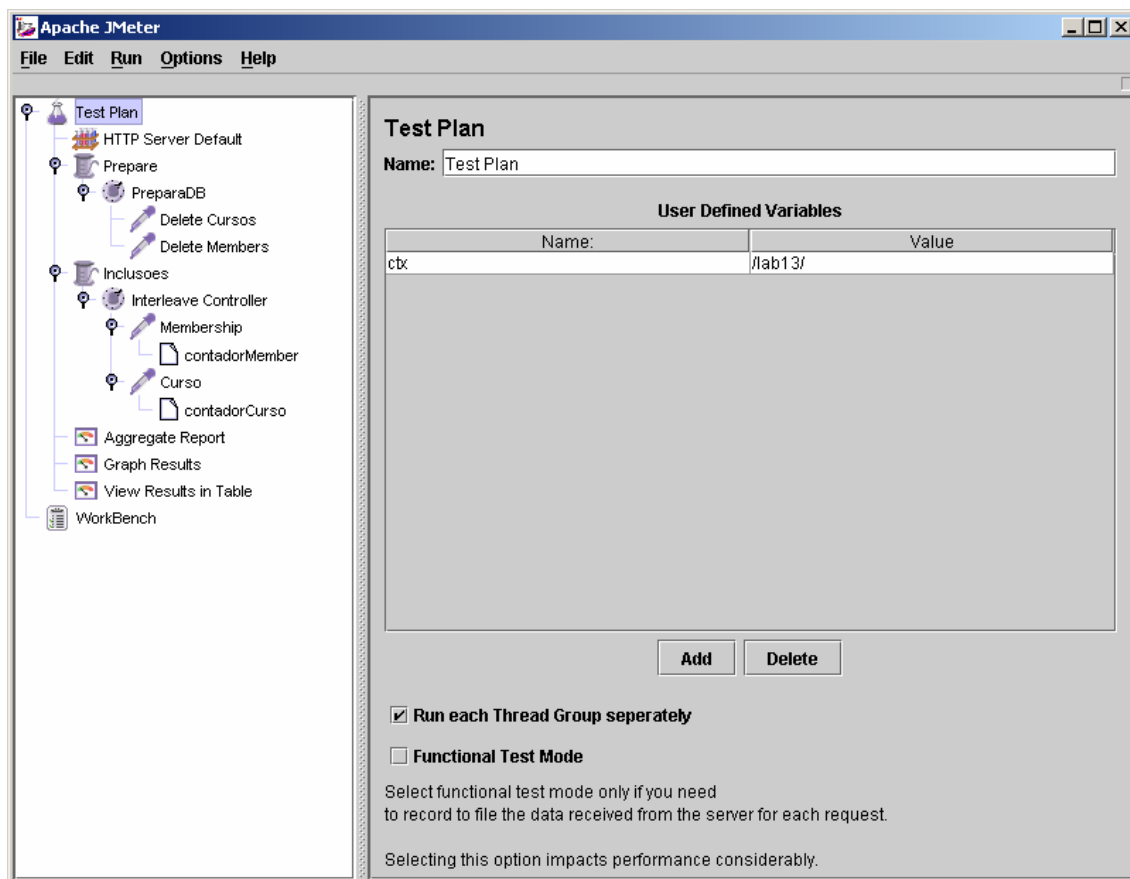
São elementos que permitem um controle mais customizado na execução das requisições dentro de uma Thread Group. Temos controladores lógicos que permitem a criação de laços, execução singular, modularização, randomização, entre outros.

Vejamos uma tabela completa:

| Controlador Lógico           | Descrição  |
|------------------------------|--|
| <b>Interleave Controller</b> | Vai executar as requisições contidas no elemento de controle de forma intervalada.   |
| <b>Simple Controller</b>     | Representa um grupo de Samplers. Utilizado com fins organizacionais, somente para agrupar um conjunto de samplers.   |
| <b>Loop Controller</b>       | Permite que determinado conjunto de Samplers tenham um laço específico.  |
| <b>Module Controller</b>     | Permite executar uma tarefa já está sendo executada em outra Thread Group. É um maneira excelente para você componentizar, modularizar e ter facilidades de manutenção do seu plano de testes.   |
| <b>Once Only Controller</b>  | Dentro de um loop, executa determinada atividade somente uma vez.  |
| <b>Random Controller</b>     | Executará um conjunto de atividades aleatoriamente.  |
| <b>Throughput Controller</b> | Para controle avançado de vazão. Permite que você opere um determinado conjunto de tarefas com uma parcela da vazão somente. Pode limitar a vazão por quantidade ou porcental, por usuário ou vazão global. Bem avançado e complicado quando combinado com outros controladores lógicos. |
| <b>Recording Controller</b>  | Quando utilizamos um servidor proxy HTTP, o Recording torna possível a gravação de dados retornados como resposta pelo Proxy. Opção bastante específica para o uso de proxy.   |

## Exemplo de Controladores Lógicos

O exemplo a seguir utiliza uma combinação de recursos para gerar um plano de testes. O plano executará as seguintes tarefas:



O primeiro detalhe é que configuramos “Run each Thread Group Separately” pois queremos que o plano seja executado em série e não paralelamente. Vejamos a sequência disparada no plano de testes:

1. **Test Plan - Prepare:** A primeira Thread Group que temos é a Prepare que vai executar JDBC Requests para excluir todos os dados da tabela Memberships e Cursos.
2. **Test Plan - Prepare - PreparaDB:** é um controlador do tipo **Once Only Controller** que vai garantir que independente de Loop Counts na Thread, esta atividade só será executada uma vez por usuário.
3. **Test Plan - Prepare – PreparaDB - Delete Cursos:** JDBC Request que remove todos os dados da tabela de cursos.

4. **Test Plan - Prepare – PreparaDB - Delete Members:** JDBC Request que remove todos os dados da tabela de memberships.
5. **Test Plan – Inclusoes:** thread group responsável pelas requisições de inclusão de dados.
6. **Test Plan – Inclusoes – Interleave Controller:** controlador que vai alternar entre as requisições de incluir um membership e incluir um curso.
7. **Test Plan – Inclusoes – Interleave Controller – Membership:** request HTTP que vai acionar o controller da aplicação para inclusão de um Membership.
8. **Test Plan – Inclusoes – Interleave Controller – Membership – contadorMember:** variável tipo Counter para gerar dados dinâmicos dos memberships, como nome\_1, nome\_2 etc. Estudaremos adiante.
9. **Test Plan – Inclusoes – Interleave Controller – Curso:** request HTTP que vai acionar o controller da aplicação para inclusão de um Curso.
10. **Test Plan – Inclusoes – Interleave Controller – Curso – contadorCurso:** variável tipo Counter para gerar dados dinâmicos dos cursos, como nome\_1, nome\_2 etc.
11. **Test Plan – Aggregate Report:** Listener para apresentação dos resultados do teste em forma de relatório de totalizações, médias, mínimo e máximo.
12. **Test Plan – Graph Results:** Apresenta resultado das requisições em modo gráfico. Inclui desvio padrão.
13. **Test Plan – View Results in Table:** dado de cada requisição em uma tabela. Inclui tempo individual de cada resposta e se obteve sucesso ou não.

## Elementos Avançados

Nesta parte vamos estudar alguns elementos que tornam o JMeter uma ferramenta robusta e para diversos fins.

### Assertions

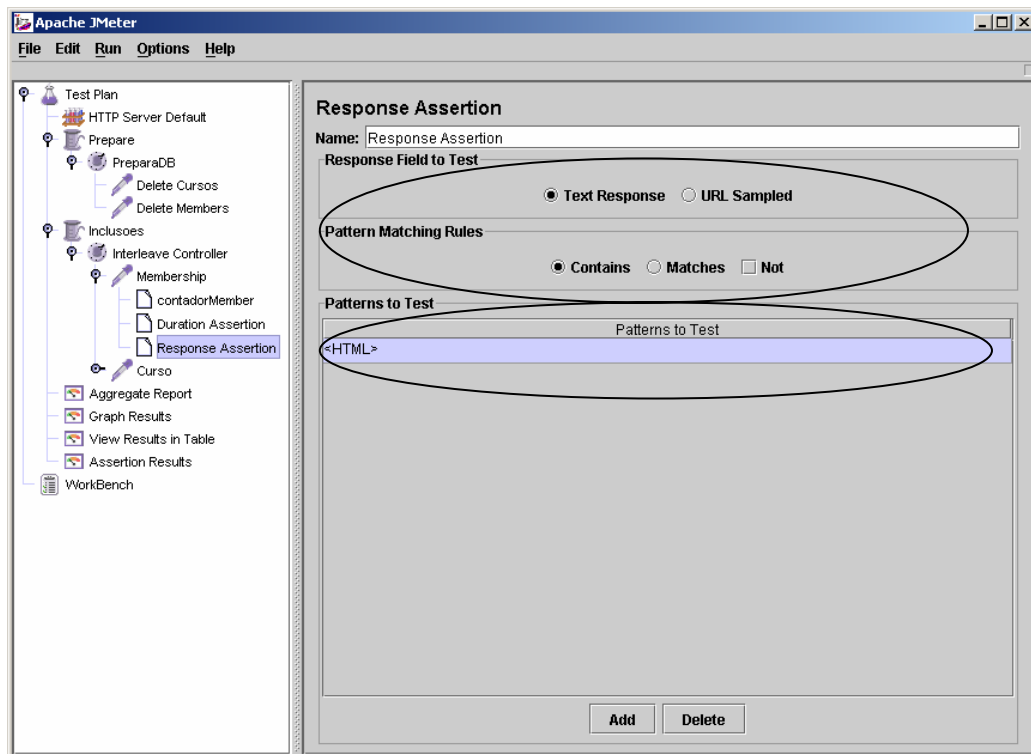
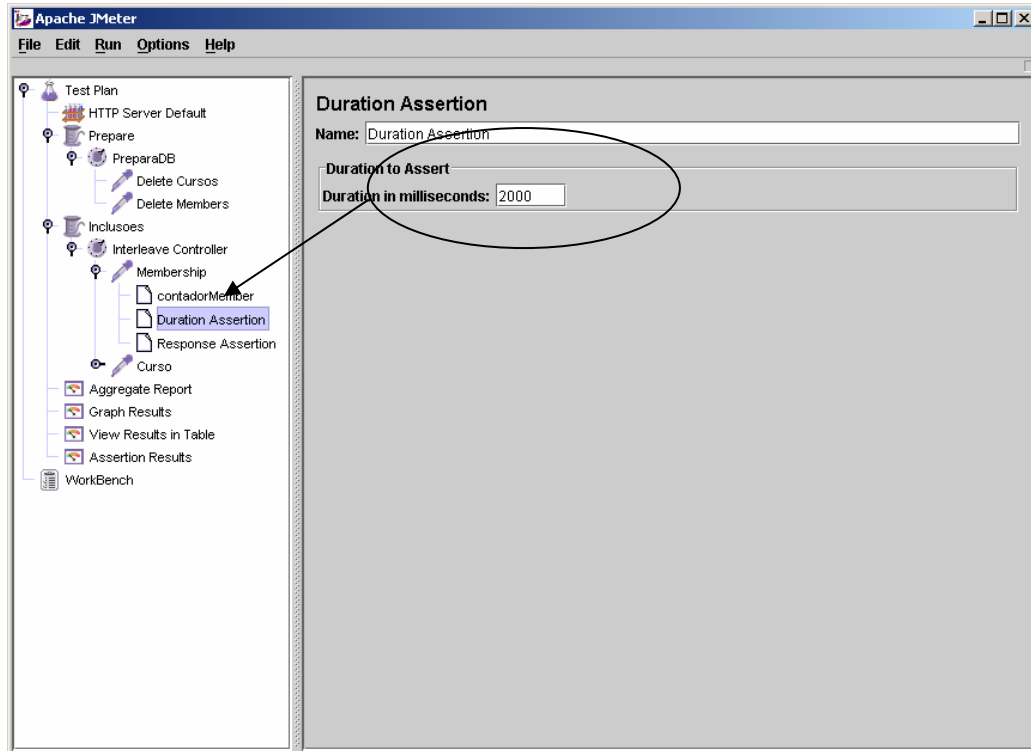
Assertion significa **afirmação**, e justamente podemos colocar afirmações em nosso plano de testes para verificar se determinada resposta está de acordo com alguma afirmação colocada no Sampler, vejamos alguns exemplos de afirmações:

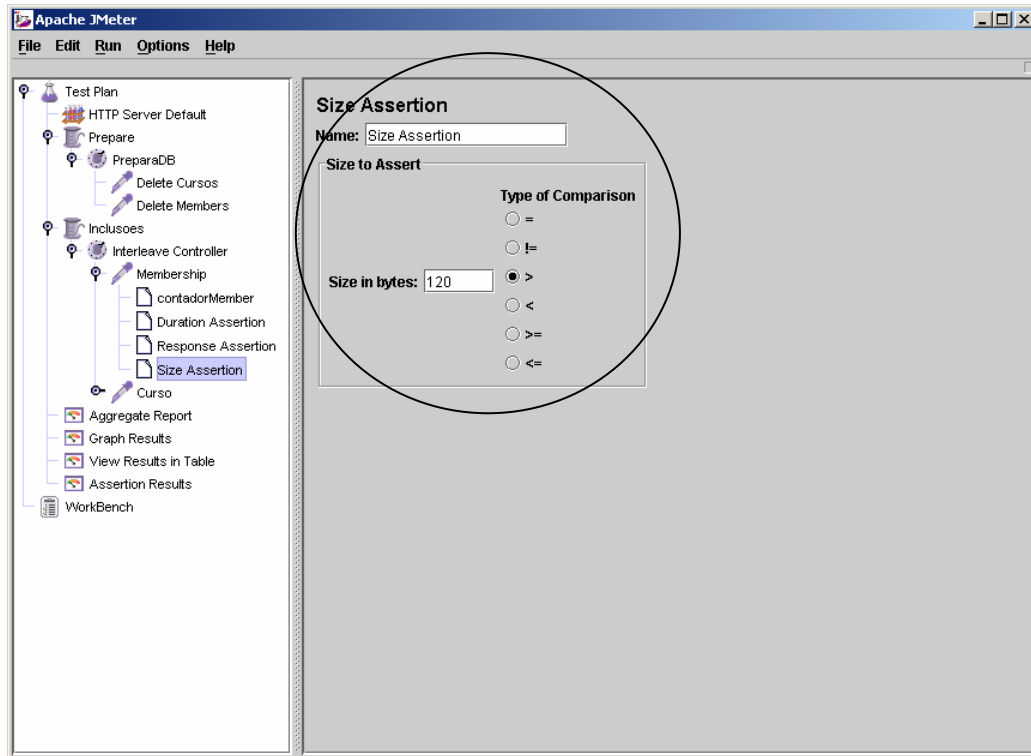
- Como parte da resposta temos que encontrar um texto contendo <!--RESULTADO OK -->;
- A requisição tem que retornar em menos de 2 segundos;
- A resposta tem que ser maior ou igual que 512 bytes;
- Como resposta temos que receber um documento XML como este (...);

Para permitir tais operações o JMeter conta com os seguintes componentes tipo Assertion:

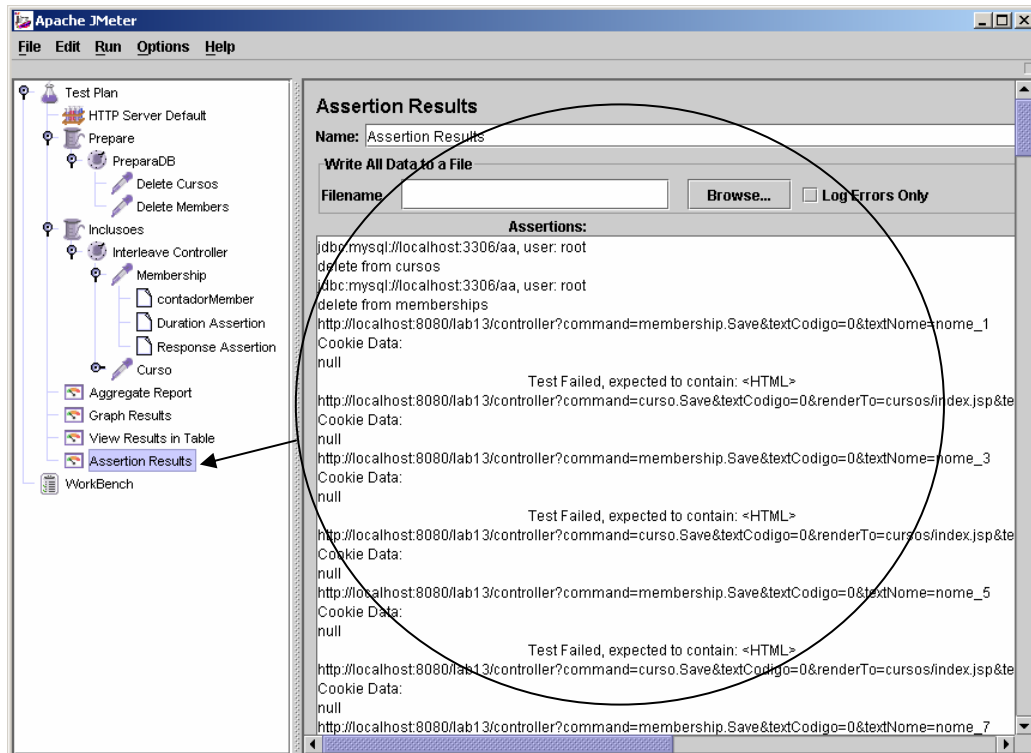
| Tipo de Assertion | Descrição   |
|-------------------|---|
| Response          | Permite que você verifique se você recebeu um conteúdo X como parte da resposta. Você pode colocar expressões lógica: deve conter, não deve conter ou deve ser exatamente tal conteúdo. |
| Duration          | Para afirmar que determinado Sampler tem que response em até X milisegundos.  |
| Size              | Determinada resposta deve ser menor, maior, igual, diferente que tantos bytes.  |
| XML               | Útil para testes de WebServices, você pode verificar se determinada resposta é equivalente a um documento XML especificado no plano de teste.   |

Vejamos alguns exemplos de telas de configuração de Assertions:





Resultado de verificações de assertion após os testes:





## Pre Processors

Pré-processadores são elementos que de alguma forma processam um dado antes de acionar um Sampler. Tipicamente utilizamos pré-processadores para gerar dados dinâmicos para o envio de solicitações para rotinas de manutenção de entidades.

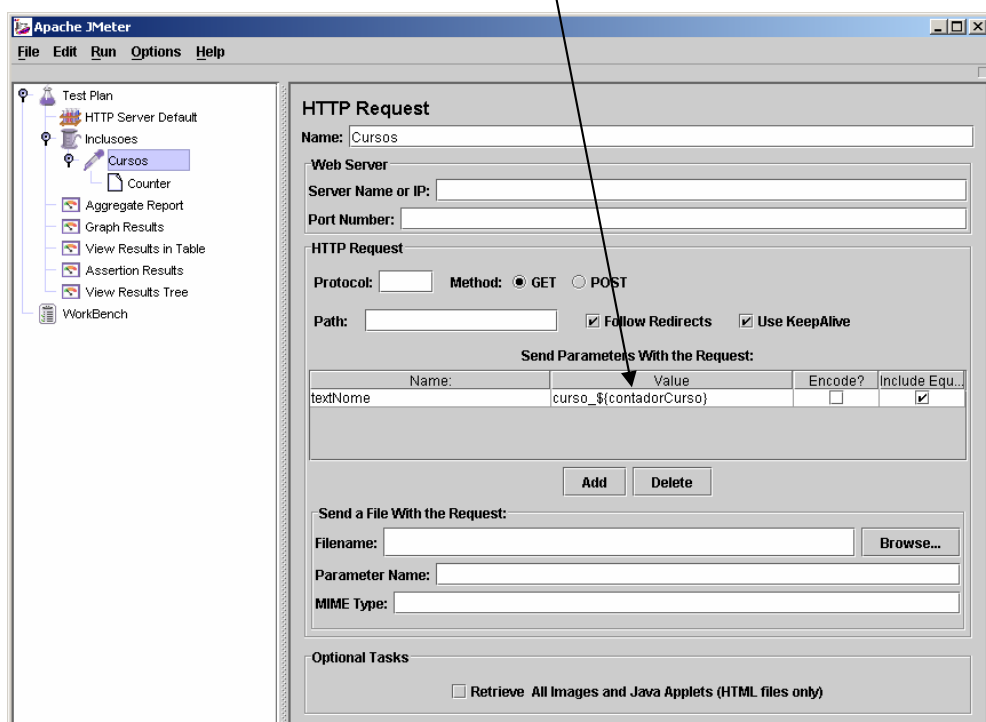
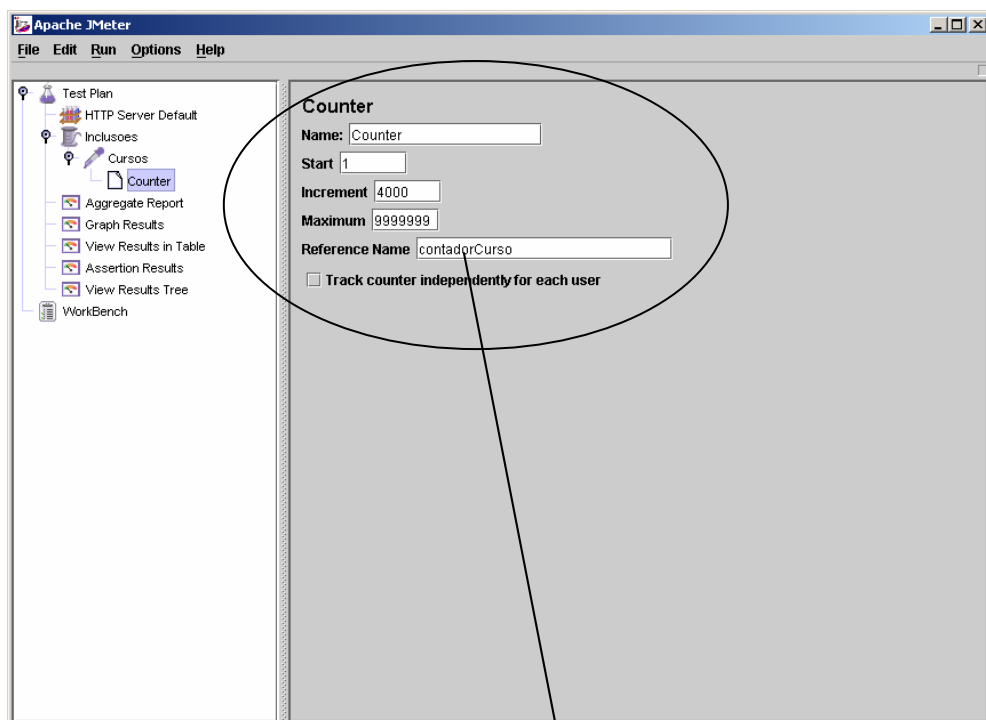
Um exemplo clássico é o pré-processador tipo Counter. Com ele criamos um contador configurável que pode ser utilizado através de um nome de variável atribuído em tempo de configuração.

O JMeter disponibiliza os seguinte pré-processadores:

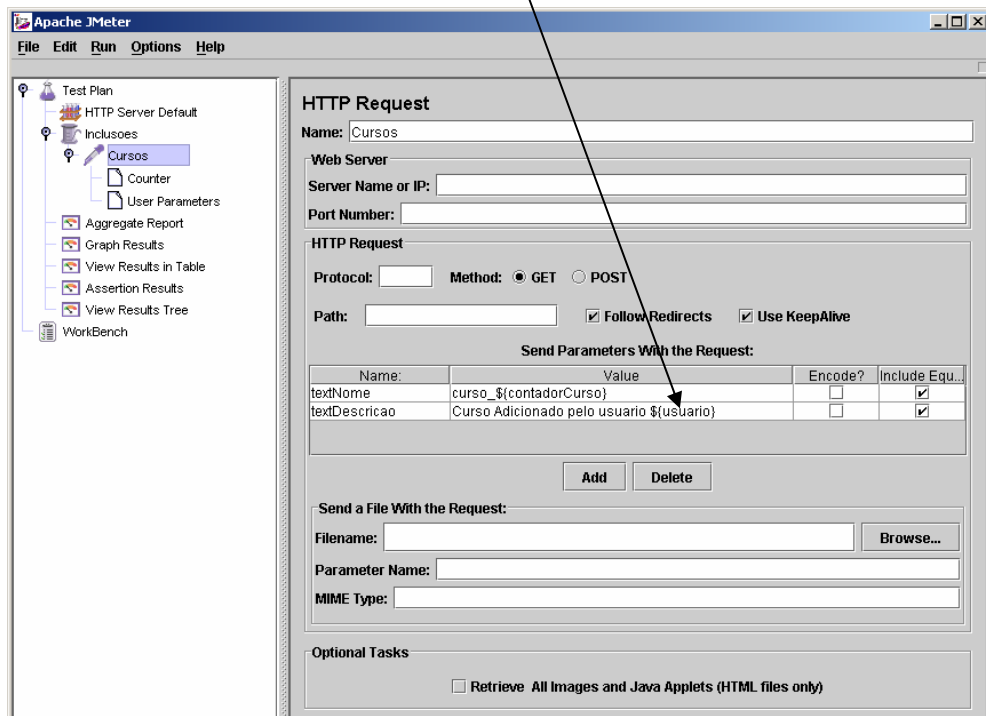
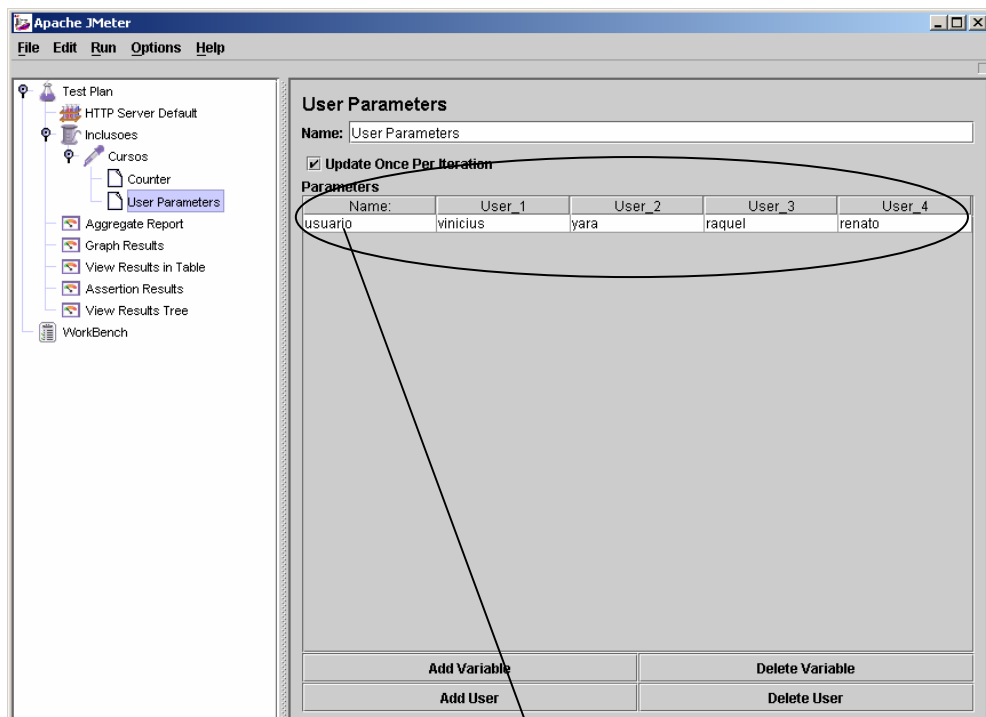
| Pré-processador              | Descrição  |
|------------------------------|--|
| Counter                      | Utilizado para estabelecer um contador. Configuramos seu valor mínimo, máximo, nome da variável e se desejamos um contador independente para cada usuário ou se todos compartilham de um mesmo contador. |
| User Parameters              | Você pode configurar uma variável que tem um valor diferente para cada usuário (thread). Recurso bastante útil.  |
| HTML Link Parser             | Para capturar links retornados por uma página e conteúdo de formulários. Pode fazer spidering no seu Website. Funcionalidade não estabilizada.   |
| HTTP URL Re-writing Modifier | Prove o mecanismo de reescrita de URL para servidores que gerenciam sessões HTTP através desta técnica.  |



Exemplo de uso de pré-processadores:



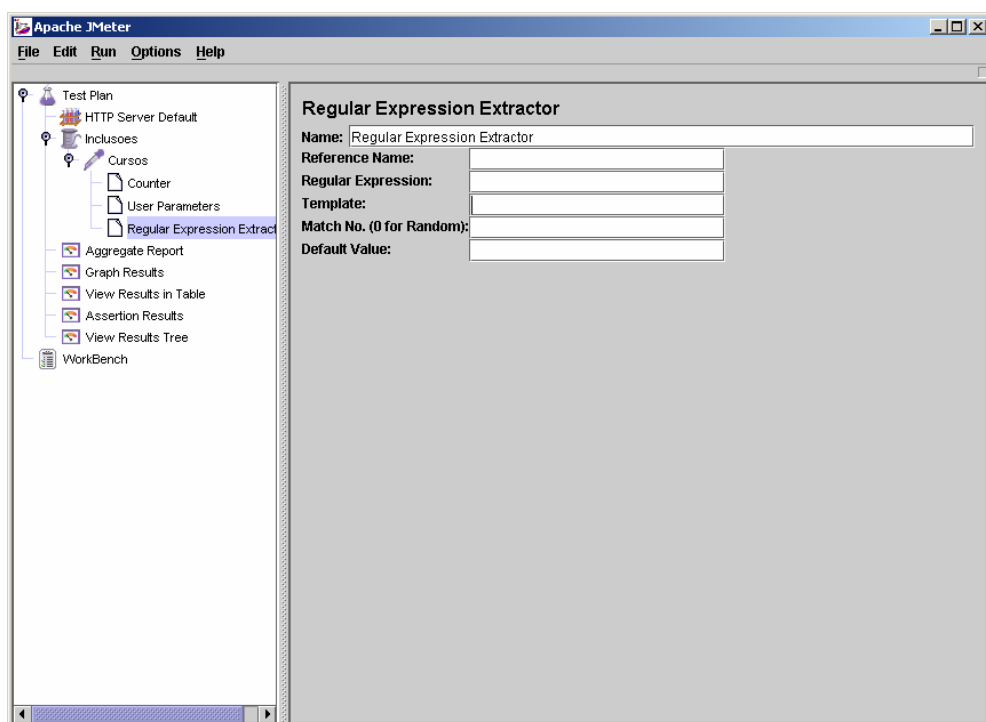
Exemplo de uso de parâmetros or usuário:



## Post Processors

Pós-processadores permitem que você faça a extração de dados da resposta de uma requisição.

Um único tipo de elemento de pós-processamento está disponível na versão atual do JMeter: Regular Expression Extractor. Este extrator permite que você configure um Regular Expression com sintaxe similar a do Perl para obter parte de um resultado.



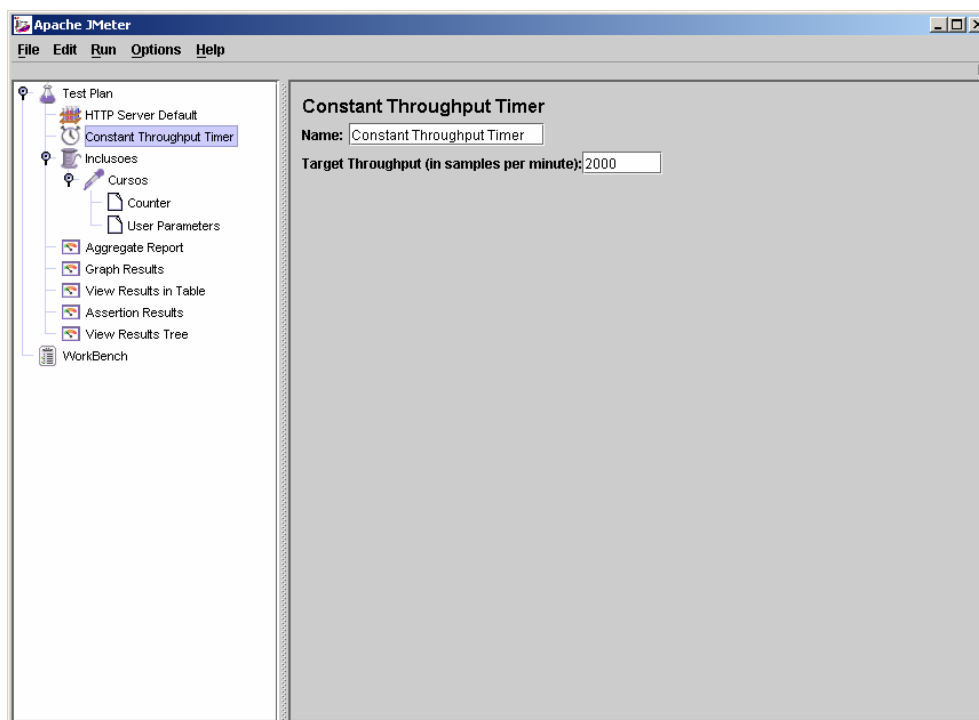
## Timers

Permitem um controle mais preciso no que se refere ao tempo de execução do teste. Podemos, por exemplo, estabelecer um intervalo de tempo padrão entre todas as threads do plano de testes, delays aleatórios entre threads e também controle de frequência de vazão.

Vejamos a seguir os elementos Timer disponíveis:

| Timer                                      | Descrição   |
|--|---|
| Constant Timer                             | Permite que você estabeleça um intervalo em milisegundos padrão entre as threads.   |
| Gaussian Random<br>Uniform Random<br>Timer | Permite configuração de intervalos aleatórios entre threads   |
| Constant<br>Throughput Timer               | Mantém a frequência de acesso de um determinado Sampler. Você pode especificar que determinada atividade você quer executar 100 vezes por segundo, por exemplo. Excelente controlador para quando você quer conhecer o comportamento do data-center frente uma demanda. |

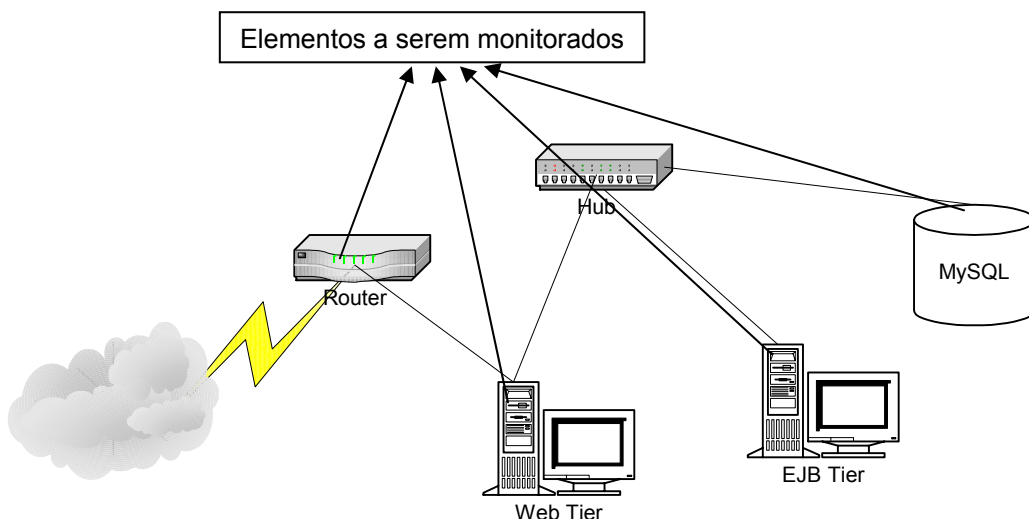
## Exemplo de Timer:



Neste caso temos o Timer de Throughput associado a todo Plano de Testes. Como resultado, o Jmeter vai disparar o número de requisições necessárias para trabalhar com 2000 amostras por segundo. Podemos monitorar a performance do servidor para saber quanto isso consumirá de recursos do nosso data-center.

## Monitoração de Ambiente de Testes

É muito importante observarmos sempre o ambiente que estamos executando o teste de stress. Cada camada da arquitetura da solução deverá ser monitorada, observando um conjunto de fatores que podem gerar “gargalos” na arquitetura.



Temos os seguintes elementos que devemos monitorar:

| Elementos  | Itens de Monitoração  |
|--|---|
| Roteador – Link Wan  | CPU do roteador<br>Link Wan: entrada de pacotes, saída, colisões<br>Memória (dado pouco crítico)  |
| Servidores Físicos:<br>Web Tier, EJB Tier e MySQL                  | CPU do servidor<br>Memória<br>Hard disk<br>Rede<br>I/O<br>Memória virtual   |
| Servidores Lógicos:<br>J2EE Web Server, J2EE<br>EJB Server e MySQL | Não temos uma forma padrão de coletarmos tais dados, cada fabricante oferece uma forma para monitoração lógica do sistema. Com a adoção de JMX (Java Management Extension) isso tende a melhor com containeres J2EE. Um dia poderemos conhecer quanta memória e CPU um EJB está utilizando. |

## Protocolo SNMP

Criado em 1988, o protocolo **S**imple **N**etwork **M**anagement **P**rotocol se tornou um padrão da indústria para monitoração de elementos de rede e foi amplamente aceito por diversas tecnologias.

Padronizado pelo **I**nternet **E**ngineering **T**ask **F**orce o SNMP se encontra na sua terceira versão. Você pode encontrar todos os detalhes da especificação através do site [www.ietf.org](http://www.ietf.org). Suas especificações são padronizadas através de documentos chamados de RFC, a arquitetura geral do SNMP você encontra no RFC de número 3416.

Este protocolo é baseado na arquitetura agente / gerente e é relativamente simples a sua implementação, encorajando os fabricantes a criarem seus agentes. Essencialmente temos um agente para cada tipo de dispositivo e um gerente monitorando ambientes heterogêneos.

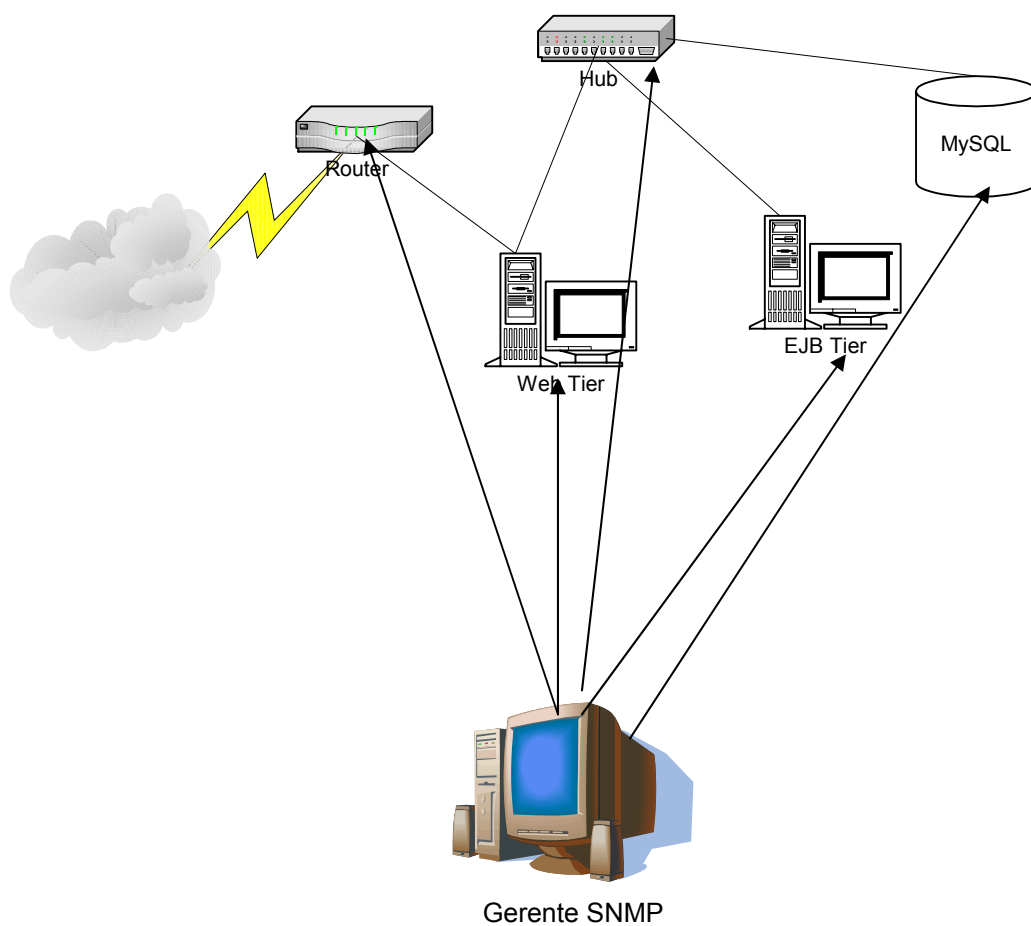
Contamos agentes SNMP para diversos dispositivos físicos e lógicos:

- Roteadores CISCO e 3Com;
- Hubs inteligentes 3Com;
- Servidor Unix;
- Servidor Windows;
- Mainframes;
- Oracle;
- Satélites e bridges;
- Algumas implementações de J2EE, como iPlanet;



## Arquitetura Monitorada

Se contarmos com agentes para todos os dispositivos da nossa rede poderemos monitorar nossa arquitetura da seguinte forma:



## Agente SNMP

Os agentes SNMP podem estar presentes por padrão, como no caso da maioria dos roteadores do mercado, ou então podem estar disponíveis para instalação opcional, como é o caso do agente SNMP para Windows NT, 2000 e XP.

Quase todos os sistemas operacionais, roteadores e switches inteligentes possuem agente SNMP. No caso de servidores J2EE ou banco de dados, você vai precisar consultar documentações para saber se os agentes estão disponíveis.

Pode-se também desenvolver um agente SNMP através de um kit de desenvolvimento. O protocolo SNMP é altamente extensível e existem adoções exóticas deste protocolo, como a monitoração do processo de envelhecimento do whiskey Jack Daniels.

## Gerente SNMP

Um gerente SNMP é geralmente um software desenvolvido por uma empresa ou grupo de trabalho open-source. Contamos com diversas implementações que oferecem diferentes recursos e inteligências.

## Management Information Base

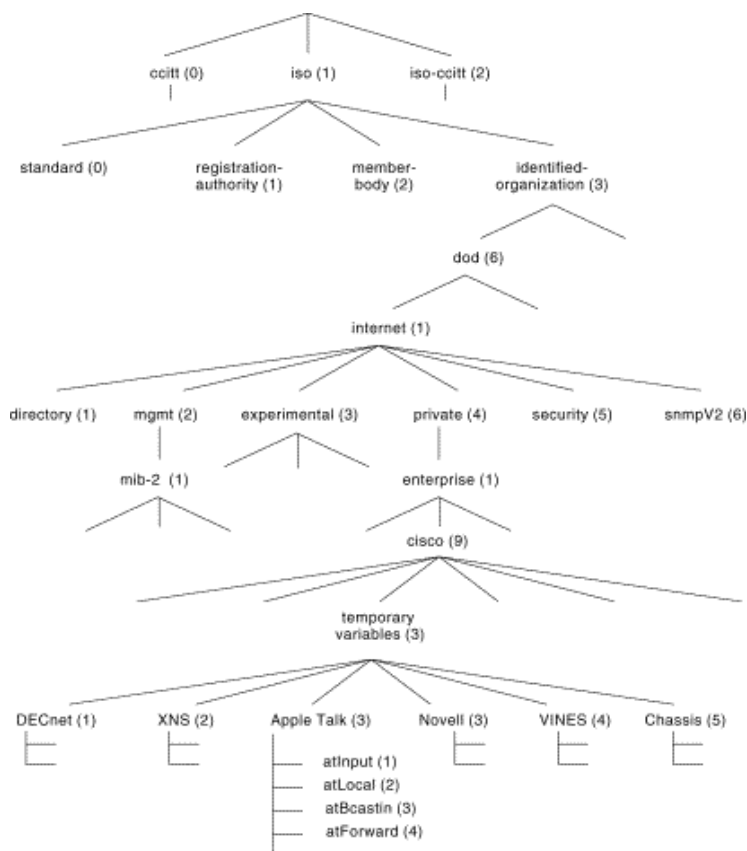
Para cada agente temos um dicionário de informações que podemos coletar sobre aquele agente. Para nos comunicarmos com um roteador Cisco modelo 1122, precisaremos da MIB da Cisco, se queremos monitorar o Windows XP, precisaremos da MIB da Microsoft.

Carregamos as MIBs no software que atua como o gerente SNMP, e este em geral vai estar munido de um MIB Browser que permite que você navegue nas informações do agente e dispare solicitações de informações manualmente.

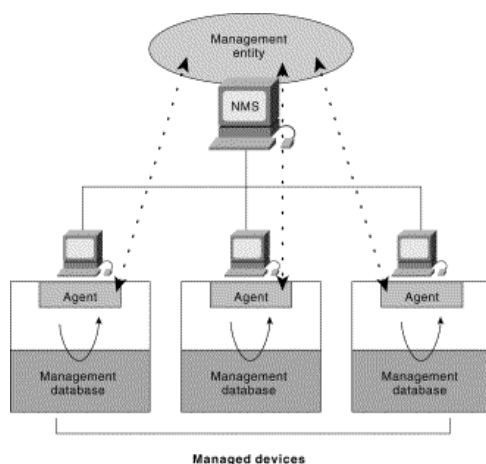
Existem softwares simples que atuam somente como MIB Browser para que você possa descobrir as informações coletáveis de um determinado elemento de rede.

As informações de uma MIB são organizadas e catalogadas através de uma sequência de inteiros que identifica o país, a empresa, o equipamento, o modelo etc.

Vejamos a seguir um gráfico de exemplo.



Perceba que para cada nó da árvore, temos um número. Por exemplo, um roteador Cisco é identificado com o número 1.3.6.1.4.1.9.1.1. Chamamos este número de Object ID, esta sequência indica somente o equipamento, cada novo inteiro colocado a sua direita, identifica uma informação sobre ele.



## Operações SNMP

Foram especificadas as seguintes operações para o protocolo SNMP:

- **get**: obtém um informação de um equipamento, você deve informar um OID;
- **get next**: quando se trata de um dado que contém múltiplas instancias, como por exemplo CPU, precisamos obter um resultado de uma pergunta paa cada CPU do servidor, neste caso utilizamos get next.
- **set**: modifica alguma informação de um elemento. Existe um famoso ataque a roteadores Cisco onde o hacker faz um SNMP Set para derrubar uma interface do roteador.
- **trap**: evento assíncrono, do agente para seu gerente. Podemos configurar um roteador Cisco para ele enviar traps para o gerente, o roteador vai utilizar de traps para informar anormalidades que eventualmente possam acontecer, como uma queda de interface.

## Segurança e características técnicas do SNMP

A primeira versão do SNMP não pode ser considerada seguro, simplesmente possui uma senha plain text que você configura para fazer set e outra para fazer get. Melhorias foram implementadas na segunda versão e finalmente a versão três pode ser considerada segura.

O SNMP é implementado com UDP tornando-o leve e rápido na rede.

Tipicamente configuramos em um agente SNMP:

- Senha para poder executar get e get next;
- Senha para executar set;
- Quem é o seu gerente;

## Ferramentas Comerciais de Monitoração

Contamos no mercado com diversas opções comerciais de framework de monitoração (a maioria baseado em SNMP) com capacidades avançadas, podemos destacar as seguintes funcionalidades:

- Monitoração ativa de ambiente;
- Configuração de alarmes e ações: se uma partição passar de 80% de uso, tente limpar seu tmp, se passar de 90% envie e-mails para administradores Unix e assim por diante;
- Armazenamento de base histórica para análise de plano de capacidade;
- Identificação de gargalos;
- Planejamento de upgrade de hardware;
- Cálculo para simulação de vazão;
- Corelacionamento de eventos;

Podemos destacar as seguintes ferramentas no mercado:

| Fabricante | Ferramenta |
|------------|------------|
| BMC        | Patrol     |
| IBM        | Tivoli     |
| HP         | Openview   |
| Novell     | ManageWise |
| Sun        | Solstice   |

## Ferramentas não-comerciais de Monitoração

Para monitoração de ambientes mais simples podemos contar com ferramentas e comandos de sistema operacional para colher as principais informações sobre o ambiente.

A monitoração com ferramentas gratuitas pode variar bastante de um equipamento para outro. De uma forma maneira simples conseguimos com facilidade monitorar os seguintes itens:

- Servidor Linux: através de comandos de linha ou até mesmo SNMP;
- Servidor Windows: utilizando seu software chamado de Performance Monitor ou então SNMP;
- Mainframes: é comum utilizarem técnicas de geração de arquivos texto com informações sobre o mainframe e disponibiliza-lo via FTP.

Para monitoração de Unix e Linux você pode utilizar os seguintes comandos;

| Comando | Funcionalidade   |
|---------|--|
| sar     | Geralmente não é instalado por padrão, monitora exclusivamente CPU   |
| vmstat  | Monitoração de memória, processos, I/O, swap e CPU.<br>Pode-se utilizar vmstat <número> para que receba atualizações de segundo a segundo. |
| netstat | Comando para monitoração de rede. Podemos utilizar -c para fazer o comando continuamente.  |

Veja o resultado da execução do comando `vmstat 1`:

| procs |   |   | memory |       |        |       | swap |    | io |    | system |     | cpu |    |    |
|-------|---|---|--------|-------|--------|-------|------|----|----|----|--------|-----|-----|----|----|
| r     | b | w | swpd   | free  | buff   | cache | si   | so | bi | bo | in     | cs  | us  | sy | id |
| 0     | 0 | 0 | 0      | 43044 | 321708 | 40348 | 0    | 0  | 0  | 2  | 0      | 10  | 2   | 2  | 8  |
| 0     | 0 | 0 | 0      | 43044 | 321708 | 40348 | 0    | 0  | 0  | 0  | 109    | 121 | 2   | 3  | 95 |
| 0     | 0 | 0 | 0      | 43044 | 321708 | 40348 | 0    | 0  | 0  | 0  | 110    | 119 | 1   | 4  | 95 |
| 0     | 0 | 0 | 0      | 43044 | 321708 | 40348 | 0    | 0  | 0  | 0  | 114    | 126 | 0   | 5  | 95 |
| 0     | 0 | 0 | 0      | 43044 | 321708 | 40348 | 0    | 0  | 0  | 1  | 117    | 119 | 0   | 6  | 94 |
| 0     | 0 | 0 | 0      | 43044 | 321708 | 40348 | 0    | 0  | 0  | 0  | 108    | 117 | 0   | 5  | 95 |
| 0     | 0 | 0 | 0      | 43044 | 321708 | 40348 | 0    | 0  | 0  | 0  | 108    | 117 | 0   | 5  | 95 |

| Coluna      | Informação                                    |
|-------------|---|
| procs ri    | Processos aguardando para serem executados    |
| procs b     | Processos em sleep não-interruptível          |
| procs w     | Processos em swap porém em estado de runnable |
| memory swpd | Quantidade de memória virtual utilizada (kb)  |
| memory free | Quantidade de memória disponível (kb)         |
| memory buff | Quantidade de memória em buffer (kb)          |
| swap si     | Memória em swap do disco                      |
| swap so     | Memória em swap para disco                    |
| io bi       | Blocos enviados para um dispositivo de I/O    |
| io bo       | Blocos recebidos de um dispositivo de I/O     |
| system in   | Número de interrupções por segundo            |
| system cs   | Número de troca de contexto por segundo       |
| cpu us      | Porcentagem de uso da CPU para usuários       |
| cpu sy      | Porcentagem de uso da CPU para o sistema      |
| cpu id      | Porcentagem da CPU disponível                 |

O comando `vmstat` é bem completo apesar de ter um foco principal na memória virtual. Seu único problema é que não temos a data e hora da coleta da informação.

Podemos ligar o comando e direcionar o resultado para um arquivo:

```
nohup vmstat 1 &
```

Este comando vai gerar um arquivo chamado de `nohup.out` com os resultados.

Poderíamos programar um script utilizando bash para imprimirmos também a hora da coleta:

### Script bash para coleta de informações

---

```
#!/bin/bash

echo Monitoração personalizada

while [ true ]
do
    echo =====
    date
    vmstat
    sleep 1
done
```

---

Coloque o conteúdo acima em um arquivo chamado `monitora.sh`, para executá-lo digite:

```
nohup monitora.sh &
```

Dicas:

É bastante simples transformar este script em um serviço do Linux.

Mude o atributo de execução do script: `chmod 755 monitora.sh`



## Plano de Capacidade e Relatório de Conclusões

Já aprendemos nos capítulos anteriores a:


- Capturar requisitos;
- Projetar um plano de stress-test documentado;
- Utilizar a ferramenta Apache Jmeter para simular demandas;
- Monitorar servidores e equipamentos diversos de um data-center;

Com o resultado gerado pelo planejamento, JMeter e pela monitoração, temos uma base concreta para estudarmos a capacidade da arquitetura que adotamos na solução.

Devemos reunir no nosso Relatório de Conclusões e Plano de Capacidade as seguintes informações:

- Documentação sobre previsão de uso da solução. Aprendemos no capítulo II a capturar as informações através de Metodologia para Testes de Capacidade.
- Documentação do ambiente físico utilizado: servidores, rede, memória, hard disk etc.
- Documentação do ambiente lógico: sistema operacional, máquina virtual, servidor de aplicação, entre outros.
- Detalhes do plano de stress elaborado com a ferramenta em questão.
- Informações coletadas na monitoração dos servidores e dispositivos que participaram do teste.
- Relatório de Conclusões.
- Plano de Capacidade e Recomendações Técnicas.

Vejamos a seguir um exemplo completo de um relatório de stress-test que recomendamos como modelo caso você / sua empresa ainda não tenha nenhum pré-definido.

|     |  |  |
|-----|--|--|
|     | <b>Relatório de Análise de Capacidade – Parte 1/7</b><br><i>Análise de Requisitos não-funcionais</i> |  |
| AA4 | Planning & Performing Stress-test  |  |


### **Análise de Requisitos não-funcionais**

**Nome do Projeto:** Estudo de Caso Academia do Arquiteto

**Data coleta dos dados:** 1/1/2000

### **Perguntas**

- *Qual é a demanda prevista para usar a solução?*  
R. 200 acessos por dia em média.
- *Existe possibilidade de picos? Se sim, qual o pico previsto?*  
R. Sim. Podemos chegar a um pico de 100 usuários simultâneos.
- *Qual é o tempo de resposta desejado?*  
R. O nível ideal de trabalho é que o usuário não espere mais do que 2 segundos por cada resposta.
- *Os acessos durante o dia vão se concentrar mais em um horário específico?*  
R. Sim, 80% devem ocorrer entre as 11:00 e 21:00.
- *Existe um nivelamento no acesso durante a semana ou existe um período de maior concentração?*  
R. Segundas e terças, acesso baixo, 100 acessos por dia.  
Quartas, quintas, médio acesso com 150 acessos por dia.  
Sextas e sábados, pico de 300 acessos por dia.  
Domingo, 200 por dia.
- *Quanto seu negócio tende a crescer no próximo ano e no ano seguinte dentro do cenário otimista do seu business plan?*  
R. Pretendemos crescer 15% neste ano e 30% no próximo.
- *O negócio que a solução atende pode apresentar variações extremas e situações atípicas com que grau de frequência?*  
R. 5%.


|     |  |  |
|-----|--|--|
|     | <b>Relatório de Análise de Capacidade – Parte 1/7</b><br><i>Análise de Requisitos não-funcionais</i> |  |
| AA4 | Planning & Performing Stress-test  |  |

#### Distribuição de utilização da solução por use-case

| Use-case   | % de acesso |
|--|-------------|
| Secretaria mantém cursos<br>(adiciona, exclui, inclui e pesquisa)  | 5%          |
| Secretaria mantém turmas<br>(criação de turmas, exclusão, matrícula de alunos e modificação de matrículas) | 35%         |
| Secretaria mantém memberships  | 50%         |
| Alunos se cadastram no sistema via Web   | 5%          |
| Alunos efetuam matrículas via Web  | 5%          |

#### Cálculo de Volume de Dados por Entidade

| Entidade   | Atual | Ano 1 | Ano 2 |
|------------|-------|-------|-------|
| Curso      | 25    | 35    | 50    |
| Membership | 4000  | 6000  | 8000  |
| Turma      | 40    | 100   | 300   |
| Matricula  | 600   | 1800  | 4000  |

|     |   |  |
|-----|---|--|
|     | <b>Relatório de Análise de Capacidade – Parte 2/7</b><br><i>Documentação do Ambiente Físico</i> |  |
| AA4 | Planning & Performing Stress-test   |  |

## Documentação do Ambiente Físico

Utilizamos três computadores para executar os testes:


- **Node #1:** Uma estação PC Windows XP com alta capacidade de processamento para atuar como simulador de clientes com a ferramenta Apache JMeter;
- **Node #2:** Um PC Linux Red Hat 9.1 com o middleware J2EE JBoss 3.2.2;
- **Node #3:** Um PC Linux Red Hat 9.1 com o servidor de banco de dados MySQL 4.0.1;

Na tabela a seguir detalhamos as configurações de hardware de cada elemento utilizado no teste.

| Elemento | Detalhes  |
|----------|---|
| Node #1  | Intel Pentium 4 2.80 GHz cachê 256KB<br>60GB (5400 rpm)<br>512 MB DDR SDRAM   |
| Node #2  | AMD Athlon XP2200+ 1800MHz cachê 256KB<br>40GB (5400 rpm)<br>512 MB DDR SDRAM |
| Node #3  | AMD Athlon XP2200+ 1800MHz cachê 256KB<br>40GB (5400 rpm)<br>512 MB DDR SDRAM |

Velocidade da rede: 100 megabits;


HUB Genérico;

|     |   |  |
|-----|---|--|
|     | <b>Relatório de Análise de Capacidade – Parte 3/7</b><br><i>Documentação do Ambiente Lógico</i> |  |
| AA4 | Planning & Performing Stress-test   |  |

### Documentação do Ambiente Lógico

Neste tópico documentamos a versão de cada um dos softwares e componentes lógicos em geral que participaram do teste.

| Elemento | Software                | Nome e Versão  |
|----------|-------------------------|--|
| Node #1  | Apache JMeter           | 1.9.1  |
| Node #1  | Java Development Kit    | Java 2 Standard Edition 1.4.2-b28                                      |
| Node #1  | Sistema Operacional     | Windows XP 5.1 build 2600.xpp2<br>Service Pack 1                       |
| Node #2  | Application Server j2EE | JBoss 3.2.2 (Wonderland)   |
| Node #2  | Driver JDBC MySQL       | Fonte <a href="http://www.mysql.com">www.mysql.com</a> – versão 3.0.10 |
| Node #2  | Java Development Kit    | Java 2 Standard Edition 1.4.2-b28                                      |
| Node #2  | Pooling de conexões     | 25 conexões configuradas no JBoss                                      |
| Node #2  | Sistema Operacional     | Red Hat Linux 9.1 2.4.20-8   |
| Node #3  | MySQL Server            | 4.0.13   |
| Node #3  | Sistema Operacional     | Red Hat Linux 9.1 2.4.20-8   |

|     |  |  |
|-----|--|--|
|     | <b>Relatório de Análise de Capacidade – Parte 4/7</b><br><i>Detalhes do Plano de Stress-test</i> |  |
| AA4 | Planning & Performing Stress-test  |  |

### **Detalhes do Plano de Stress-test**


Elaboramos com a ferramenta JMeter um script que simula o uso da aplicação conforme dados coletados na tabela de previsão distribuição de uso de requisitos.

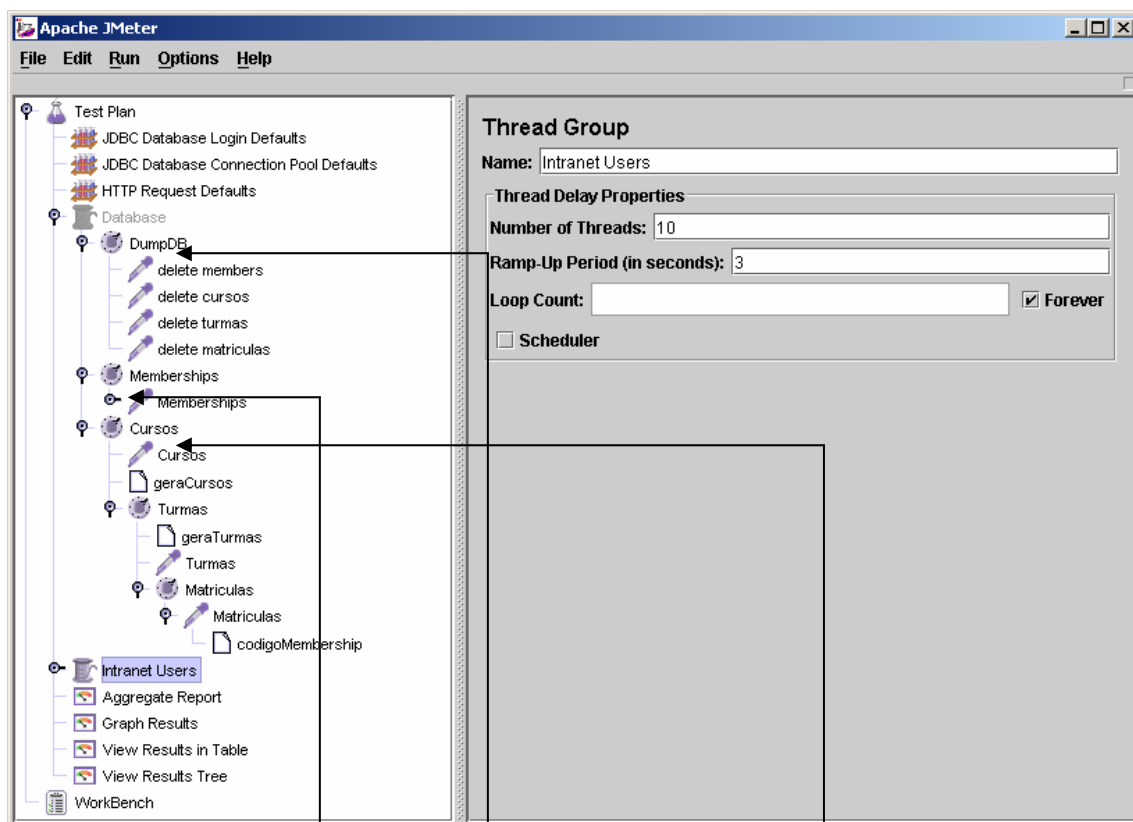
Envolvemos neste teste todas as funcionalidades que representam 80% dos requisitos do aplicativo.

Nosso script JMeter pode ser dividido em duas partes:

1. Geração de massa de dados;
2. Simulação de usuários;


A primeira parte é responsável por gerar uma massa inicial para podermos testar operações de edição de dados desde o começo do teste.

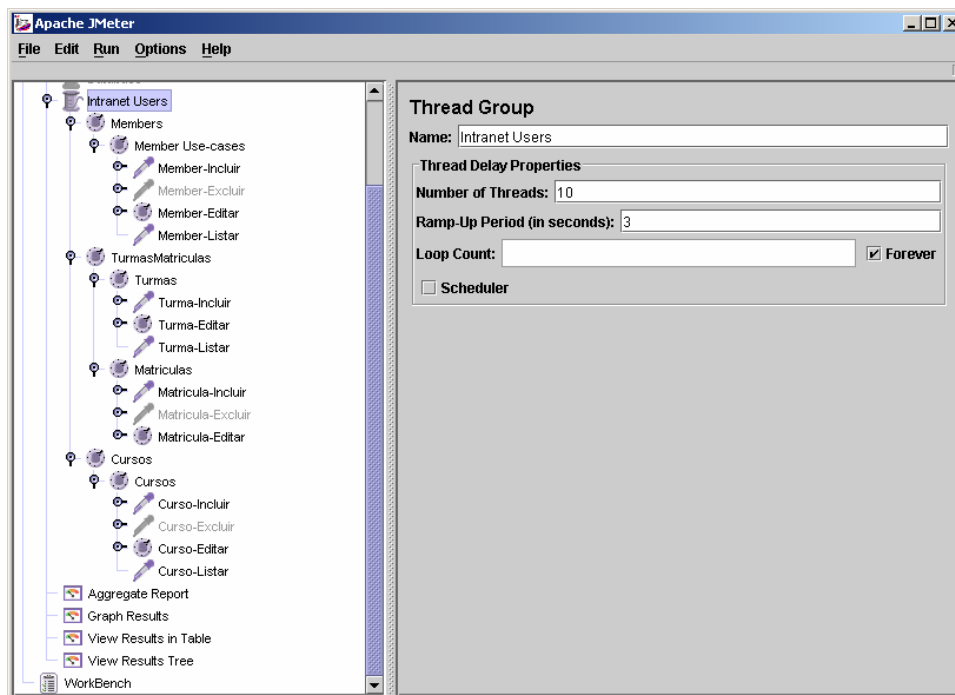
|     |  |   |
|-----|--|---|
|     | <b>Relatório de Análise de Capacidade – Parte 4/7</b><br><i>Detalhes do Plano de Stress-test</i> |  |
| AA4 | Planning & Performing Stress-test  |   |



As operações envolvidas no controller **DumpDB** limpam a tabela e seus numeradores de auto-incremento, em **Memberships** criamos um cadastro inicial de alunos.

Em seguida temos as operações em cascata no controller **Cursos**. Para cada curso adicionado,  $n$  turmas são adicionadas para o curso e  $n$  matrículas são adicionadas na turma. É uma operação de geração de massa em cascata que exige bastante cuidado no planejamento, pois tais dados serão utilizados pelas ações de alteração e exclusão disparadas pelo JMeter.


|     |  |  |
|-----|--|--|
|     | <b>Relatório de Análise de Capacidade – Parte 4/7</b><br><i>Detalhes do Plano de Stress-test</i> |  |
| AA4 | Planning & Performing Stress-test  |  |



Na segunda parte temos um Thread Group principal nomeada de Internet Users. Foram utilizados controladores de vazão para agrupar um conjunto de tarefas.

| Elemento   | Função  |
|--|---|
| Members  | Throughput Controller: as tarefas agrupadas neste controlador deverão representar 50% das requisições enviadas ao servidor. |
| Member Use-cases   | Random Controller.  |
| Member Incluir, Excluir, Editar e Listar                             | Simulam as operações disponíveis para a entidade membership.  |
| TurmasMatriculas   | Throughput Controller: as tarefas agrupadas neste controlador deverão representar 45% das requisições enviadas ao servidor. |
| Turmas   | Random Controller.  |
| Turmas Incluir, Editar, Listar, Matriculas Incluir, Excluir e Editar | Simulam as operações disponíveis para as entidades turmas e matrículas.   |
| Cursos   | Throughput Controller: as tarefas agrupadas neste controlador deverão representar 5% das requisições enviadas ao servidor.  |
| Cursos -> Cursos   | Random Controller.  |
| Cursos Incluir etc.  | Simulam as operações da entidade curso.   |



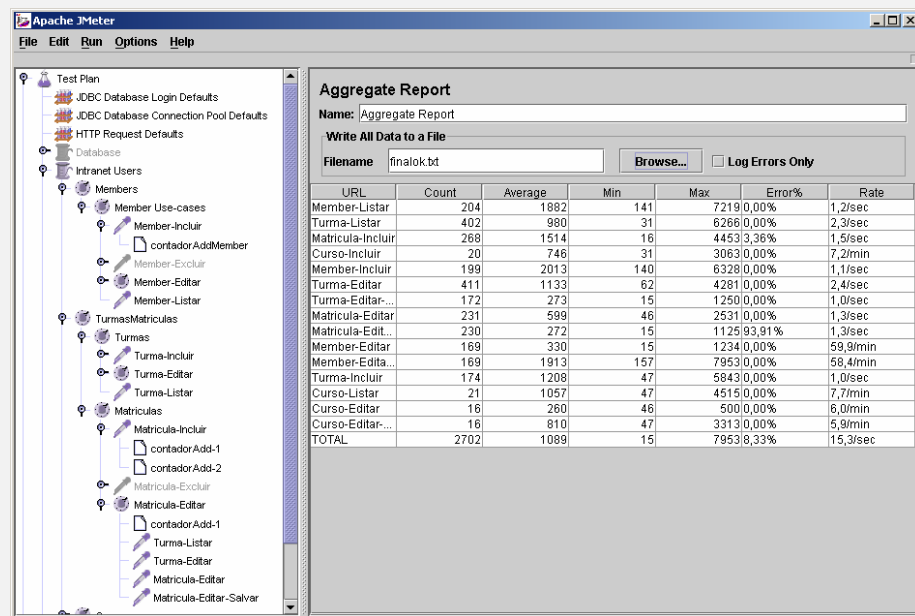
|     |  |   |
|-----|--|---|
|     | <b>Relatório de Análise de Capacidade – Parte 5/7</b><br><i>Informações Coletadas na Monitoração</i> |  |
| AA4 | Planning & Performing Stress-test  |   |

### Informações Coletadas na Monitoração do Ambiente

Apresentamos um resumo do estado geral dos servidores e elementos de rede envolvidos nos testes. Os arquivos reais de monitoração estão disponíveis no CD entregue como parte do relatório.


O período de observação foi de 3 minutos com carga total de 80 usuário com intervalo de 5 em 5 segundos entre as operações de cada usuário.

| Servidor | Resumo   |
|----------|--|
| Node #3  | O servidor apresentou-se totalmente estável e com capacidade de escalar se necessário. O consumo de CPU não passou de 15% durante todo o período de monitoração e não foi observado crescimento no uso da memória.   |
| Node #2  | O servidor apresentou-se totalmente estável e com capacidade de escalar se necessário. O consumo de CPU ficou em 30% na média e notamos somente alguns picos.<br>Houve remanejamento de 4 megabytes de memória para cachê e tivemos uma redução de 3 megas de memória no servidor. |
| Node #1  | Veja na imagem algumas estatísticas geradas pelo JMeter  |



The screenshot shows the Apache JMeter 'Aggregate Report' window. On the left is a tree view of the test plan, including sections for Test Plan, JDBCs, HTTP Request Defaults, Database, Intranet Users, Members, Turmas, and Matriculas. The main area displays a table of test results for various URLs, including Member-Listar, Turma-Listar, Matricula-Incluir, Curso-Incluir, Member-Incluir, Turma-Editar, Matricula-Editar, and others. The table columns are URL, Count, Average, Min, Max, Error%, and Rate. The 'TOTAL' row shows a count of 2702, an average of 1089, and a maximum of 15.

| URL               | Count | Average | Min | Max  | Error% | Rate     |
|-------------------|-------|---------|-----|------|--------|----------|
| Member-Listar     | 204   | 1882    | 141 | 7219 | 0,00%  | 1,2/sec  |
| Turma-Listar      | 402   | 980     | 31  | 6266 | 0,00%  | 2,3/sec  |
| Matricula-Incluir | 268   | 1514    | 16  | 4453 | 3,36%  | 1,5/sec  |
| Curso-Incluir     | 20    | 746     | 31  | 3063 | 0,00%  | 7,2/min  |
| Member-Incluir    | 199   | 2013    | 140 | 6328 | 0,00%  | 1,1/sec  |
| Turma-Editar      | 411   | 1133    | 62  | 4281 | 0,00%  | 2,4/sec  |
| Matricula-Editar  | 172   | 273     | 15  | 1260 | 0,00%  | 1,0/sec  |
| Matricula-Editar  | 231   | 599     | 46  | 2531 | 0,00%  | 1,3/sec  |
| Matricula-Editar  | 230   | 272     | 15  | 1125 | 93,91% | 1,3/sec  |
| Member-Editar     | 169   | 330     | 15  | 1234 | 0,00%  | 59,9/min |
| Member-Editar     | 169   | 1913    | 157 | 7953 | 0,00%  | 59,9/min |
| Turma-Incluir     | 174   | 1208    | 47  | 5843 | 0,00%  | 1,0/sec  |
| Curso-Listar      | 21    | 1057    | 47  | 4515 | 0,00%  | 7,7/min  |
| Curso-Editar      | 16    | 260     | 46  | 500  | 0,00%  | 6,0/min  |
| Curso-Editar      | 16    | 810     | 47  | 3313 | 0,00%  | 5,9/min  |
| TOTAL             | 2702  | 1089    | 15  | 7953 | 8,33%  | 15,3/sec |

|     |   |   |
|-----|---|---|
|     | <b>Relatório de Análise de Capacidade – Parte 6/7</b><br><i>Relatório de Conclusões</i> |  |
| AA4 | Planning & Performing Stress-test   |   |

## Relatório de Conclusões

Com as cargas programadas no JMeter conseguimos atingir mais de 900 requisições por minuto, o que significam 16 requisições por segundo. A média do tempo de resposta está dentro do esperado, ou seja, no máximo 2 segundos.

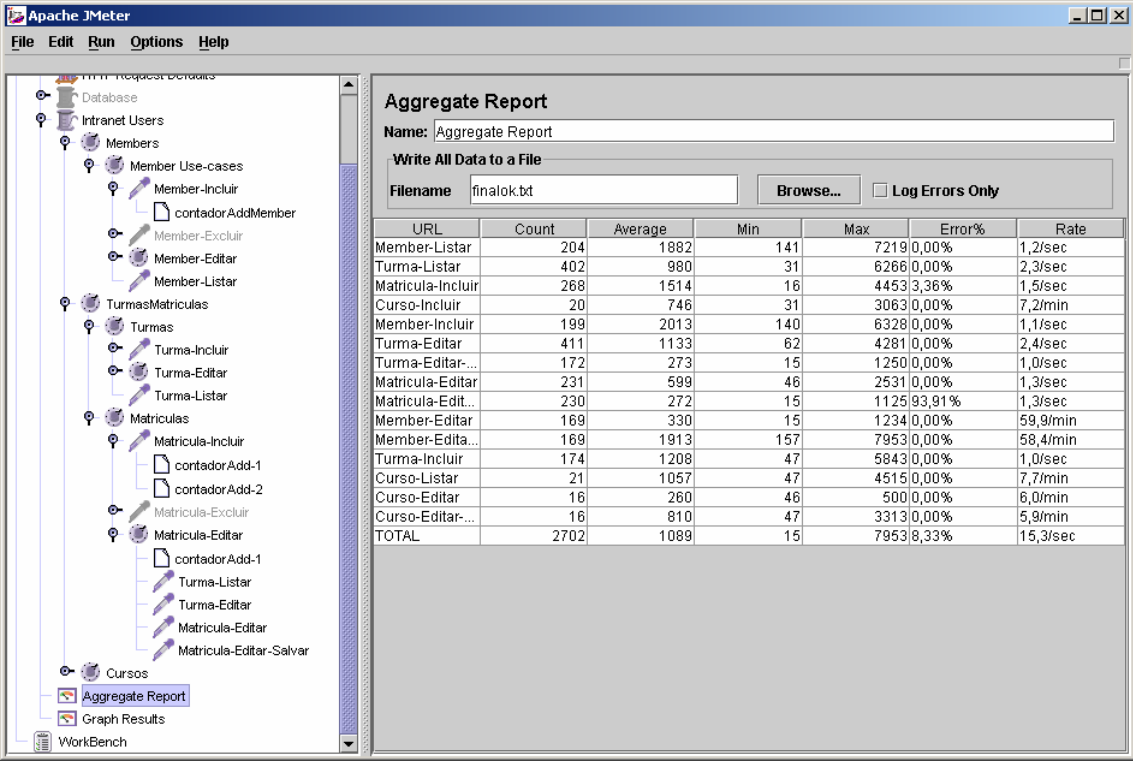
- Tempo total de teste: 3 minutos;
- Vazão aferida: 921 requisições por minuto;
- Número de transações efetuadas em 3 minutos: 2702;

Considerando que o intervalo do usuário entre as solicitações efetuadas seja de 5 segundos, temos os seguintes resultados:

Média do número de requisições por minuto de um usuário: 12 requisições por minuto.


Capacidade / Vazão do sistema: 970 requisições por minuto.

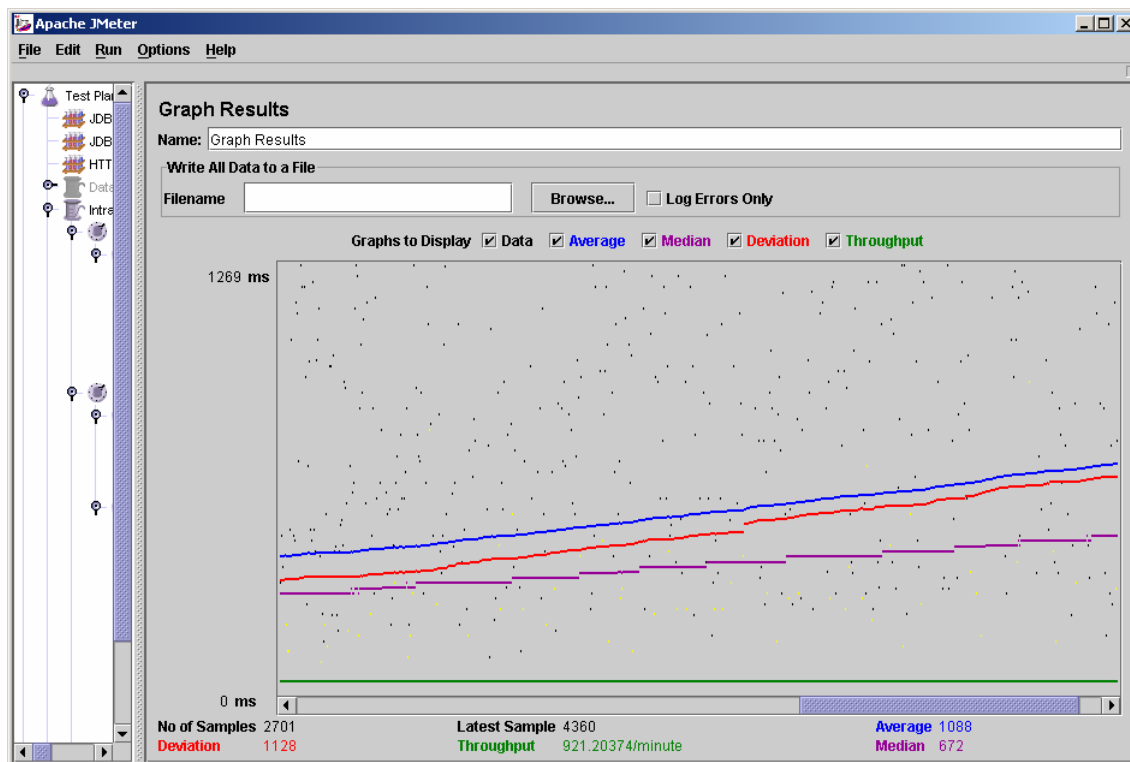
Total de usuários simultâneos:  $970 / 12 = 80,83$




The screenshot shows the Apache JMeter interface with the 'Aggregate Report' window open. The report displays the following data:

| URL               | Count       | Average     | Min       | Max         | Error%       | Rate            |
|-------------------|-------------|-------------|-----------|-------------|--------------|-----------------|
| Member-Listar     | 204         | 1882        | 141       | 7219        | 0,00%        | 1,2/sec         |
| Turma-Listar      | 402         | 980         | 31        | 6266        | 0,00%        | 2,3/sec         |
| Matricula-Incluir | 268         | 1514        | 16        | 4453        | 3,36%        | 1,5/sec         |
| Curso-Incluir     | 20          | 746         | 31        | 3063        | 0,00%        | 7,2/min         |
| Member-Incluir    | 199         | 2013        | 140       | 6328        | 0,00%        | 1,1/sec         |
| Turma-Editar...   | 411         | 1133        | 62        | 4281        | 0,00%        | 2,4/sec         |
| Turma-Editar...   | 172         | 273         | 15        | 1250        | 0,00%        | 1,0/sec         |
| Matricula-Editar  | 231         | 599         | 46        | 2531        | 0,00%        | 1,3/sec         |
| Matricula-Edit... | 230         | 272         | 15        | 1125        | 93,91%       | 1,3/sec         |
| Member-Editar     | 169         | 330         | 15        | 1234        | 0,00%        | 59,9/min        |
| Member-Edita...   | 169         | 1913        | 157       | 7953        | 0,00%        | 58,4/min        |
| Turma-Incluir     | 174         | 1208        | 47        | 5843        | 0,00%        | 1,0/sec         |
| Curso-Listar      | 21          | 1057        | 47        | 4515        | 0,00%        | 7,7/min         |
| Curso-Editar      | 16          | 260         | 46        | 500         | 0,00%        | 6,0/min         |
| Curso-Editar...   | 16          | 810         | 47        | 3313        | 0,00%        | 5,9/min         |
| <b>TOTAL</b>      | <b>2702</b> | <b>1089</b> | <b>15</b> | <b>7953</b> | <b>8,33%</b> | <b>15,3/sec</b> |

|     |   |   |
|-----|---|---|
|     | <b>Relatório de Análise de Capacidade – Parte 6/7</b><br><i>Relatório de Conclusões</i> |  |
| AA4 | Planning & Performing Stress-test   |   |



Conforme observamos no gráfico acima, o sistema apresentou uma constância na vazão, notamos uma pequena queda do meio do teste para frente. A queda foi de 1050 requisições por minuto para 921. Isso se deve ao fato de que não temos paginação dos dados quando visualizamos memberships, cursos e turmas. Portanto a medida que o sistema adiciona novos dados, a página de visualização cresce.

|     |   |  |
|-----|---|--|
|     | <b>Relatório de Análise de Capacidade – Parte 7/7</b><br><i>Plano de Capacidade e Recomendações</i> |  |
| AA4 | Planning & Performing Stress-test   |  |

### **Plano de Capacidade e Recomendações**

A arquitetura apresentada revelou que atende aos requisitos não-funcionais considenrado o máximo de 100 usuários concorrentes em picos esporádicos. Em geral a aplicação está suportando 80 usuários concorrentes e o tempo de resposta está com média de 800 millesegundos, 1500 milisegundos no máximo e estabilidade no desvio padrão.

Para os próximos dois anos o servidor de aplicativos, banco de dados e arquitetura J2EE adotada estará atendendo aos requisitos do negócio da empresa.

Para aumentar a escalabilidade sugerimos a consideração do uso de cluster de Web Containers tendo em vista que o servidor de banco de dados está bastante ocioso e pode comporta mais um Web Server solicitando dados.

Notamos uma queda de performance no decorrer do uso do aplicativo que pode ocasionar em um aumento de latência significativo. A principal causa do problema é a falta de paginação de dados quando o usuário solicita a visualização de memberships, cursos e turmas. Pela falta de uso de paginação, esta página tende a crescer a medida que se inclui novos dados.

Recomenda-se fortemenete o refactoring da aplicação e uso de Value List Handler para solucionar o problema.