

# Portal para Submissão de Tarefas Utilizando Grade Computacional Oportunista

Diogo Edegar Mafrá - Acadêmico  
Prof. Paulo Fernando da Silva - Orientador



# Roteiro

- Introdução
- Objetivos
- Fundamentação teórica
- Especificação
- Implementação
- Operacionalidade da implementação
- Resultados e discussão
- Conclusão
- Extensões



# Introdução

- O poder de processamento dos computadores atuais é bastante grande
- A demanda por processamento aumentou nos últimos anos
- Supercomputadores continuam sendo são bastante caros
- Em média, cerca de 10% do poder de processamento de um computador é utilizado
- Uma grade computacional oportunista permite o uso do poder computacional desperdiçado



# Objetivos

- Permitir a submissão, por meio de um portal web, de tarefas a serem processadas
- Distribuir o processamento das tarefas entre os computadores conectados à grade
- Permitir o uso do tempo ocioso de computadores para executar o processamento das tarefas
- Permitir o acompanhamento do estado de execução das tarefas
- Permitir a obtenção dos resultados da execução da tarefa submetida



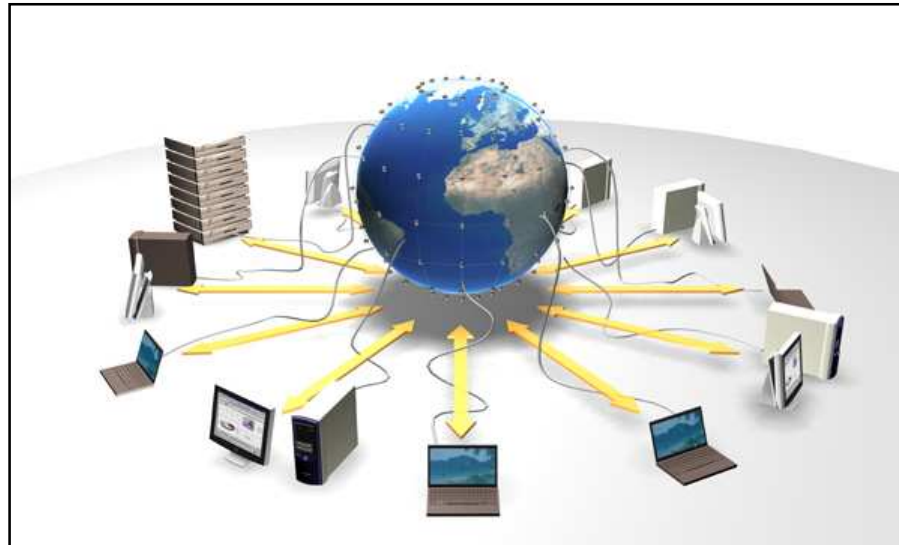
# Fundamentação teórica

- Grades computacionais
- Aplicações do tipo bag-of-tasks
- Escalonamento das tarefas



# Grades computacionais

- Múltiplos domínios administrativos
- Heterogeneidade
- Escalabilidade
- Dinamicidade e adaptabilidade



# Aplicações do tipo bag-of-tasks

- Tarefas independentes entre si
- Adéqua-se perfeitamente ao processamento em grade
- Usada em cálculos fractais, biologia computacional e processamento de imagens



# Escalonamento das tarefas

O escalonamento é a distribuição das tarefas entre os computadores da grade.

Desafios do escalonamento em uma grade computacional:

- **Heterogeneidade** – equipamento com características diversas
- **Dinamicidade** – inclusão e remoção de máquinas a qualquer momento
- **Dificuldade na estimativa** do tempo de execução das tarefas





# Escalonamento das tarefas

## Algoritmo **Workqueue**:

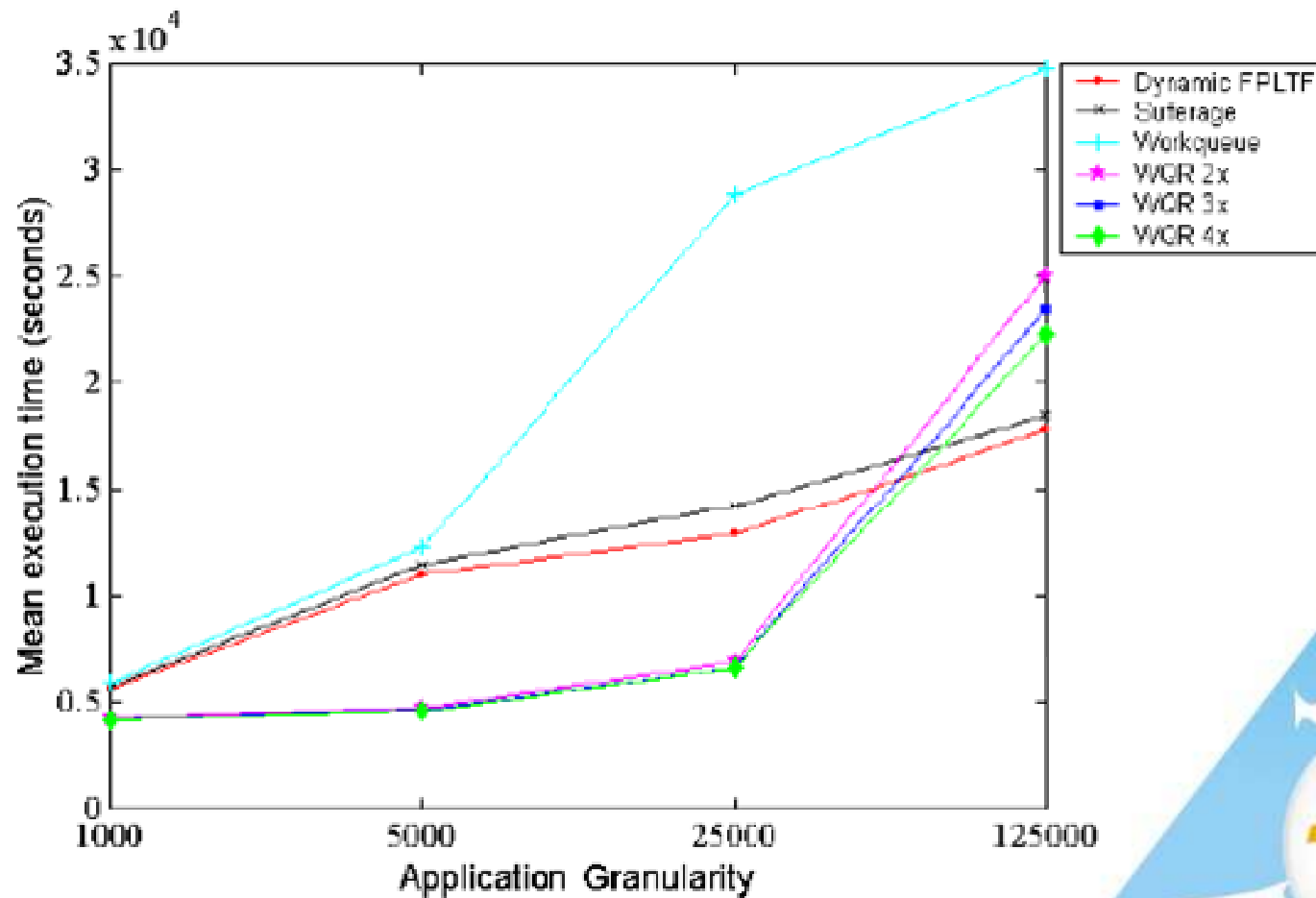
- Não depende de informações das tarefas
- Simplesmente cria uma fila de tarefas
- Máquinas mais rápidas processarão mais tarefas
- Uma máquina lenta pode receber uma tarefa grande e demorar bastante para processá-la

## Algoritmo **Workqueue com Replicação**:

- Replica as tarefas em máquinas que estão em espera
- O número de réplicas é pré-definido
- O tempo de finalização é menor
- Ocorre desperdício de processamento



# Escalonamento das tarefas



# Trabalhos correlatos



# Especificação

- Requisitos funcionais
  - permitir a submissão de tarefas por meio de um portal web
  - permitir o acompanhamento do estado da execução das tarefas
  - permitir a obtenção dos resultados da execução da tarefa submetida
  - distribuir, entre os computadores participantes da grade, as tarefas submetidas pelo portal
  - utilizar o tempo ocioso dos computadores conectados à grade para processar as tarefas
  - tornar disponível uma API a ser utilizada pelas tarefas submetidas à grade



# Especificação

- **Requisitos não-funcionais**

- restringir as permissões de execução da tarefa, impedindo a execução de códigos maliciosos nos computadores participantes da grade
- ser implementado utilizando o ambiente de desenvolvimento Microsoft Visual Studio 2010;
- ser implementado utilizando a linguagem C#
- utilizar a biblioteca ASP.NET MVC 2 para a construção do portal web
- funcionar no sistema operacional Windows XP ou superior
- utilizar banco de dados Microsoft SQL Server 2005 para armazenamento de dados utilizados pelo portal web e tarefas submetidas a grade
- as tarefas submetidas ao portal devem ser desenvolvidas utilizando o Microsoft .NET Framework

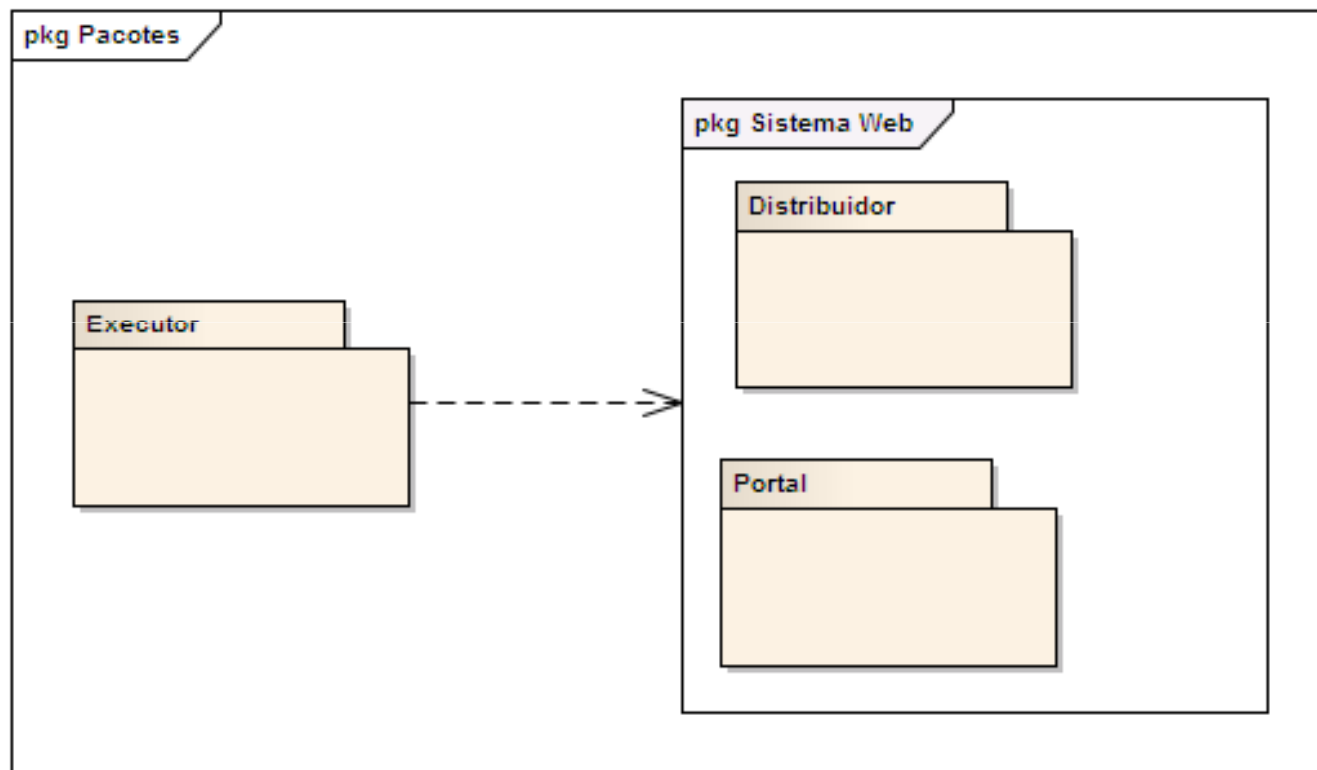


# Especificação

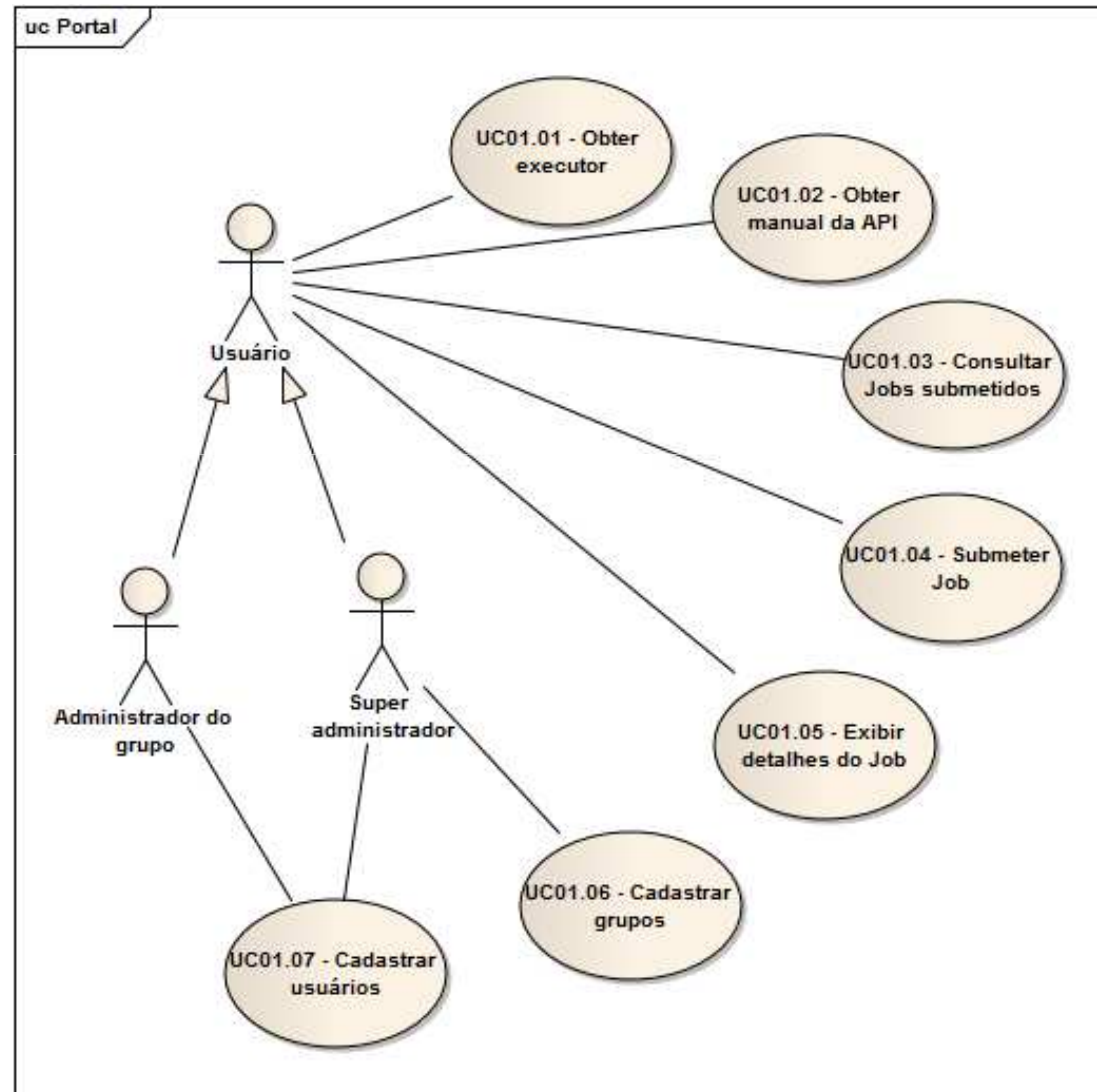
- Partes do sistema
- Casos de uso
- API para submissão de tarefas
- Formato do Job
- Grupos
- Diagramas de Classe



# Partes do sistema

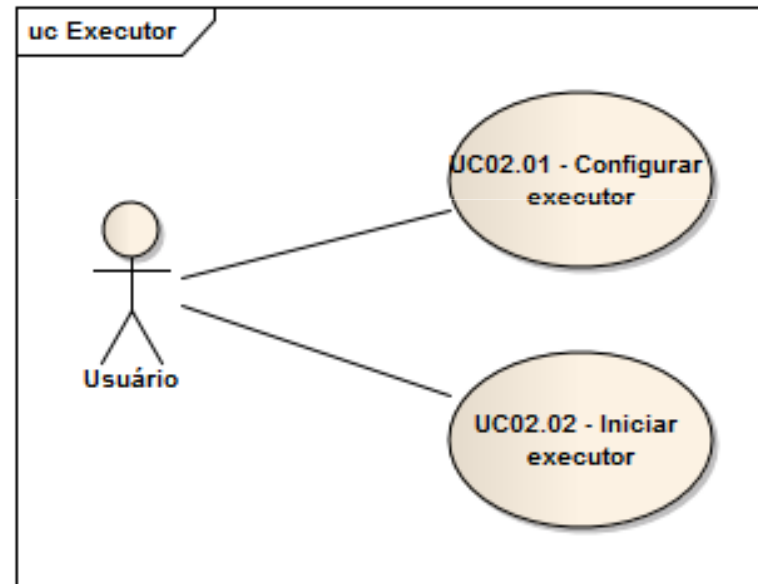


# Casos de uso - Portal





# Casos de uso - Executor

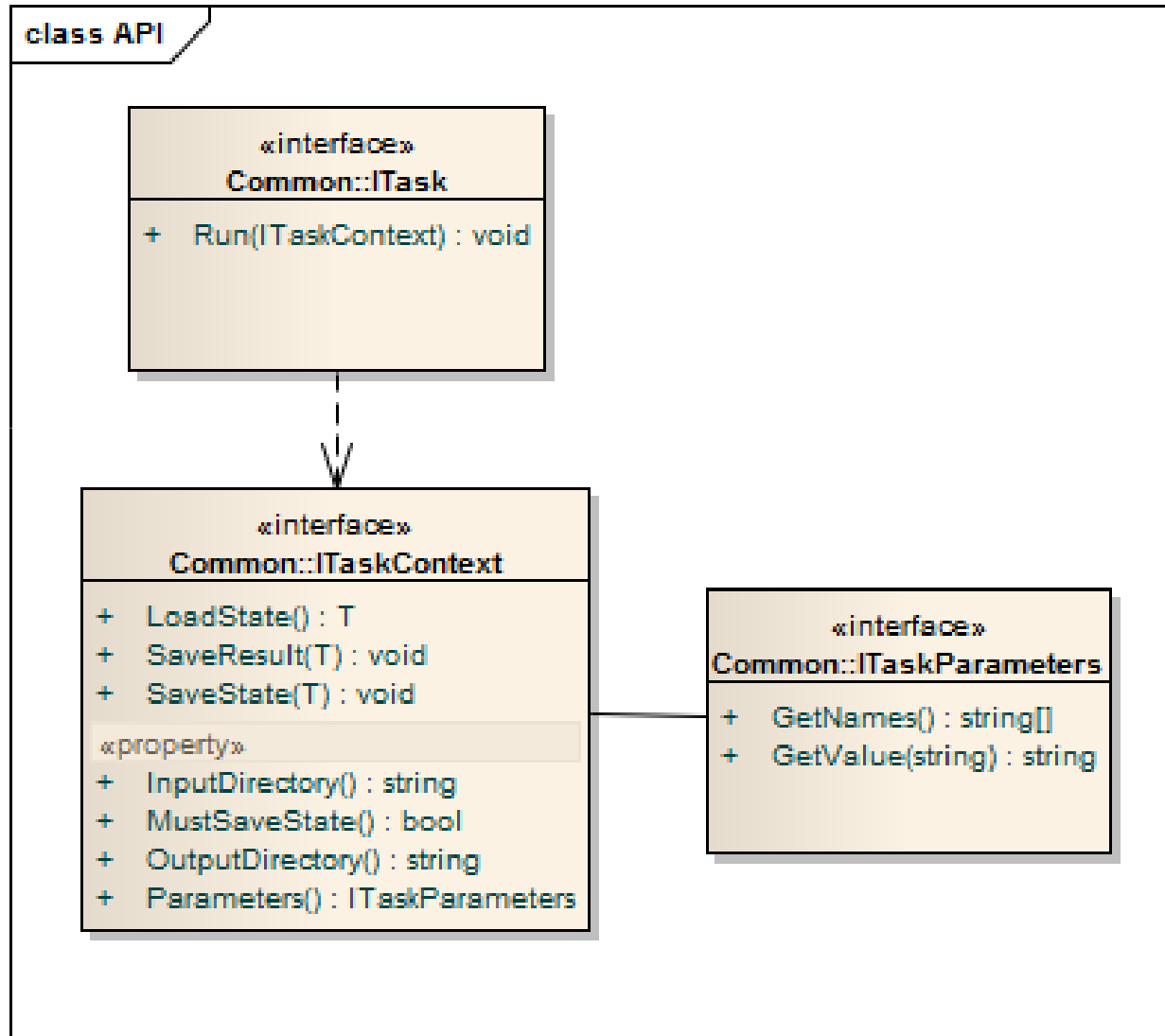


# API da grade

- DLL criada em .NET
- Deve ter uma classe que implementa a interface ITask
- A tarefa tem acesso aos arquivos de entrada e parâmetros
- Salva os dados em um diretório de saída

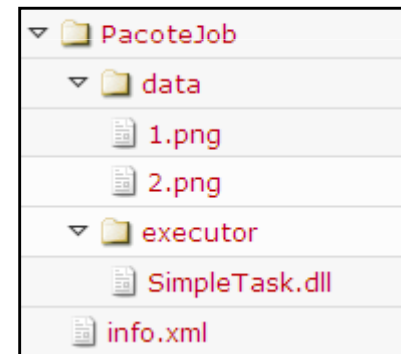


# API da grade



# Formato do Job

- Job é um conjunto de tarefas submetidas
- As tarefas são iguais, variando apenas os parâmetros
- Pacote ZIP contendo:
  - Arquivos de entrada
  - DLL do executor
  - Arquivo info.xml



# Formato do Job – Arquivo info.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<job name="AlgumJob">
  <executor>
    <library name="SimpleTask.dll"/>
    <entrypoint name="SimpleTask.SimpleTask" />
    <params>
      <param name="Outro" value="true" />
      <param name="MaisUm" value="1" />
    </params>
  </executor>
  <tasks>
    <task name="Tarefa 1">
      <data>
        <file>1.png</file>
      </data>
      <params>
        <param name="Abc" value="true" />
      </params>
    </task>
    <task name="Tarefa 2">
      <data>
```

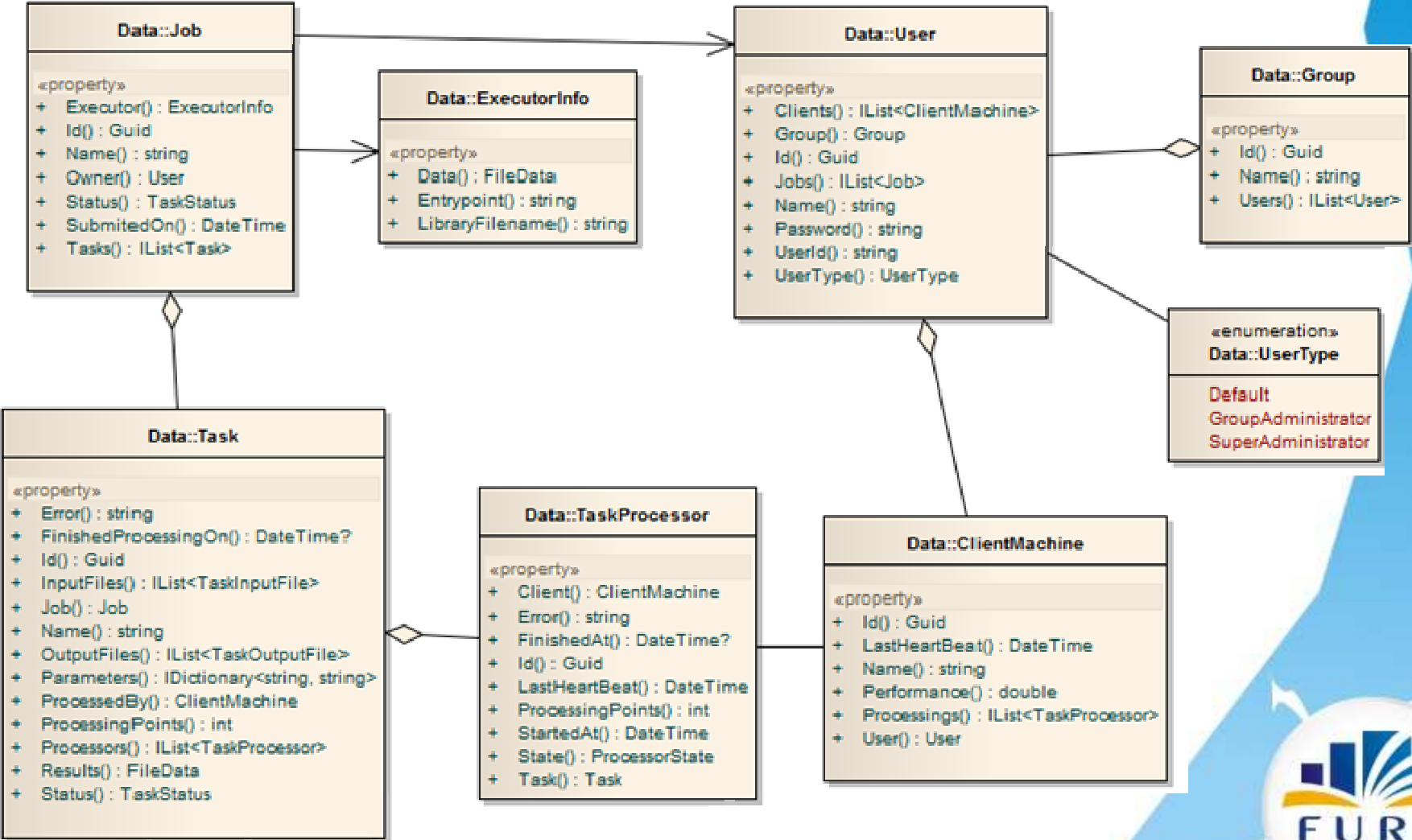


# Grupos

- Todo usuário faz parte de um grupo
- O usuário visualiza apenas dados do seu grupo
- O processamento utilizado pelo grupo é proporcional ao processamento cedido à grade



# Classes - Portal



# Implementação

- Microsoft Visual Studio
- Microsoft SQL Server
- Bibliotecas utilizadas:
  - ASP.NET MVC
  - NHibernate
  - NUnit





# Operacionalidade – Criar tarefa

```
public class AntiAliasingTask : ITask
{
    public void Run(ITaskContext context)
    {
        // Obtém o parâmetro que indica quantas vezes deve aplicar o antialiasing
        int repeatCount;
        if (!int.TryParse(context.Parameters.GetValue("RepeatCount"), out repeatCount))
        {
            repeatCount = 1;
        }

        // Busca no diretório de entrada os arquivos a serem processados
        string[] files = Directory.GetFiles(context.InputDirectory, "*.jpg");
        foreach (var file in files)
        {
            // Carrega a imagem
            Bitmap bitmap = new Bitmap(file);
            // Aplica o antialiasing
            for (int i = 0; i < repeatCount; i++)
            {
                ProcessaAntialiasing(bitmap);
            }
            // Salva o arquivo processado
            string novoArquivo = Path.Combine(context.OutputDirectory,
                Path.GetFileName(file));
            bitmap.Save(novoArquivo, ImageFormat.Jpeg);
        }
        //...
    }
}
```

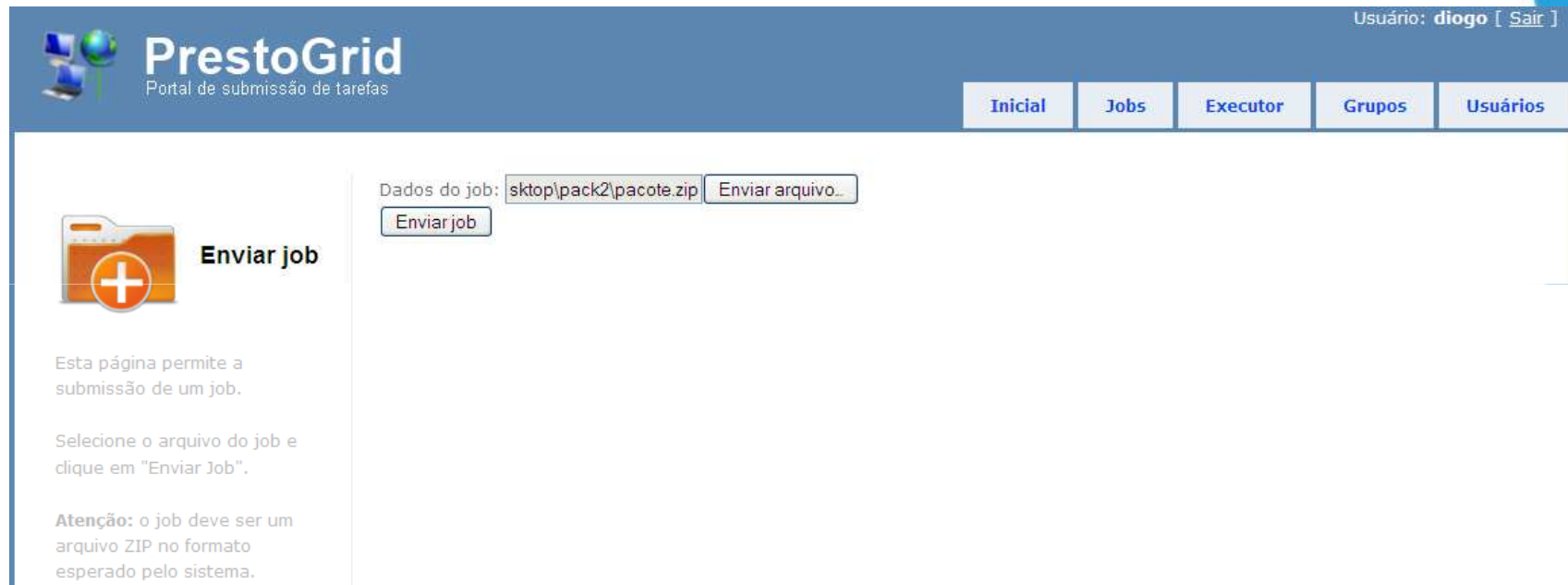


# Operacionalidade – Criar job

```
<?xml version="1.0" encoding="utf-8" ?>
<job name="Processa imagens">
  <executor>
    <library name="SimpleTask.dll"/>
    <entrypoint name="SimpleTask.AntiAliasingTask" />
    <params>
      <param name="RepeatCount" value="2" />
    </params>
  </executor>
  <tasks>
    <task name="Imagem 1">
      <data>
        <file>foto1.jpg</file>
      </data>
    </task>
    <task name="Imagem 2">
      <data>
        <file>foto2.jpg</file>
      </data>
    </task>
  </tasks>
</job>
```



# Operacionalidade – Submeter job



The screenshot shows the PrestoGrid web interface. At the top left is the logo and name "PrestoGrid" with the subtitle "Portal de submissão de tarefas". At the top right, it shows the user "diogo" with a "Sair" link. A navigation menu contains "Inicial", "Jobs", "Executor", "Grupos", and "Usuários". The main content area is titled "Enviar job" and features a folder icon with a plus sign. Below the icon, there is instructional text: "Esta página permite a submissão de um job.", "Selecione o arquivo do job e clique em 'Enviar Job'.", and "Atenção: o job deve ser um arquivo ZIP no formato esperado pelo sistema." To the right of the text, there is a form with "Dados do job:" followed by a text input containing "sktop\pack2\pacote.zip" and an "Enviar arquivo..." button. Below this is an "Enviar job" button.



# Operacionalidade - Obter resultados


Usuário: diogo [ Sair ]

**PrestoGrid**  
Portal de submissão de tarefas

[Inicial](#) [Jobs](#) [Executor](#) [Grupos](#) [Usuários](#)

## Job

**Job:** Processa imagens - Repete 1  
**Situação:** Finalizada  
**Grupo:** Grupo 1  
**Submetido por:** diogo  
**Submetido em:** 3/11/2010 14:35:44

 [Download do job](#)

	Nome	Finalizado em	Executado por	Executado pelo grupo	Entrada	Saída	Pontos de execução
✓	Imagem 4	3/11/2010 14:37:09	diogo (máquina1)	Grupo 1	<a href="#">Parâmetros Arquivos</a>	<a href="#">Resultado Arquivos</a>	11788583
✓	Imagem 2	3/11/2010 14:37:47	diogo (máquina2)	Grupo 1	<a href="#">Parâmetros Arquivos</a>	<a href="#">Resultado Arquivos</a>	10999607
✓	Imagem 1	3/11/2010 14:36:50	diogo (máquina2)	Grupo 1	<a href="#">Parâmetros Arquivos</a>	<a href="#">Resultado Arquivos</a>	8369190
✓	Imagem 3	3/11/2010 14:37:10	diogo (máquina3)	Grupo 1	<a href="#">Parâmetros Arquivos</a>	<a href="#">Resultado Arquivos</a>	11927717

[Voltar](#)

Nesta página são exibidos os detalhes do job e das tarefas pertencentes a ele.

Na listagem ao lado pode-se visualizar o status de cada tarefa e o resultado do processamento.

# Resultados e discussão

- Pode ser utilizada para qualquer finalidade (ao contrário do SETI@home)
- É uma solução pronta e não apenas o fundamento (ao contrário do BOINC)
- Utiliza o tempo ocioso dos computadores (ao contrário do Globus)
- É restrito a tarefas do tipo bag-of-tasks



# Conclusão

- Todos os objetivos foram alcançados
- As ferramentas utilizadas se mostraram adequadas
- Cerca de 90% de processamento de um computador não é utilizado
- A grade é de fácil utilização
- Uma grade computacional é bastante complexa e permite estudos de cada uma de suas partes separadamente



# Extensões

- Possibilitar o uso de linguagens dinâmicas
- Adaptar o projeto para poder ser executado em Linux e outros sistemas operacionais
- Exibir um ranking de usuários e grupos que mais contribuem para a grade



# Demonstração

