**UNIVERSIDADE FEDERAL DE SANTA MARIA**
**CENTRO DE TECNOLOGIA**
**CURSO DE CIÊNCIA DA COMPUTAÇÃO**

# UM SISTEMA WEB PARA EXECUÇÃO REMOTA DE APLICAÇÕES DE ALTO DESEMPENHO

**TRABALHO DE GRADUAÇÃO**

**Otávio Migliavacca Madalosso**

**Santa Maria, RS, Brasil**

**2015**

# UM SISTEMA WEB PARA EXECUÇÃO REMOTA DE APLICAÇÕES DE ALTO DESEMPENHO

**Otávio Migliavacca Madalosso**

Trabalho de Graduação apresentado ao Curso de Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de

**Bacharel em Ciência da Computação**

**Orientadora: Profª. Drª. Andrea Schwertner Charão**

**Santa Maria, RS, Brasil**

**2015**

**Universidade Federal de Santa Maria**
**Centro de Tecnologia**
**Curso de Ciência da Computação**

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

**UM SISTEMA WEB PARA EXECUÇÃO REMOTA DE APLICAÇÕES DE ALTO DESEMPENHO**

elaborado por
**Otávio Migliavacca Madalosso**

como requisito parcial para obtenção do grau de
**Bacharel em Ciência da Computação**

**COMISSÃO EXAMINADORA:**

**Andrea Schwertner Charão, Dr[a].**
(Presidente/Orientadora)

**silva non se sabe, Prof[a]. Dr[a].** (UFSM)

**incognito anon, Prof. Dr.** (UFSM)

Santa Maria, 08 de Outubro de 2015.

# RESUMO

Trabalho de Graduação
Curso de Ciência da Computação
Universidade Federal de Santa Maria

**UM SISTEMA WEB PARA EXECUÇÃO REMOTA DE APLICAÇÕES DE ALTO DESEMPENHO**
AUTOR: OTÁVIO MIGLIAVACCA MADALOSSO
ORIENTADORA: ANDREA SCHWERTNER CHARÃO
Local da Defesa e Data: Santa Maria, 08 de Outubro de 2015.

Algumas áreas de pesquisa utilizam constantemente algoritmos que demandam alto desempenho dos seus ambientes de execução. Ocasionalmente, surgem algoritmos novos, com diferentes propriedades, que se propõem a resolver um problema de forma mais eficiente e/ou completa. Infelizmente, é comum que esses algoritmos fiquem restritos a ambientes institucionais, limitando muito a sua visibilidade para a comunidade de pesquisa. Este trabalho tem como objetivo criar um portal que permita ao usuário solicitar a execução remota de um algoritmo de acordo com as configurações que o sistema oferecer.

**Palavras-chave:** Computação de alto desempenho. Programação Web. !!!!!!!!!!!!!!!1. !!!!!!!!!!!!!!!3. !!!!!!!!!!!!!!!2.

# SUMÁRIO

# 1 INTRODUÇÃO

Algoritmos com grande custo computacional são facilmente encontrados em áreas como meteorologia, biologia e astronomia. Esses algoritmos possuem a característica de utilizar um nível elevado de processamento para concluir sua execução, e consequentemente, seus tempos de execução podem variar dependendo da máquina aonde estão sendo executados.

É comum pesquisadores destas e de outras áreas desenvolverem novas implementações de algoritmos utilizados por seus colegas. Implementações essas que podem trazer muitos benefícios para outros pesquisadores que necessitam deste tipo de solução. Infelizmente, é comum essas implementações ficarem restritas a ambientes privados, não por questões de licença, mas simplesmente pela ausência de um método prático para disponibilizar a nova ferramenta ao público.

Com isso, surge a ideia de desenvolver um portal web que permita ao usuário o cadastro de um experimento, no qual ele poderá ditar os dados de entrada desse experimento, e qual algoritmo (disponível no sistema) ele deseja utilizar para processar os dados. Depois de requisitar o experimento, o sistema deve providenciar sua execução e quando finalizar, retornar o resultado do experimento ao usuário que o requisitou.

Para este portal, será utilizado um algoritmo desenvolvido para a área de astronomia, uma versão do algoritmo Friends-of-Friends de complexidade n*log(n) paralelizada através do framework OpenMP.

## 1.1 Objetivos

### 1.1.1 Objetivo Geral

O objetivo deste trabalho é criar um portal web que possibilite aos usuários cadastrados no sistema executar algoritmos utilizando diferentes dados e disponibilizar o resultado da execução após sua conclusão.

## 1.2 Justificativa

O projeto é capaz de gerar benefícios significativos para a comunidade de pesquisa de diversas áreas, criando um modelo de ambiente que facilite a divulgação e teste de resultados

de algoritmos alternativos para resolução de problemas comuns.

Além de servir como modelo, o projeto disponibilizará um algoritmo que se enquadra na categoria alvo do projeto: a versão de complexidade n*log(n) e paralela do friends-of-friends.

# 2 FUNDAMENTOS E REVISÃO DE LITERATURA

## 2.1 Framework Django

Django(DJANGO PROJECT, ????) é um framework para criação de aplicações web que encoraja o desenvolvimento ágil, em alto nível e com design pragmático. Foi criado inicialmente para manter o portal de notícias online do Lawrence Journal World pelos programadores Adrian Holovaty, Simon Willison e Jacob Kaplan-Moss.

Por se tratar de um portal inicialmente desenvolvido para administrar notícias, o framework lida muito bem com gerenciamento de conteúdo e agilidade quando é necessário fazer alterações no sistema.

## 2.2 Friends of Friends

Simulações de N-corpos têm sido utilizadas para promover vários avanços na compreensão de questões relevantes em astrofísica, como por exemplo o processo de formação e evolução de estruturas do Universo. Este tipo de simulação tem um papel fundamental(BERTSCHINGER, 1998; G. EFSTATHIOU M. DAVIS, 1985) no estudo da evolução cósmica em tópicos como a distribuição de matéria escura em grande escala, a formação de halos de matéria escura, e a formação e evolução de galáxias e aglomerados.

A manipulação e análise da grande quantidade de dados produzidos em tais simulações também é algo desafiador. Neste contexto, é essencial o desenvolvimento de técnicas computacionais eficientes para extrair informação significativa a partir dessas fontes de dados, em um período apropriado de tempo.

Etapas importantes neste tipo de análise são a identificação de halos de matéria escura e o estudo do espectro da energia potencial gravitacional de tais objetos. Uma abordagem para este tipo de análise consiste em usar o algoritmo de percolação Friends-of-Friends (FoF) (HUCHRA J. P., 1982) . A ideia básica deste algoritmo é a seguinte: considere uma esfera de raio R ao redor de cada partícula do conjunto total; se dentro desta esfera existirem outras partículas, elas serão consideradas pertencentes ao mesmo grupo e serão chamadas de amigas. Em seguida, toma-se uma espfera ao redor de cada amiga e continua-se o procedimento usando a regra "qualquer amigo de meu amigo é meu amigo". O procedimento para quando nenhuma amiga nova puder ser adicionada ao grupo.

Na 15ª edição da Escola Regional de Alto Desempenho do Rio Grande do Sul (XV ERAD-RS)(referenciar), foram publicados resultados de execuções de uma nova implementação do Friends-of-Friends(OTáVIO MIGLIAVACCA MADALOSSO, 2015) cuja complexidade computacional era reduzida em relação as versões anteriores, resultando em redução do tempo de processamento.

# 3 DESENVOLVIMENTO

A metodologia do projeto concentrou-se em dar prioridade aos requisitos que apresentavam maior risco, priorizando a funcionalidade de executar remotamente tarefas. Por causa disso foi feita uma "Sprint"para agilizar o desenvolvimento de funcionalidades mais básicas do projeto, que dariam base ao sistema no qual iria ser desenvolvida a funcionalidade de requisição de execuções remotas pelos usuários, parte crítica do projeto.

Para isso, o projeto iniciou através do estudo e familiarização do framework Django, através de uma série de video aulas disponíveis no youtube e da documentação do framework, foram desenvolvidas as páginas de 'contato' e 'informações' da aplicação.

Através desse estudo de video aulas foram selecionados também algumas aplicações extensíveis que, conforme observou-se, poderiam trazer ganhos ao projeto em questão de agilidade de desenvolvimento. As aplicações estudadas foram o django-registration-redux, para registro de usuário, e o django-crispy-forms, que efetua alterações de formulários conforme o resultado da validação do mesmo.

## 3.1 Formulário de Contato

A página de contato do sistema foi desenvolvida utilizando um formulário que exige informações necessárias para identificação e endereço de resposta de quem efetuou o contato. Para isso foi utilizada a aplicação django-crispy-forms, ela trata de casos como o de quando um usuário tenta efetuar um contato e não preenche um dos campos necessários, dependendo do resultado da validação, efetuada no backend do sistema, é retornado um novo formulário semelhante ao original, porém, com mensagens deixando explícito ao usuário quais foram os erros cometidos na tentativa de efetuar contato.

## 3.2 Registro de Usuário

Depois disso houve o estudo da logística de registro de usuários, como a aplicação deve ser aberta a qualquer pessoa, é imprescindível que haja uma forma autônoma de registro de novos usuários. No primeiro momento tentou-se implementar a funcionalidade utilizando formulários e verificações desenvolvidas no proprio projeto, mas durante o desenvolvimento foi encontrada uma solução mais rápida e mais adequada, o uso do app Django-Registration-Redux.

O django-registration-redux é uma aplicação extensível que provê as funcionalidades de registro de usuários para sites que utilizam Django. Essa aplicação já dispõe de templates e formulários para fazer a sua função. Por padrão ela mantém apenas os dados de email, nome e senha do usuário que solicitou o registro, o que satisfaz os requisitos mínimos para executar sua funcionalidade, porém para o sistema em desenvolvimento, pareceu interessante ter a possibilidade de obter mais dados do usuário em seu cadastro, tais como instituição a qual o usuário pertence (Universidade, empresa, etc.), grau de escolaridade, e outras informações que podem ser úteis posteriormente ao administrador do sistema. Além disso, o django-registration-redux não tem por padrão, nenhuma forma de filtro de domínios de endereços de email, o que também pode ser útil ao sistema, caso o administrador queira limitar o acesso do portal aos usuários que detenham um email de um domínio específico.

Para contornar essas questões, foi criado um novo modelo de dados que extende a aplicação registration-redux, tornando possível extender as funções de registro para esse modelo novo, de modo que no processo de registro novos dados podem ser requisitados, verificados e validados pelo sistema antes de proceder com a criação do usuário.

### 3.3 Execução de tarefas

A execução remota de tarefas foi um requisito que exigiu outro estudo em busca de ferramentas e tecnicas para sua realização. É necessário esclarecer que não é viável que o próprio processo que mantém a aplicação no ar e trata de todos os requests realizados por usuários também lide com as execuções dos experimentos solicitados pelos usuários, pois isso causaria uma lentidão muito grande no sistema.

Para contronar esse problema foram verificadas duas técnicas: a criação de um novo processo que faria a execução do experimento, ou então o uso de alguma aplicação que gerencia-se filas de tarefas e distribuição das mesmas para demais processos e/ou máquinas.

Compreendida a dimensão e complexidade de criar um sistema para execução de tarefas a partir do zero, optou-se por buscar por aplicações compatíveis com as tecnologias utilizadas do projeto e que pudessem satisfazer a necessidade identificada.

—PORQUE CELERY—

BROKER lidar mensagens

Worker

### 3.4 Sistema de Arquivos

Como as tarefas realizadas pelo portal exigem dados para serem processados pelo algoritmo, e esse algoritmo gera novos dados, foi necessário criar um sistema de arquivos para manter uma ordem na qual seja possível recuperar esses arquivos após o seu processamento. O sistema foi implementado conforme a imagem ilustra. Nele, é escolhido um diretório como sendo a raíz de todos os arquivos que o portal irá manter referente a dados de entrada e saída dos algoritmos, e esses dados serão mantidos em diretórios nomeados de acordo com o id do experimento ao qual pertencem, e esse diretório, por sua vez, será mantido em um outro diretório nomeado de acordo com o id do usuário que requisitou o experimento.

Após a implementação desse sistema de arquivose e combinando as funcionalidades implementadas com o que já havia sido implementado referente a execução de tarefas, foi possível desenvolver uma página no portal na que o usuário pudesse obter os arquivos de dados referentes a cada experimento solicitado por ele e verificar o estado de cada experimento. Esses estados são definidos de acordo com o andamento do processo de execução, sendo que até o momento, são possíveis 3 estados:

- "Aguardando": Experimento aguarda sua execução por um worker.
- "Executando": Experimento está sendo executado, mas ainda não concluíu.
- "Finalizado": Experimento já foi executado e já possui os dados disponíveis para o usuário.
- Ambiente de Desenvolvimento
- Organização do Projeto
- Modelos de dados
- Funcionalidades
- Design
- Administração

## 3.5 Ambiente de Desenvolvimento

### 3.5.1 Sistema Operacional

Para o desenvolvimento do projeto está sendo utilizado o sistema operacional Ubuntu 14.04(REF?), que contém um interpretador da linguagem Python(REF)?, sendo necessário apenas fazer a instalação do framework Django 1.8(REF?) para ter um ambiente pronto para começar o desenvolvimento do projeto. Além disso, também foi instalado o sistema de gerenciamento de pacotes pip(REF), que foi utilizado para instalação de outras ferramentas (descritas abaixo) que auxiliaram no desenvolvimento.

### 3.5.2 Ferramentas de Programação

Para auxiliar na tarefa de programação do sistema, foi utilizado o editor de texto Sublime Text 2 (REF!), um blabla.... utilizando também o plugin Anaconda... Django algo...

### 3.5.3 Controle de Versão - Repositório

Para realizar o controle de versões foi utilizado o *Git*[1], também utilizado pelo projeto original.

## 3.6 Organização do Projeto

O framework Django tem uma proposta de criar aplicações fáceis de serem reutilizadas por outros projetos, e por conta disso tem uma abordagem bastante metódica e pouco flexível em relação a distribuição dos arquivos que pertencem ao projeto. Consequentemente, a organização utilizada é a mesma proposta pela documentação do framework, que pode ser observada na figura abaix (incluir figura)

### 3.6.1 *manage.py*

explicar a funcionalidade do manage e sua importancia para todo o projeto

[1] O repositório com o código pode ser encontrado em https://github.com/Madalosso/TG

### 3.6.2 *Static Files*

Os arquivos estáticos que o projeto mantém são todos organizados dentro do diretório "static in pro", lá são mantidos arquivos que precisam estar disponíveis por qualquer documento html, em grande maioria são arquivos .css, .js e imagens comuns às páginas do projeto.

### 3.6.3 *Media url*

Media URL é a variável responsável por indicar ao sistema qual diretório será a raiz dos arquivos que serão manipulados pelo sistema. Dentro desse diretório raiz existe uma pasta "Users"e dentro dela, uma pasta com o id de cada usuario cadastrado no banco de dados. Para cada experimento criado por um usuário, em seu respectivo diretório será criado um novo diretório cujo nome será o identificador do experimento, e dentro desse diretório serão mantidos os arquivos de entrada e saída do experimento.

### 3.6.4 *views.py*

Esse arquivo contém as funções que definem telas a serem exibidas para os utilizadores do sistema. Como padrão toda função nesse arquivo recebe com um *request* e retorna alguma resposta html, essa resposta na maioria dos casos utiliza arquivos presentes no diretório "Templates".

## 3.7 Modelos de dados

adicionar imagem de algum software mostrando as relações entre os tipos de dados? explicar o que cada classe / atributo tem como objetivo

## 3.8 Funcionalidades

As funcionalidades do sistema podem ser divididas até agora em 3 grupos distintos.

### 3.8.1 Usuário Anonimo

Um usuário anônimo possui a permissão de acessar áreas de informação a respeito do sistema, de contato com o administrador do sistema, e da área de registro, onde pode solicitar o

registro e, seguindo as orientações apresentadas, fazer login como um usuário registrado.

### 3.8.2 Usuário registrado

O usuário registrado tem permissão de criar experimentos no portal, para isso ele faz o *upload* de um arquivo que será utilizado como entrada no algoritmo selecionado. Além disso o usuário também pode monitorar o estado dos experimentos que ele requisitou e fazer o *download* dos arquivos de cada experimento, tanto o arquivo de entrada, como a saída(se houver) do algoritmo.

### 3.8.3 Administrador

O administrador tem as mesmas capacidades do que um usuário registrado e detem previlégios de acesso ao painel de administração do django. Isso permite a ele cadastro de novos algoritmos no sistema, de novos experimentos padrão e a editar qualquer informação que o sistema detenha no banco de dados.

# 4 PRÓXIMAS ETAPAS

Como continuação do trabalho realizado até o momento, são planejadas as seguintes atividades:

1. fazer a execução das tarefas em uma máquina diferente da que mantém o portal.
2. Avaliar novas idéias de funcionalidades e implementar as que forem validadas.
3. tornar site responsivo via bootstrap.

# REFERÊNCIAS

BERTSCHINGER, E. Simulations of structure formation in the universe. **Annu. Rev. Astron. Astrophys 36**, [S.l.], 1998.

DJANGO Project. Accessed: 2015-08-14, https://www.djangoproject.com/.

G. EFSTATHIOU M. DAVIS, S. D. M. W. C. S. F. Numerical techniques for large cosmological N-body simulations. **Astrophysical Journal Supplement Series (ISSN 0067-0049), vol. 57**, [S.l.], 1985.

HUCHRA J. P., G. M. J. Groups of galaxies. I - Nearby groups. **Astrophysical Journal, Part 1, vol. 257, June 15, 1982, p. 423-437. Research supported by Cambridge University**, [S.l.], 1982.

MEDIGRID. Accessed: 2015-08-14, http://www.medigrid.de/index_en.html.

NEW Zeland eScience Infrastructure. Accessed: 2015-08-14, https://www.nesi.org.nz/.

OTáVIO MIGLIAVACCA MADALOSSO, A. S. C. Implementação do algoritmo Friends of Friends de complexidade n*logn(n) para classificação de objetos astronômicos. **ERAD-RS XV**, [S.l.], 2015.