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, Drª.
(Presidente/Orientadora)

Lizandra Manzoni Fontoura, Profª. Drª. (UFSM)

Eduardo Piveta, 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. Framework Django. Execução Remota. Respon.

SUMÁRIO

1 INTRODUÇÃO

Algoritmos com grande custo computacional são facilmente encontrados em áreas como meteorologia, biologia e astronomia. Esses algoritmos possuem a característica de consumir um nível elevado de processamento, e consequentemente, os tempos necessários para suas conclusões tendem a ser longos e variam dependendo do ambiente aonde são executados.

É comum pesquisadores destas 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 disponibiliza-la ao público.

Baseado nessa situação, surge a ideia de desenvolver um portal web que permita a execução de algoritmos remotamente, de acordo com as configurações feitas pelo administrador. Desta forma, o usuário seria capaz de utilizar dados próprios para que sejam processados pelos algoritmos, e consiga obter os resultados quando a tarefa for concluída.

Um algoritmo que se enquadra no propósito do portal e que será utilizado durante o desenvolvimento do mesmo, é a uma versão do algoritmo Friends-of-Friends de complexidade n*log(n) paralelizado através do *framework* OpenMP.

1.1 Objetivos e Justificativa

1.1.1 Objetivo Geral

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

1.1.2 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

As ferramentas e recursos utilizadas nesse trabalho são: framework Django, Celery, algoritmo Friends-of-Friends e trabalhos relacionados. Eles serão apresentados mais profundamente a seguir.

2.1 Framework Django

Django(DJANGO PROJECT, ????) é um *framework* escrito na linguagem *Python* para criação de aplicações web que encoraja o desenvolvimento ágil, em alto nível e com design pragmático. Foi criado em 2003 para gerenciar um site jornalístico da cidade de Lawrence, no Kansas e tornou-se um projeto de código aberto publicado sob a licença BSD em Julho de 2005.

O principal objetivo do framework é facilitar a criação de *websites* dirigidos a banco de dados e se relaciona muito com a política de DRY (Dont Repeat Yourself) e *pluggability*. Dentre as características desejadas para realização do projeto, o Django apresentou-se suficiente por apresentar suporte a tecnologias como:

- Suporte Ajax (asynchronous JavaScript and XML)
- Template Model–view–controller
- Object-relational mapping
- Suporte a frameworks de validação
- Suporte a frameworks de migração de banco de dados

2.1.1 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, também contém um sistema de ativação de contas no qual o usuário que solicitou registro recebe um e-mail no endereço usado no cadastro que contém um link para ativar sua conta, forçando uma verificação de que o e-mail utilizado existe e pertence mesmo ao usuário.

Por padrão ela mantém apenas os dados de e-mail, 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 e-mail, o que também pode ser útil ao sistema, caso o administrador queira limitar o acesso do portal aos usuários que detenham um e-mail de um domínio específico.

2.1.2 Django Crispy Forms

Django-crispy-forms é uma aplicação Django que trata do processo de validação e renderização de formulários...

2.2 Celery

Celery Task Queue é um gerenciador de tarefas assíncronas baseado em trocas de mensagens. Ele permite a execução e o agendamento de tarefas em um ou mais *"Workers"*. *"Workers"* são os outros processos que se comunicam com o gerenciador, que detém a função de realizar as tarefas solicitadas. Celery é escrito em Python e admite vários programas para transmissão de mensagens (*Brokers*) entre o gerenciador e seus *"Workers"*, sendo o mais recomendado deles o RabbitMQ.

2.2.1 RabbitMQ

2.3 Friends of Friends

O Friends-of-Friends (FoF) é um algoritmo utilizado para manipular e analisar grandes quantidades de dados produzidos por simulações da área da astronomia, mais especificamente 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. Essas simulações tem um papel fundamental(BERTSCHINGER, 1998; G. EFSTATHIOU M. DAVIS, 1985) no estudo desses assuntos.

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(OTAVIO MIGLIAVACCA MADALOSSO, 2015) originados de um projeto de pesquisa cujo objetivo era reduzir a complexidade do algoritmo e paraleliza-lo para que obtivesse uma diminuição de seu tempo de execução.

2.4 Trabalho Relacionados

A seguir será apresentada uma lista com alguns trabalhos que se relacionam ao projeto em questões como a disponibilização de recursos para pesquisadores remotamente.

- O New Zeland eScience Infrastructure - NeSI(NEW ZELAND ESCIENCE INFRASTRUCTURE, ????)é um portal web que provê plataformas de grande capacidade computacional para auxiliar pesquisadores na Nova Zelândia. Atualmente eles possuem 5 ambientes disponíveis em diferentes instalações. Cada um desses possui hardware e software capazes de resolver problemas relacionados a áreas de pesquisa, especialmente ligados à química e bioinformática.

- o MediGRID (MEDIGRID, ????) também é um portal web idealizado para auxiliar pesquisadores, porém esse é focado em pesquisa na área da biomedicina, ele oferece aplicações e infraestrutura para os pesquisadores cadastrados no portal realizarem experimentos conforme a sua vontade e a disponibilidade do sistema. O objetivo principal do MediGRID quando foi desenvolvido era ser uma plataforma de integração middleware ligando serviços eScience com as pesquisas de biomedicina. Porém hoje ele executa tarefas em 3 áreas majoritárias: Biomedicina, processamento de imagens e pesquisa clinica.

3 DESENVOLVIMENTO

A metodologia do projeto concentrou-se em dar prioridade aos requisitos que apresentavam maior risco, priorizando a funcionalidade de executar tarefas remotamente. Por causa disso foi feita uma *Sprint* para agilizar o desenvolvimento de funcionalidades mais básicas do projeto, focando especialmente na questão de execuções remotas solicitadas pelos usuários cadastrados no sistema.

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 Crispy que tratou da validação de formulário e seguiu-se o método padrão disponível pelo Django para envio de email.

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, também contém um sistema de ativação de contas no qual

o usuário que solicitou registro recebe um e-mail no endereço usado no cadastro que contém um link para ativar sua conta, forçando uma verificação de que o e-mail utilizado existe e pertence mesmo ao usuário.

Por padrão ela mantém apenas os dados de e-mail, 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 e-mail, o que também pode ser útil ao sistema, caso o administrador queira limitar o acesso do portal aos usuários que detenham um e-mail 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 Modelos de dados

Quando a etapa de registro de usuários foi completa, foram criados modelos de dados para representar os algoritmos disponíveis pelo sistema, execuções que seriam requisitadas pelos usuários, e modelos de execuções.

Essas três classes, combinados com a classe de usuário, que mantém os dados do registro, compõem o modelo de dados(Figura 3.1) que integram todas as funcionalidades do sistema até o presente estado de desenvolvimento.

3.3.1 Algorithm

Essa classe é responsável por manter os dados referentes ao(s) algoritmo(s) que o sistema disponibiliza, é composto por 3 atributos que representem o nome do algoritmo, a descrição de seu propósito, e o comando (shell) que deverá ser utilizado para executá-lo.

3.3.2 ExecModel

Essa classe foi criada para permitir ao usuário testar os serviços do sistema sem precisar de um conjunto de dados. Assim, essa classe guarda dados referentes a execução pré-definida de um algoritmo, utilizando os dados de entrada já configurados pelo administrador do sistema. Além disso, ele mantém um campo descrição criado com o intúito de explicar ao usuário como funciona o conjunto de dados utilizados como entrada e o que se espera do resultado após o processamento.

3.3.3 Execution

A classe Execution mantém todas as informações referentes a um pedido de execução. Ela inclui uma chave estrangeira que referencia o usuário que requisitou-a, a data na qual a requisição foi feita, o status da execução, outra chave estrageira que referencia qual algoritmo será utilizado, e dois campos para os endereços nos quais devem ser mantidos os dados de entrada e saída à serem usados nesta execução.

3.3.4 PortalUser

Representa o usuário no protal, mantém os dados de cadastro e mais alguns referêntes a datas e preferências:

- Username (Utilizado para o login)
- Password
- E-mail
- Nickname (Utilizado para mensagens de contato e alertas pelo sistema)
- DateRegister (Data de quando foi feito o registro do usuário)
- LastAcess (Data do último acesso do usuário no sistema)
- ResultsPerPage (Preferência de quantos resultados por página o usuário deseja quando for listar os experimentos que requisitou)

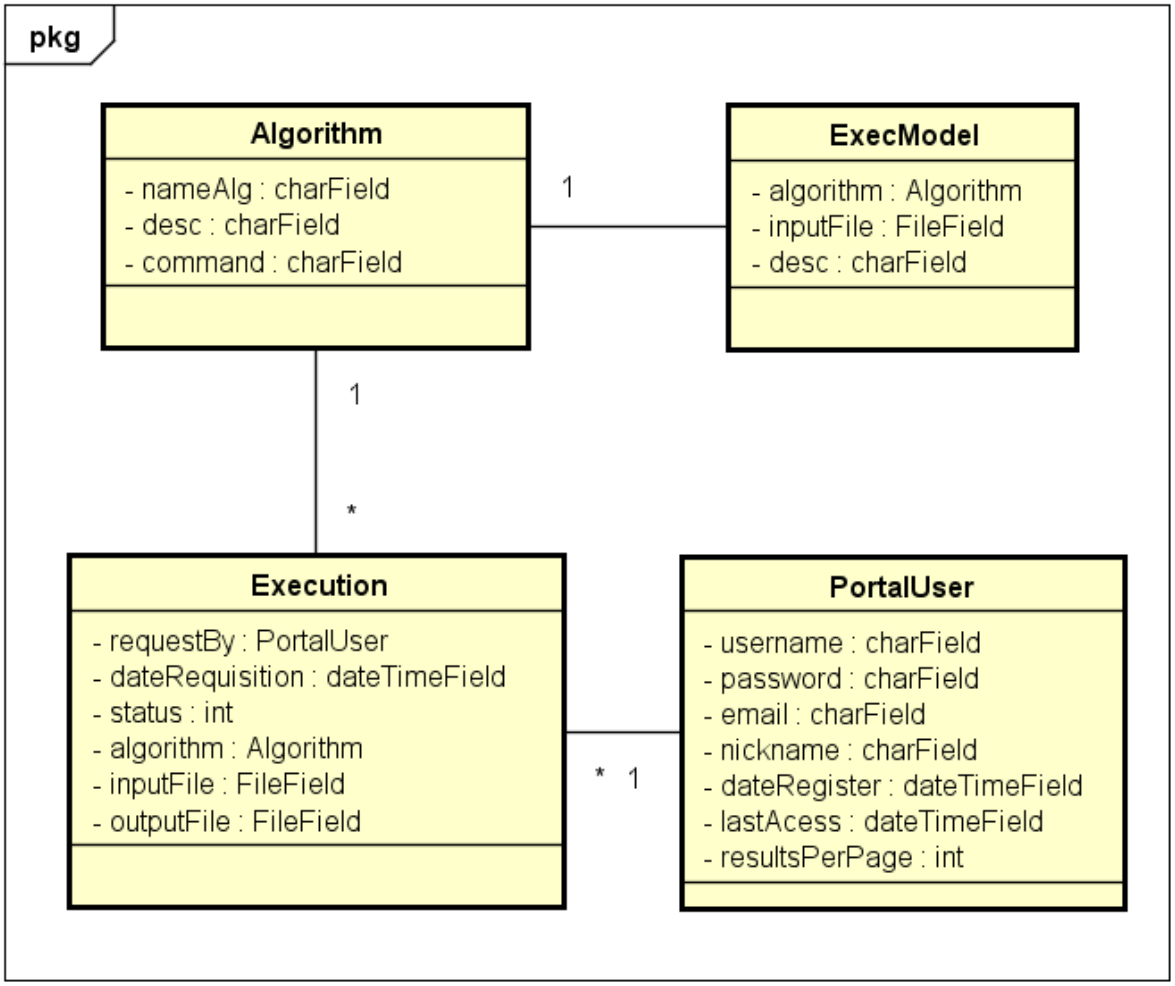

Figura 3.1 – Diagrama de classes do projeto.

3.4 Execução Remota 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.

A aplicação escolhida foi o Celery, uma aplicação que gera filas de execução de tarefas por meio de troca de mensagens. A grande vantagem dessa escolha é a integração nativa que o Celery tem com Django, e a vasta documentação existente para configuração de suas funcionalidades.

Essa aplicação permite ao sistema django criar tarefas que serão executadas pelos chamados workers. Os workers são processos independentes que devem ser iniciados nas máquinas que irão executar o processamento dos algoritmos e que trocam mensagens com o sistema que solicitou a execução.

No momento, a execução das terefas criadas pelo sistema estão todas sendo realizadas por um worker executado na mesma máquina que mantém o portal, posteriormente espera-se fazer com que isso seja alterado para que a máquina que realiza o processamento seja exclusivamente dedicada à essa tarefa.

3.5 Funcionalidades

As funcionalidades do sistema podem ser divididas até agora em 3 grupos distintos. (Figura 3.2).

3.5.1 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.

3.5.2 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.

3.5.3 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.

3.6 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 refe-

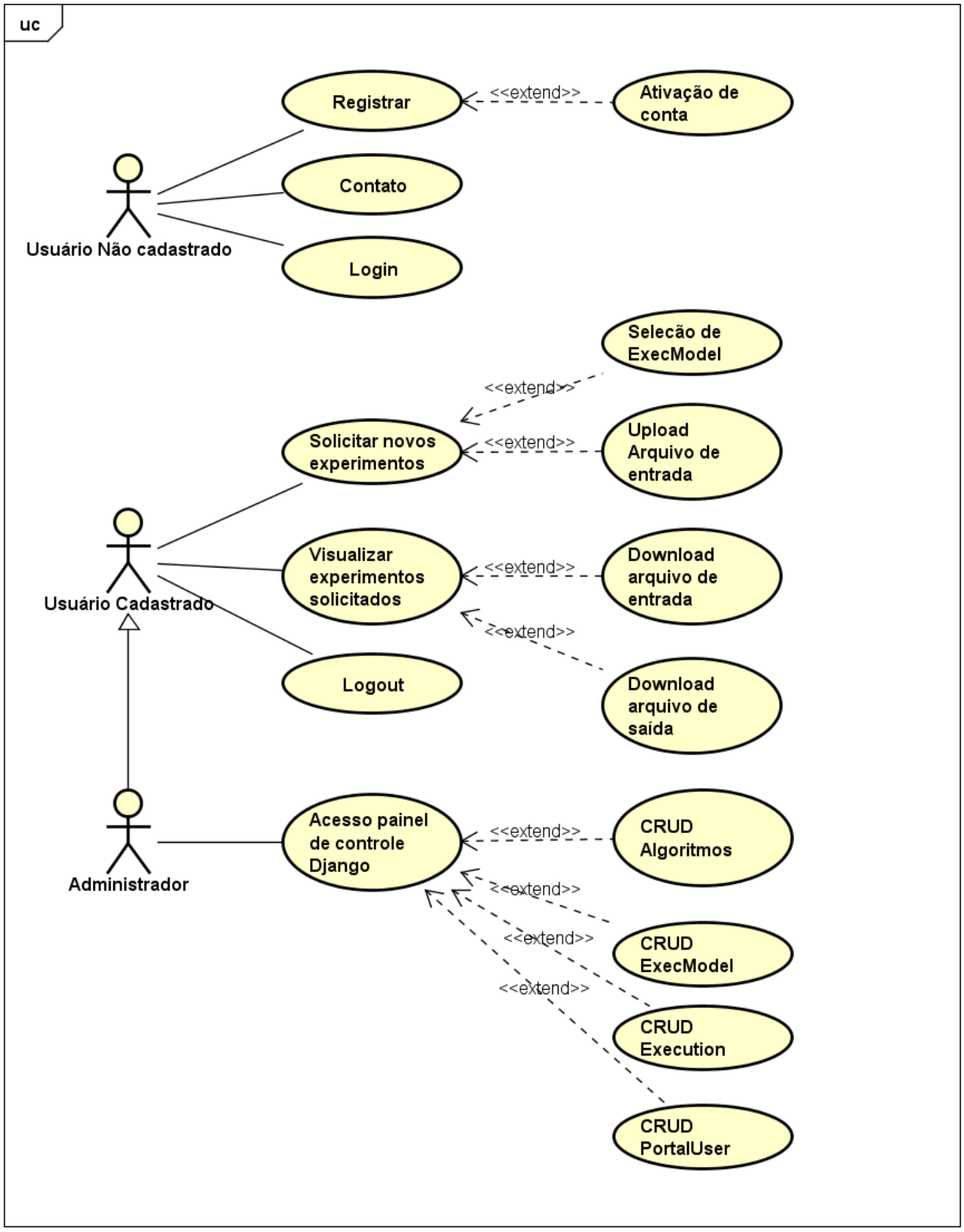

Figura 3.2 – Diagrama de casos de uso do projeto.

rentes 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.

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 utilizando o *framework* 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.

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/.

OTAVIO 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.