UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

ESCOLA DE INFORMÁTICA APLICADA

AMAO - DESENVOLVIMENTO DE UM AMBIENTE ONLINE DE AUXÍLIO À CORREÇÃO E RESOLUÇÃO DE AVALIAÇÕES DE PROGRAMAÇÃO

Felipe Arruda Pontes

Zanoni de Castro de Miranda

Orientadora

Drª. Geiza Maria Hamazaki da Silva

RIO DE JANEIRO, RJ – BRASIL

AGOSTO DE 2013

AMAO - DESENVOLVIMENTO DE UM AMBIENTE ONLINE DE AUXÍLIO À CORREÇÃO E RESOLUÇÃO DE AVALIAÇÕES DE PROGRAMAÇÃO

Felipe Arruda Pontes
Zanoni de Castro de Miranda

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

Aprovada por:

_____

Drª. Geiza Maria Hamazaki da Silva (UNIRIO)

_____

Drª. Leila Cristina Vasconcelos de Andrade (UNIRIO)

_____

Msc. Bruno Lopes Vieira (PUC - Rio)

RIO DE JANEIRO, RJ – BRASIL.
AGOSTO DE 2013

Agradecimentos

Gostaríamos aos integrantes da banca que se dispuseram a participar e prestigiar nosso trabalho. A nossa orientadora Geiza Maria Hamazaki da Silva pelo apoio e auxilio durante o trabalho, a Leila Cristina Vasconcelos de Andrade que sempre nos incentivou na programação ao longo do curso e ao Bruno Lopes Vieira pela atenção que deu ao nosso projeto. Agradecemos também a todos os outros que apoiaram nosso trabalho e nos deram forças para fazer o melhor de nós.

RESUMO

O presente trabalho discute a importância de uma plataforma de auxílio à criação, correção e resolução de avaliações de programação. Seu objetivo é propor um sistema, denominado AMAO, sendo este uma plataforma online de autocorreção para auxiliar no aprendizado de programação com as características de ser, escalável, modularizado e fazendo uso exclusivamente de softwares livres.

Optou-se, ao longo de todo processo de construção da ferramenta, pelo emprego de boas práticas de desenvolvimento e de tecnologias que atualmente são utilizadas por empresas de referência no mercado de Tecnologia da Informação.

Num primeiro estágio foram pesquisadas as principais características de ferramentas existentes de autocorreção a fim de aproveitar tais conceitos na formulação do AMAO. Os resultados obtidos orientaram, então, o desenvolvimento do software proposto.

Palavras-chave: Informática na Educação, Avaliações Online, Corretor Automático, Software Livre

ABSTRACT

The present work will discuss the importance of a platform to assist at the process of creating, grading and resolution of programming exercises. Its goal is to propose an scalable, modularized, online auto-grading system (called AMAO) to help the programming learning using exclusively free softwares.

During the development process we adopted the use of good development practices and technologies that are used by companies that are reference in the Information Technology market.

In the first stage were surveyed the main characteristics of existing auto-grading tools, in order to reuse their concepts in AMAO's formulation. Afterwards, the obtained results guided in the proposed software development.

Keywords: Informatics in Education, Online Evaluations, Automatic Grading, Free Software

SUMARIO

Lista de Figuras

1 INTRODUÇÃO

O estudo da programação é muito importante para estudantes de computação, pois este forma a base de conhecimento para diversas disciplinas da área, além de prepará-los melhor para disciplinas mais avançadas do curso (MOREIRA; FAVERO, 2009). Entretanto, o aprendizado de programação é considerado árduo requerendo muita dedicação para a atividade prática (PRIOR, 2003). Quando a última não ocorre, acarreta reprovações e mal desempenho nas outras disciplinas que a tem como base (TOBAR et al, 2001). Assim, pode-se afirmar que a habilidade de escrever um programa funcional é um pré-requisito para ter um bom conhecimento na análise e desenvolvimento de sistemas (CHEANG et al, 2003).

Alguns dos pontos chave para o processo educacional são as avaliações e o *feedback*[1] (BENFORD et al, 1994). Estes recursos, além de beneficiar o estudante, proporcionam uma maneira melhor para o professor identificar as causas das reprovações em uma disciplina (HIGGINS et al, 2005).

Da mesma forma, segundo (PRIOR, 2003), um dos passos fundamentais para o bom aprendizado de programação é a realização de muitos exercícios práticos. Vale ressaltar a importância de um *feedback* eficiente, permitindo com que o aluno perceba não só onde está o erro, mas também entenda o que está errando. Isto é necessário, pois iniciantes em programação normalmente tem dificuldade em reconhecer suas próprias deficiências, uma vez que durante a explicação do professor muitos alunos deduzem que compreenderam o conceito apresentado em aula, mas durante a utilização dos mesmos, na resolução dos problemas, sentem dificuldade em aplicá-los (ALA-MUTKA, 2004).

[1] Retorno de informações relacionados à correção de uma avaliação, tanto para o aluno quanto para o professor.

Em turmas maiores, a utilização constante de avaliações e exercícios práticos torna-se inviável, devido a dificuldade do avaliador em produzir um grande volume de exercícios e retornar rapidamente o resultado das avaliação dos mesmos (MOREIRA; FAVERO, 2009). Esta é uma das razões da importância da utilização de um sistema para a aplicação e correção automática dos exercícios.

Dentre os benefícios de um sistema de correção automático pode-se destacar (KJOLLERSTROM; MARTENSSON, 2009):

- Melhor acompanhamento do desempenho individual dos alunos;
- Menor esforço por conta da ajuda da ferramenta;
- Uma melhor qualidade de ensino devido ao estímulo de uma maior atividade prática;
- Administração mais eficiente dos alunos e suas tarefas.

Assim, foi cogitado a utilização de sistemas de correção automática de programas para as disciplina de Técnicas de Programação 1 e Técnicas de Programação 2 da UNIRIO. Como todos os sistemas encontrados não tinham alguns dos requisitos buscados, foi decidida a implementação de um sistema com as seguintes características:

- Um mecanismo que permitisse a criação de questões e avaliações de maneira eficiente pelo professor;
- A aplicação e o monitoramento destas avaliações nas turmas;
- Realizar a correção de maneira automatizada;
- Possuir um bom feedback tanto para os alunos como para os professores;
- Ser online, e acessado por um navegador, o que permite ser utilizado de qualquer lugar, característica importante em função do problema das instituições não terem laboratórios suficientes para a aplicação de provas práticas;
- Facilidade para expansão e evolução do sistema, permitindo a inserção de novas funcionalidades, linguagens suportadas e ter uma boa escalabilidade;
- Uso das novas tecnologias de referência no mercado, na área de desenvolvimento;
- Emprego exclusivo de software livre no desenvolvimento;

Organização da monografia

A monografia é dividida em 8 capítulos. Este capítulo é uma introdução ao trabalho e sua motivação. No segundo capítulo são descritas as ferramentas e soluções atuais para o problema de autocorreção bem como os mecanismos usados pelas mesmas. Os detalhes da estrutura geral e as funcionalidades do sistema (AMAO) são apresentadas no terceiro capítulo. O quarto capítulo aborda as metodologias e tecnologias que foram utilizadas no desenvolvimento do sistema, no quinto capitulo são listadas as etapas de instalação do ambiente de desenvolvimento e do ambiente de produção, no capítulo seis são apresentados os diagramas e modelos, no sétimo capítulo são discutidas as ideias sobre desdobramentos ou possíveis melhorias no projeto, e as vantagens que estas podem trazer e no ultimo capítulo se encontra a conclusão do trabalho.

2 ESTADO DA ARTE

No campo de estudo sobre a aplicação de autocorreção em avaliações, é encontrado um grande número de ferramentas, além de diversas abordagens para tratar este problema.

Na maioria dos casos, na avaliação automática de problemas de programação são levadas em consideração duas métricas (MOREIRA; FAVERO, 2009):

1. O resultado do programa (se a saída é a esperada);
2. A complexidade da solução utilizada pelo aluno.

Além desses pontos, existem ferramentas que propõe tratar:

- do caso de similaridade entre avaliações (plágio);
- da execução segura dos programas avaliados e
- da facilidade na formulação de questões e/ou avaliações.

Nas seções seguintes serão descritos cada um desses critérios, bem como algumas das ferramentas existentes que os implementam. Vale destacar que a dificuldade de acesso ao código fonte dos softwares analisados impossibilitou uma melhor análise e comparativo entre estes.

2.1 Sistemas de Avaliação

ASAP

ASAP é um sistema para a avaliação automática de trabalhos de programação. Desenvolvido na Kingston University, Reino Unido, focado principalmente no aprendizado de Java (DOUCE; ORWELL, 2005).

ASSYST

O ASSYST é um sistema de avaliação semi-automatizado no qual humanos avaliam programas em C e Ada, onde a automação não é possível. No entanto, a natureza semi-automática do sistema, combinado com o método de apresentar soluções por e-mail, exclui o retorno de *feedback* imediato para os alunos (JACKSON; USHER, 1997).

AutoLEP

AutoLEP é um sistema automático de aprendizagem e avaliação para auxiliar o desenvolvimento de iniciantes em programação. Ele utiliza um mecanismo de avaliação que combina análise estática com testes dinâmicos para julgar o trabalho do aluno. Esse sistema é online e desktop, e foi usado no curso de programação de C no Harbin Institute of Technology, e muitas outras universidades desde 2004 (WANG et al, 2010).

BOCA

O BOCA (**B**OCA **O**nline **C**ontest **A**dministrator) é um sistema desenvolvido para dar apoio a competições de programação, construído em PHP e Postgresql para ser usado na Maratona de Programação da Sociedade Brasileira de Computação. O sistema é utilizado também no apoio a disciplinas nas quais é realizada a submissão e correção de trabalhos durante as aulas (CAMPOS; FERREIRA, 2004).

BOSS

BOSS é um sistema que serve para a apresentação e análise semi-automatizada de exercícios de programação. Este analisa as soluções em relação aos dados de teste definidos, retornando o percentual de acerto com o comentário para ajudar o aluno a corrigir o seu programa. Ele tem um apenas um mecanismo limitado para pontuação automática, baseado nos resultados dos testes (LUCK; JOY, 1999).

Codelab

Codelab é um sistema web em que os alunos apresentam suas respostas através de um software cliente. Uma das principais características do sistema é que o Codelab tendem a se concentrar em exercícios que possuem respostas simples e curtas, que se tornam mais complexas com a progressão do aluno. Os exercícios visam cobrir todos os conceitos dentro de cada tópico técnico antes de permitir que os alunos a sigam em frente (ARNOW; BARSHEY, 1999).

CourseMarker

CourseMarker é um sistema didático CBA(Computer Based Assessment), da Universidade de Nottingham. Os alunos usam-no para resolver exercícios de programação e apresentar as suas soluções. O CourseMarker retorna resultados imediatos, e permite os alunos refazerem a questões utilizando o feedback para obter resultados melhores (HIGGINS et al, 2005).

Petcha

Petcha é uma ferramenta que tem por objetivo funcionar como auxiliar de aprendizagem em cursos de programação, entre as suas tarefas ele realiza correção automática de exercidos. Seu objetivo principal é aumentar o numero de exercícios de programação resolvidos pelos estudantes, para isso possui duas tarefas principais: ajudar os professores na criação dos exercícios através do seu sistema e ajudar os alunos em resolvê-los por meio de funcionalidades que integram o sistema com as IDEs utilizada pelos alunos.(QUEIRÓS; LEAL, 2012)

RoboProf

RoboProf CBA é um sistema baseado na web que apresenta notas do curso e exercícios para os alunos, aceita submissões de estudantes, e testa estas submissões em relação aos dados de teste, fornece *feedback* para os alunos e arquivos os resultados. O feedback fornecido aos alunos pelo RoboProf é imediata, mas mínima. O *feedback* é majoritariamente abrangido por declarações correto / incorreto sobre a saída do código de estudante.(DALY, 1999)

Scheme-robo

Scheme-robo é um sistema de avaliação automática de exercícios de programação da linguagem Scheme, esse sistema avalia procedures individualmente ao invés do programa completo, sendo capaz de analisar outras propriedades do programa como: estrutura, tempo de execução e plágio (SAIKKONEN et al, 2001).

TRAKLA

O sistema CBA TRAKLA auxilia no aprendizado de estruturas de dados e algoritmos. Além de conter ferramentas de avaliação automática. O sistema emprega visualização, animação e simulação para apresentar conceitos para o aluno. TRAKLA apresenta o feedback as alunos através de e-mail (HYVÖNEN; MALMI, 1993).

The Huxley

Esta ferramenta, tem uma proposta bastante similar a do AMAO, entretanto já é mais madura e com muito mais funcionalidades disponíveis na mesma, é usada atualmente em algumas disciplinas de programação da UFAL (Universidade Federal de Alagoas).

Atualmente esse sistema é mantido por uma empresa nascida na UFAL e seu uso para estudantes é gratuito, bem como para professores de instituições públicas. Vale ressaltar que algumas das funcionalidades presentes no The Huxley são propostas de melhorias futuras para o AMAO, conforme o capítulo de Trabalhos Futuros.

2.2 Métodos de Analise de Programação

Nesta seção serão apresentados os métodos para a análise dos programas enviados pelos alunos. Estes podem ser categorizados em métodos de validação do resultado, validação da qualidade, de identificação de plágio, de execução segura dos exercícios e de facilitação nas criações de questões e avaliações.

2.2.1 Métodos de Validar o Resultado do Programa

O método de validação do resultado mais utilizado consiste em comparar se a saída esperada é igual a gerada.

Este método é considerado um dos mais básicos e essa abordagem é apenas a primeira medida de correção possível. É recomendado que o aluno seja avaliado também com base na qualidade da solução (MOREIRA; FAVERO, 2009).

Em (SAIKKONEN et al, 2001) é proposto um ambiente que ao invés de analisar a saída de cada programa, verifica a própria chamada aos métodos criados pelos alunos, eliminando os problemas relativos a formatação da saída.

2.2.2 Métodos de Avaliação da Qualidade do Programa

Para medir a qualidade do algoritmo, existem várias propostas, entre elas:

- Analisar se a estrutura do código está de acordo com o exigido, como ocorre no ambiente Scheme-robo (SAIKKONEN et al, 2001).
- Utilizar métricas de engenharia de software, que segundo (ALMEIDA, 2005) avaliam a complexidade de um código-fonte. Algumas dessas métricas são:
    - número de linhas do programa;
    - quantidade de uso de funções da linguagem;
    - número de palavras reservadas;
    - número de declarações;
    - complexidade de McCabe[2]; e
    - volume de Halstead[3].

Essas métricas de engenharia de software podem compor indicadores utilizados em um modelo de regressão linear para auxiliar na medição do nível de proximidade em termos de complexidade en0tre a resposta do aluno e do modelo (MOREIRA; FAVERO, 2009).

[2] A Complexidade de McCabe, também conhecida como Complexidade Ciclomática, é uma métrica de software usada para indicar a complexidade de um programa. Desenvolvida por Thomas J. McCabe em 1976, essa mede a quantidade de caminhos de execução independentes de um código fonte(McCABE, 1976).

[3] O Volume de Halstead (HALSTEAD, 1977) é parte das Medidas de Complexidade de Halstead, em que é calculado a partir da formula $\mathbf{V = N \times Log_2 n}$, onde **N** é Numero total de operadores e operandos, e **n** é Número de operadores e operandos distintos.

2.2.3 Métodos de Identificação de Plágio

Outra preocupação recorrente na avaliação do aprendizado em programação são os métodos para identificar plágios entre soluções de alunos.

Em (MOREIRA; FAVERO, 2009) pode ser observado a utilização de n-gramas para medir o quanto uma solução do estudante é similar a resposta-modelo.

N-gramas são sequências continuas de *n* itens de uma dada sequência de texto.

Dividindo duas sequências de caracteres em n-gramas, pode-se medir quantos n-gramas estas compartilham. Se estas possuem um alto índice de igualdade, pode-se dizer que houve plágio entre os textos (SUKKARIEH et al, 2003).

Uma ideia interessante, para tentar diminuir os plágios entre avaliações, é a criação de avaliações com questões diferentes, mas com o mesmo tipo de conteúdo, para cada aluno.

2.2.4 Métodos para Execução Segura dos Programas Avaliados

Na área da computação deve-se ter o cuidado com aquilo o que o usuário tem liberdade para fazer em um sistema. No caso de um corretor automático para programação, deve-se levar em consideração que o corretor executa os códigos dos alunos, que nem sempre são bem intencionados e podem enviar arquivos fontes maliciosos para beneficio próprio ou apenas para desestabilizar o serviço.

Assim todo o processo de correção deve ser realizado em um ambiente controlado, comumente chamado de sandbox(MARTINS et al, 2009). Dessa forma, os programas dos alunos podem apenas realizar operações que estejam dentro do controle desse ambiente.

Em (SAIKKONEN et al, 2001) foi implementado uma sandbox que executa os códigos dos alunos, na linguagem Scheme, em um ambiente com um interpretador Scheme metacircular especialmente desenvolvido para esse propósito. Tal interpretador possuí algumas funcionalidades como:

- desabilita acesso de leitura e escrita em arquivo (visto que para a implementação utilizada não era necessária tal funcionalidade).
- verificar existência de loops infinitos no código; e
- medir se o algorítimo funciona em tempo linear.

Outra abordagem muito interessante é a empregada em sistemas como BOCA (CAMPOS; FERREIRA, 2004) e Mooshak[4], em que todos os programas do aluno são executados utilizando um programa chamado SafeExec.

Esse programa, feito em C, executa os códigos como um usuário desprivilegiado e com diversos tipos de controle, tais como:

- o numero máximo de threads que podem ser abertas pelo programa;
- o tamanho máximo de memória que pode ser utilizada; e
- o tempo máximo de execução, entre outros.

2.2.5 Métodos para Facilitar a Criação de uma Questão e/ou Avaliação

A proposta de um sistema de correção automática é facilitar tanto as tarefas para o professor quanto para o aluno.

No sistema Schema-robo como este utiliza uma abordagem diferenciada de correção, os professores devem aprender como criar uma questão que possa ser utilizada nesse corretor. Tal aprendizado pode ser penoso, dificultando a criação de um gabarito podendo não ser mais vantajoso o custo/beneficio do processo.

Uma maneira desse sistema tornar essa atividade menos custosa é através do reaproveitamento das questões que já foram criadas. Dessa forma, uma questão criada por um professor A, poderia ser utilizada novamente por um professor B, e o custo da mesma é contabilizado apenas uma vez.

2.3 Escalabilidade e Implementação de Novos Recursos

Vale salientar, a importância da facilidade de se escalar um sistema, e acoplar novos módulos a este. Pois este deve poder ser utilizado tanto em um ambiente pequeno, em uma turma ou uma matéria, como também deve ter a capacidade para ser facilmente expandido para suportar uma maratona de programação de um país (como é o caso do sistema BOCA (CAMPOS; FERREIRA, 2004).

Outro ponto importante, mas muitas vezes ignorado nas ferramentas de correção automática de programação, é a facilidade em implantar novos recursos e funcionalidades. Como muitas vezes são sistemas que tem o foco voltado ao problema da autocorreção, essa característica acaba sendo ignorada. Assim, alguns sistemas tem componentes tão amarrados, que se fosse necessário implementar o suporte a uma nova

[4] http://mooshak.dcc.fc.up.pt/

linguagem de programação, seria necessário criar um sistema novo, reduzindo o reaproveitamento do que já foi feito. Tal preocupação pode ser vista em sistemas como o CourseMaker que foi proposto levando-se em consideração a inclusão de novas funcionalidades (FOXLEY et al, 2001).

3 O SISTEMA AMAO

Neste capitulo será descrito de forma geral o sistema AMAO (**AM**biente de **A**valiações **O**nline), com um posterior aprofundamento de cada um dos seus módulos. Esse sistema foi implementado com o objetivo de suprir as necessidades descritas anteriormente. Isto é, um ambiente de autocorreção com:

- facilidade para expansão e evolução do sistema, permitindo a inserção de novas funcionalidades, suporte a outras linguagens e escalabilidade;
- mecanismo que permita a criação de questões e avaliações de maneira eficiente pelo professor;
- aplicação e o monitoramento destas avaliações nas turmas;
- realização da correção de maneira automatizada;
- existência de um bom *feedback* tanto para os alunos quanto para os professores;
- funcionamento online e acessado por um navegador; e
- utilizar exclusivamente softwares livres de referência no mercado.

3.1 Visão Geral do Sistema

O AMAO possui duas visões diferentes de usuário, uma para o professor e outra para o aluno e ambas serão explicadas detalhadamente nas seções 3.2.2 e 3.2.3 respectivamente.

Em linhas gerais o sistema possui a seguinte forma de funcionamento: na visão do professor, primeiramente são incluídas as questões de programação no Banco de Questões. Para tal, o professor deve primeiro definir ou criar (caso não exista) o tipo em que a questão incluída se enquadra, permitindo assim uma fácil categorização destas questões. O próximo passo é informar os dados, enviar os códigos fonte e os arquivos de entrada que serão testados para a questão adicionada.

Em seguida é preparada a avaliação de uma turma, definindo informações sobre a data de inicio e término. Então serão escolhidas quais questões estarão presentes nessa avaliação. O professor pode selecionar especificamente uma que deseje utilizar ou pode usar filtros para obter alguma questão dentre as armazenadas no sistema que se enquadre nos critérios estabelecidos. Os filtros são baseados nos tipos de questões criados anteriormente.

A partir do momento que a data da avaliação criada expirar, o professor pode consultar a avaliação realizada por cada aluno, ver suas respostas e implementação da programação de cada questão, e se achar necessário, alterar a nota e deixar notas de revisão para cada estudante.

Vale ressaltar que as questões presentes no Banco de Questões podem ser editadas e utilizadas em novas avaliações, inclusive por diferentes professores.

Já na visão do aluno é possível visualizar diversas listas de avaliações: as futuras, as que estão dentro do prazo e que ainda não foram iniciadas, as já iniciadas e as já concluídas. Para começar uma avaliação o aluno seleciona a que deseja na lista de avaliações disponíveis para início. Em seguida este seleciona qual questão de sua avaliação deseja responder primeiro.

Para responder uma questão, o aluno deve enviar seus arquivos fonte e, se for o caso, escolher as respostas das opções múltipla escolha presentes. Uma vez que este passo é concluído solicita-se corrigir a programação, para que o aluno possa ter instantaneamente um feedback do resultado da correção automática da parte de programação da questão. O restante das correções, como a parte de múltipla escolha e a discursiva, o aluno só terá acesso depois da avaliação ser encerrada.

3.1.1 Ambiente Online

Com o intuito de facilitar o uso da ferramenta por um usuário em diversas plataformas, bem como para possibilitar o acesso ao sistema mesmo em máquinas de baixa performance optou-se pelo desenvolvimento em plataforma web. Desta forma a utilização da ferramenta seria viabilizada mesmo em instituições cujos laboratórios de computação não possuem muitos recursos, além de possibilitar o seu uso no âmbito de disciplinas EAD[5].

[5] Ensino à Distância

3.1.2 Corretor Automático

Para a correção semi-automática de questões de programação, optou-se pela adoção de uma abordagem com base no sistema BOCA. Que funciona como um teste de caixa preta, onde o professor fornece na criação da questão: o código fonte, o gabarito do exercício e as entradas para serem testadas.

No momento da correção do exercício, o sistema compila e executa o código que o aluno enviou. Em seguida compara a saída gerada pelo o aluno com a saída do programa gabarito.

Desta maneira, a criação e resolução de questões e avaliações mantém-se similar a forma em que professores e alunos já estão acostumados a fazer no papel, deixando o sistema mais amigável ao usuário neste quesito.

O diagrama de atividades da Figura 1 apresenta em detalhes todo o processo que ocorre durante a correção automática de uma questão de aluno.

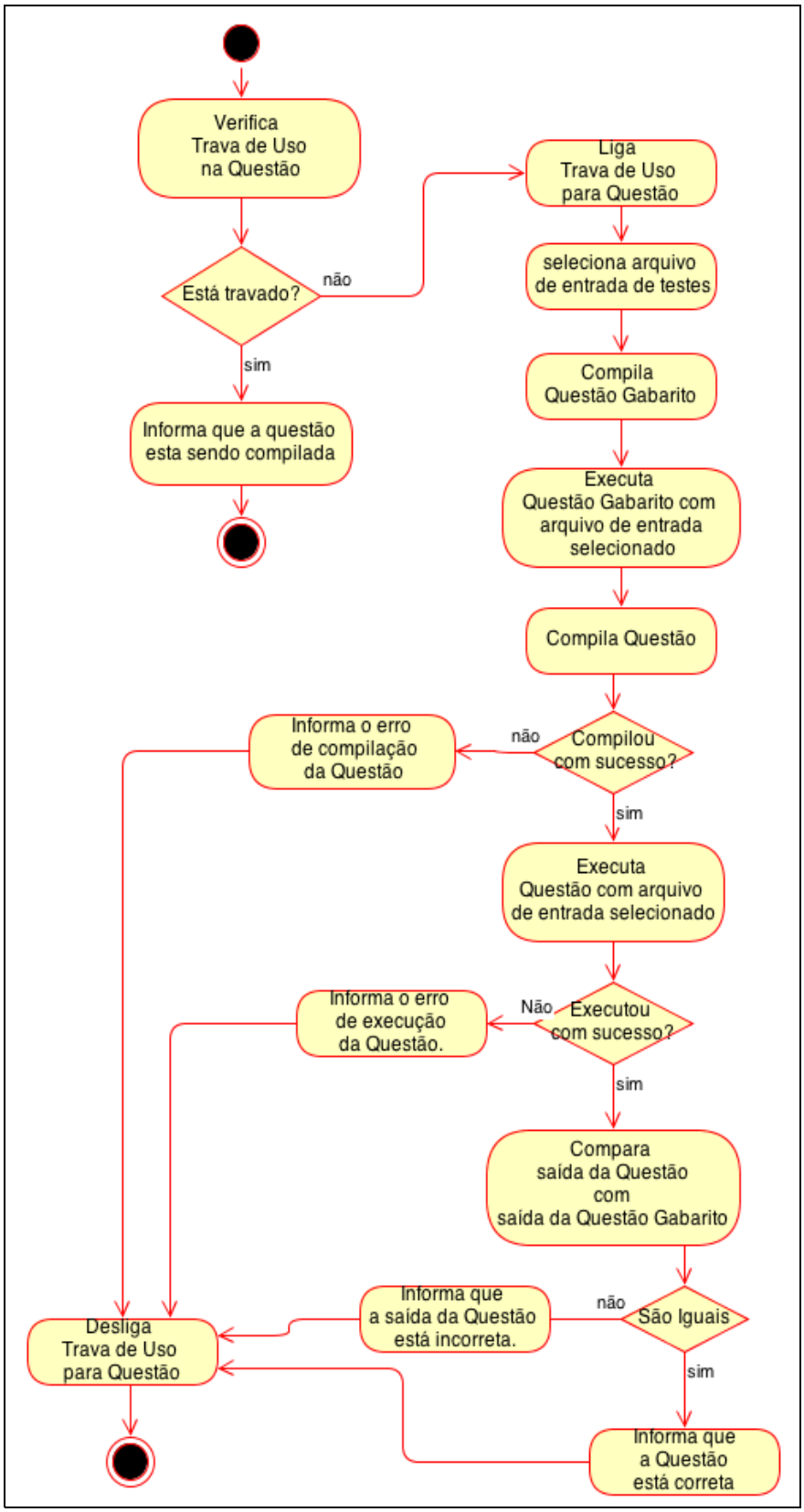

Figura 1 - Diagrama de atividades do processo de correção automática de uma questão de aluno.

3.1.3 FeedBack

Com o intuito de proporcionar um adequado *feedback* da autocorreção, tanto para os alunos quanto para os professores, separou-se o retorno da correção em cinco categorias.

- **Correto**: Quando a questão está totalmente correta.
- **Erro de Saída**: Quando o programa do aluno não tem a mesma saída do gabarito.
- **Erro de Execução**: Quando o programa não executou ou teve um término inesperado.
- **Erro de Compilação:** Quando o programa não compilou com sucesso.
- **Erro de Lock**: Quando o programa já está sendo corrigido e ainda não acabou todas as etapas da correção.

O tratamento de cada uma dessas cinco categorias de retorno, no processo de correção automática pode ser melhor visualizado no diagrama de atividades da Figura 1.

Com essa categorização dos possíveis resultados da correção tanto o aluno quanto o professor conseguem entender melhor em que ponto o programa não está executando como deveria, possibilitando um maior entendimento dos passos necessários para o devido acerto no código.

3.2 Detalhamento do Sistema

A seguir serão detalhados os diversos pontos do sistema AMAO, com suas telas e dados explicados primeiramente com a visão do professor, e por fim a do aluno.

3.2.1 Autenticando no Sistema

Com o objetivo de proporcionar mais segurança no sistema, é realizada uma autenticação onde o usuário (aluno ou professor) deve informar seu email cadastrado e senha, conforme a Figura 2:

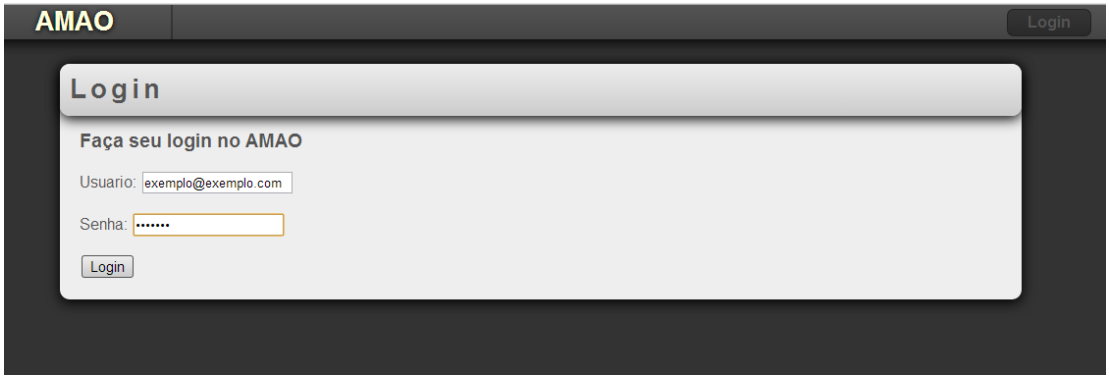

Figura 2 - Tela de login do sistema

Dada a autenticação, o usuário terá a visão do sistema dependendo da sua categoria (seção 3.2.2 e seção 3.2.3).

3.2.2 Visão do Professor

Um usuário cadastrado no sistema na categoria "professor" pode criar questões, tipos de questões e avaliações. Também é possível consultar, editar, excluir e revisar questões e avaliações já armazenadas.

Cada uma dessas funcionalidades é descrita nas seções seguintes.

3.2.2.1 Criação de Questões

Para adicionar uma questão no sistema, é necessário acessar a área de **Criar Conteúdo** e em seguida a opção **Criar Questão** (Figura 3).

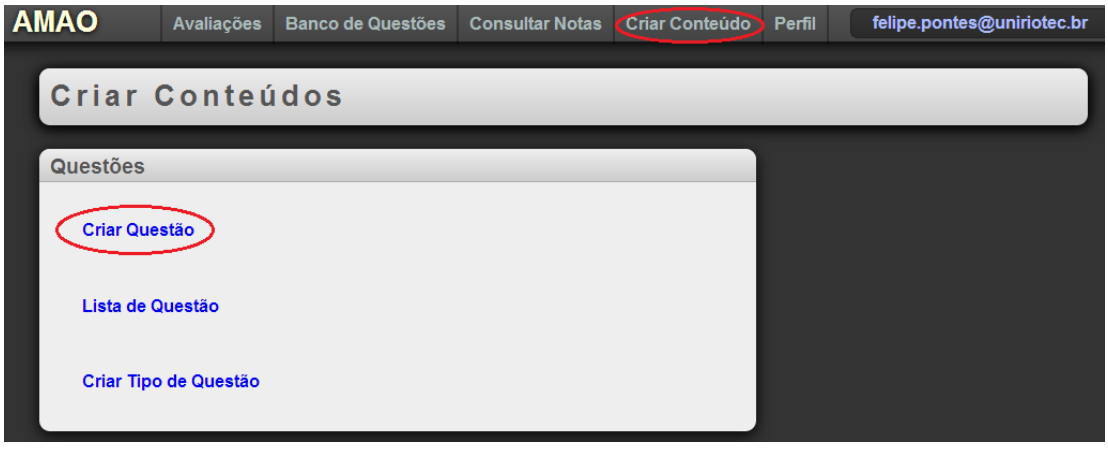

Figura 3 - Tela para criação de conteúdo

Na elaboração de uma questão é necessário a inserção de informações, que podem ser organizadas em 5 categorias:

a) Informações Gerais da Questão;

b) Informações Relacionadas a Correção Automática;

c) Informações sobre Múltipla Escolha;

d) Informações sobre Discursiva; e

e) Informações de Pontuação.

Na Figura 4 é apresentado um diagrama com o intuito de proporcionar uma visão geral da criação de uma questão pelo professor. A seguir é detalhado como estas informações serão inseridas no sistema.

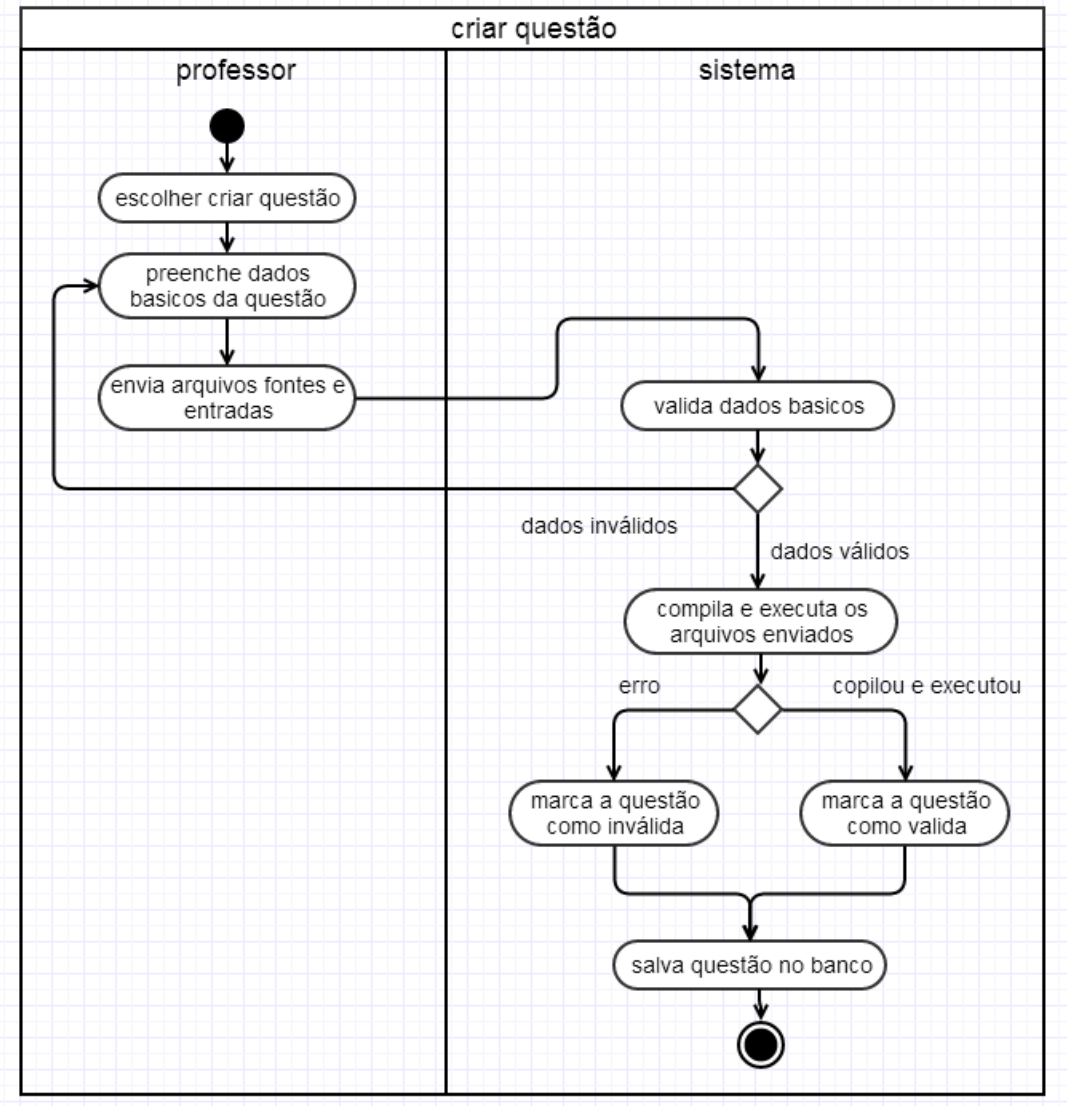

Figura 4 - Diagrama de atividades para a criação de uma questão pelo professor

a) Informações Gerais da Questão

Nessa categoria são informados o título da questão, o enunciado, os tipos a qual ela pertence e o corretor a ser utilizado, conforme pode ser visualizado na Figura 5.

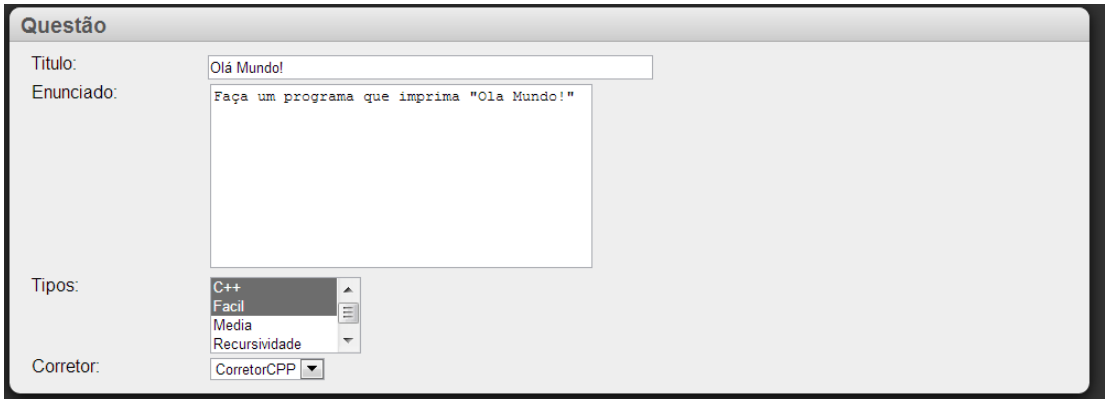

Figura 5 - Tela de informações sobre a questão

b) Informações Relacionadas a Correção Automática

Neste ponto são informados dados relacionados à correção automática da questão, tais como arquivos fontes de gabarito e arquivos de entrada que testam se o programa de um aluno está respondendo corretamente.

Ao incluir um arquivo fonte, é possível selecionar a opção de utilizar o mesmo na resolução. Desta forma quando o aluno for resolver a questão, automaticamente ele receberá este arquivo fonte junto com seus próprios, sendo que este não pode ser alterado pelo aluno. Essa funcionalidade permite que os professores criem questões em que se defina um ou mais arquivos fontes fixos e os alunos devem fazer os que estão faltando. Um exemplo de enunciado que usa esse tipo de recurso seria:

> *"Dado um arquivo fonte **"main.cpp"** escreva uma biblioteca com nome **"helloworld.h"** que tenha uma função **"imprimir_mensagem"**. Essa função deve imprimir a mensagem **"Olá Mundo!"**."*

Neste exemplo, quando o professor enviar os arquivos fontes, selecionará o arquivo "*main.cpp*" para ser utilizado pelos alunos, que ao resolverem a questão terão automaticamente esse arquivo entre os seus fontes, faltando apenas que eles elaborem e enviem seus próprios "*helloworld.h*". (Figura 6)

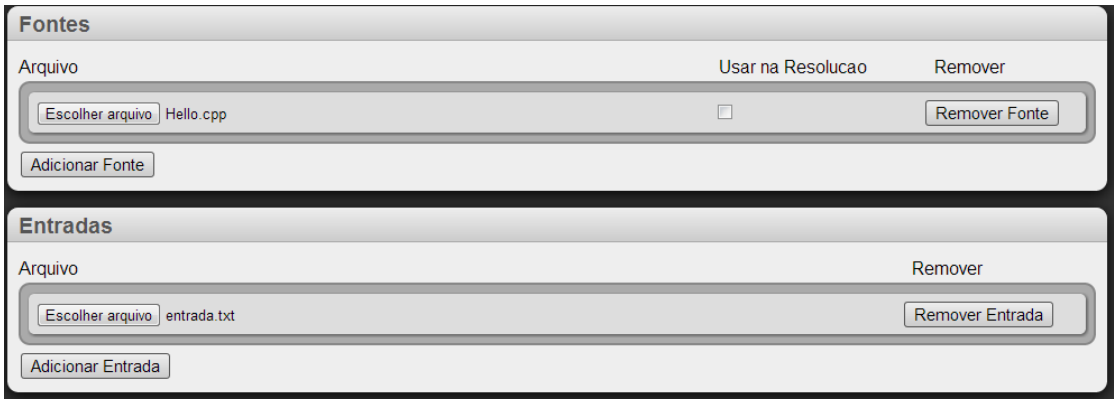

Figura 6 - Tela para inclusão de arquivos a serem usados na resolução da questão

c) Informações sobre Múltipla Escolha

Na formulação de uma questão tem-se a possibilidade de colocar na mesma opções de múltipla escolha em conjunto com a parte de programação. Basta preencher o campo "Opção" com o texto que será exibido naquela opção e determinar quanto ela vale no campo "Correta".

Pode-se também determinar que uma opção ao ser escolhida pelo aluno, anulará qualquer outra escolhida e apenas a pontuação da mesma é considerada. Para isso marca-se o campo "Anular".

A Figura 7 é um exemplo no qual a questão terá as seguintes opções de resposta: "1 + 1 = 2", "2 + 2 = 4" e "0 = 12345". Tanto a primeira quanto a segunda opções são corretas e portanto contabilizam grau 1.00 de pontos ao aluno, enquanto que a última contabiliza 0.00 pontos ao aluno pois está incorreta. Como na última opção foi assinalado o parâmetro "**Anular**", então se esta opção for escolhida pelo aluno isso fará com que qualquer outra opção também selecionada, tenha sua pontuação ignorada.

Figura 7- Tela para questão de múltipla escolha

d) Informações sobre Questão Discursiva

O professor pode também solicitar que alunos respondam alguma pergunta discursiva, porém a correção da mesma fica fora do escopo da correção automática.

Dessa forma esta não é contabilizada na nota do aluno automaticamente. No momento em que o professor revisar as avaliações, este deve corrigir as respostas discursivas manualmente (Figura 8).

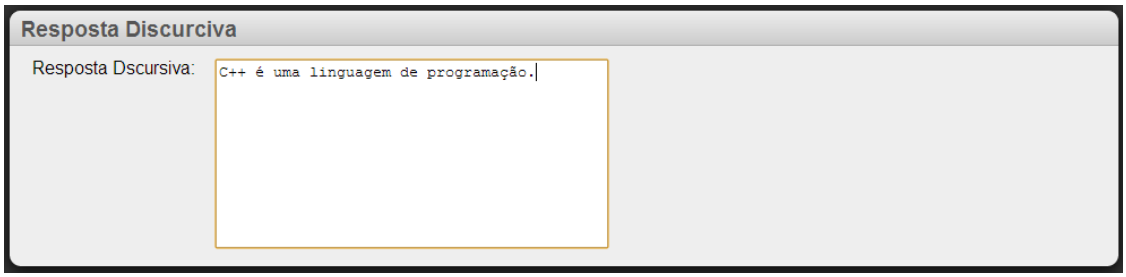

Figura 8 - Tela para inserção de questões discursivas

e) Informações de Pontuação

Neste item o professor deve informar qual a proporção que cada parte da questão terá na nota da mesma.

No exemplo abaixo, 80% da nota total da questão será dada pela parte de programação, 10% pelas opções de múltipla escolha, e 10% pela discursiva (vide Figura 9).

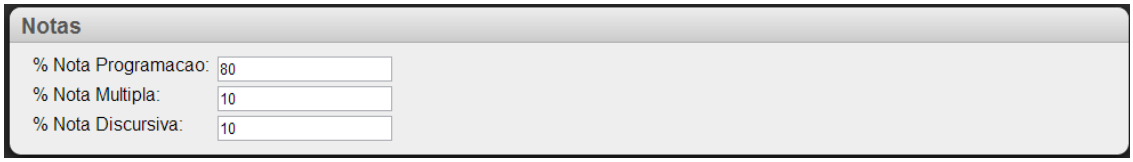

Figura 9 - Tela para definição da pontuação das questões

3.2.2.2 Criação de Tipo de Questão

Se ao criar uma nova questão o professor verificar que esta não se enquadra em nenhum tipo de questão disponível, basta o professor criar um novo tipo.

Para criar um novo tipo, o professor deve acessar a tela de "**Criar Conteúdo"** e selecionar a opção **"Criar Tipo de Questão"** (vide Figura 10).

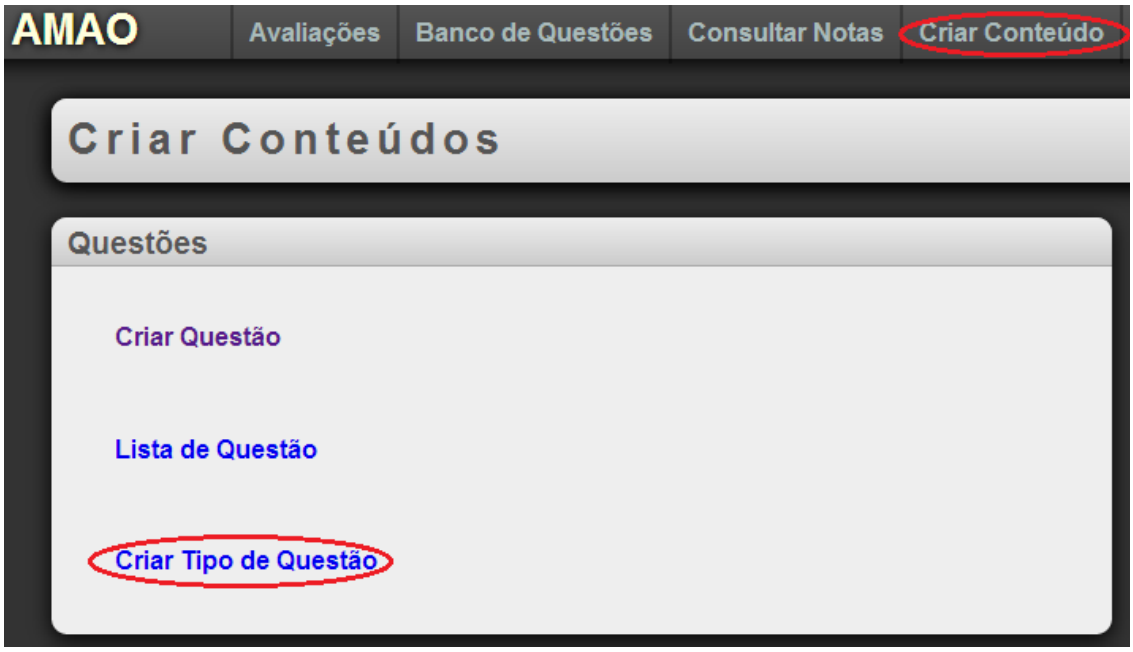

Figura 10 - Tela com Opção de menu para criar tipo de questão

Uma vez na tela de criação de Tipo de Questão, o professor deve informar qual o novo tipo, e se ele tem algum tipo pai, ou seja, se esse tipo é uma especialização de algum outro tipo, conforme pode ser visto na Figura 11.

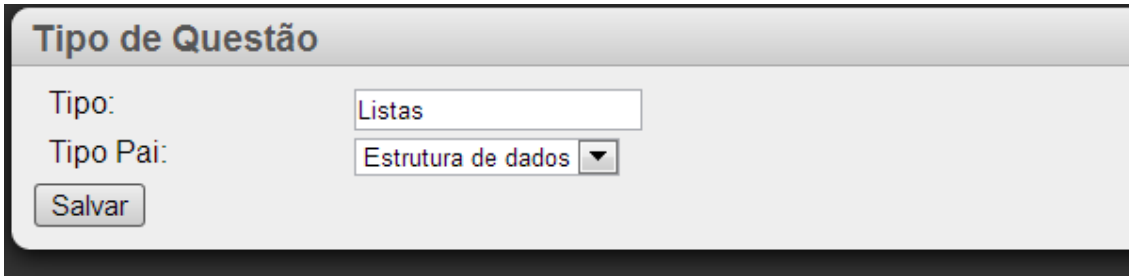

Figura 11 - Tela para detalhamento do tipo de questão

3.2.2.3 Banco de Questões

Para verificar quais questões já estão disponíveis no ambiente, o professor deve acessar a tela de Banco de Questões, conforme ilustrado na Figura 12.

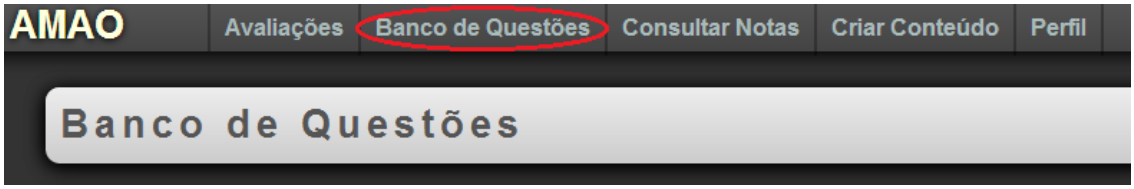

Figura 12 - Opção de menu para o Banco de Questões

No sistema AMAO foi implementado um Banco de Questões nos mesmos moldes disponibilizados no sistema Moodle (MOODLE, 2012). Dessa forma as questões cadastradas nesse Banco podem ser reutilizadas em diversas avaliações por diferentes professores e em uma variedade de matérias, como ilustrado no diagrama na Figura 13.

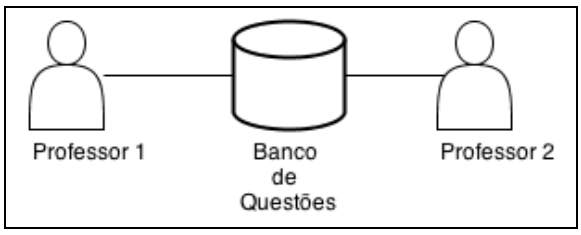

Figura 13 - Esquema representativo do uso do Banco de Questões

Este recurso reduz a repetição desnecessária no trabalho de elaboração de várias questões similares. O professor utiliza filtros, que serão detalhadamente abordados na seção 3.2.2.4, para selecionar questões já existentes no Banco de Questões a fim de compor uma avaliação.

Com o objetivo de facilitar a obtenção de uma determinada questão ou um determinado tipo, os filtros foram organizados em um menu. Os filtros são dispostos por título, situação de validação e seleção de tipos de questão.

A Figura 14 exemplifica a utilização de filtros para obter questões que possuem "Olá Mundo" em seu título, que sejam válidas e que sejam do tipo "Fácil", ou seja, que atendam a combinação dos três critérios.

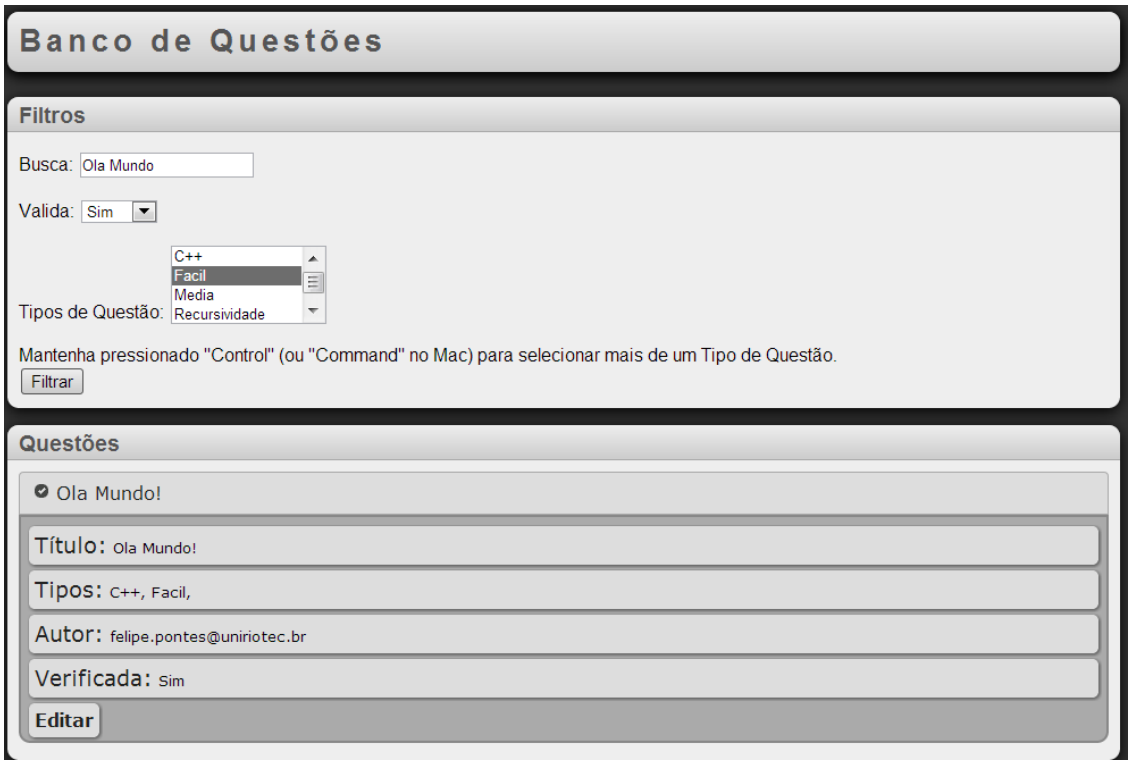

Figura 14 – Exemplo de uso da função de busca de questão com filtro

Depois de filtrar questões, se houver alguma com os critérios utilizados, uma lista aparece logo abaixo dos filtros. A fim de melhorar a visualização de busca, as questões filtradas apareceram de forma comprimida, ou seja, apenas o Título e um ícone representando se as questões são válidas ou não.

Se o avaliador preferir mostrar mais detalhes de uma questão, pode clicar no título da mesma, o que faz com que sejam exibidos mais dados sobre a questão, como mostra a Figura 15. Se quiser obter ainda mais detalhes ou editar clica-se, então no botão **"Editar"**.

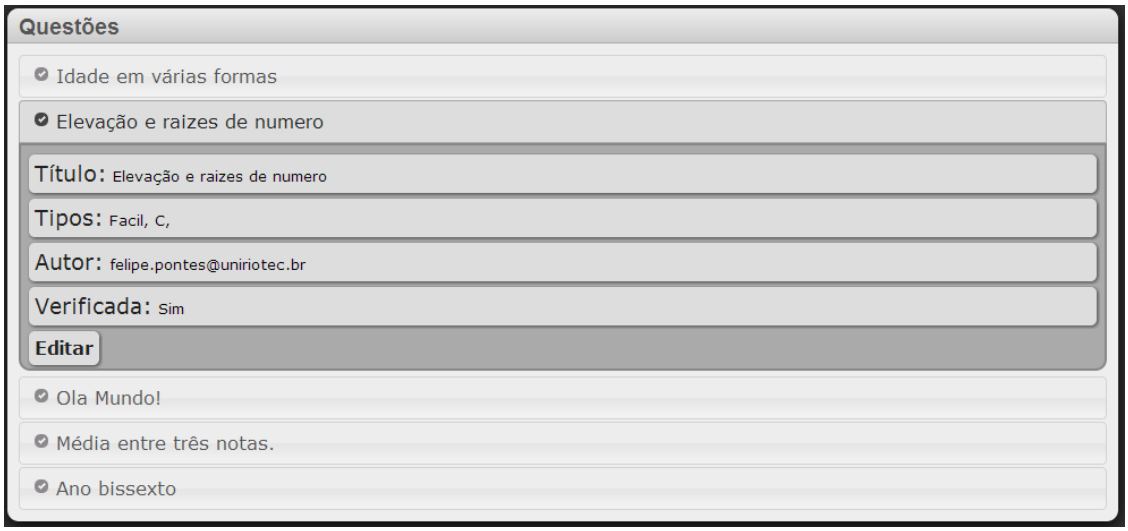

Figura 15 - Tela com detalhes da questão obtida com o filtro

3.2.2.4 Criação de Avaliações

Para criar uma nova avaliação o avaliador deve acessar a tela de "**Criar Conteúdo"** e selecionar a opção "**Criar Avaliação"** (vide **Figura 16**).

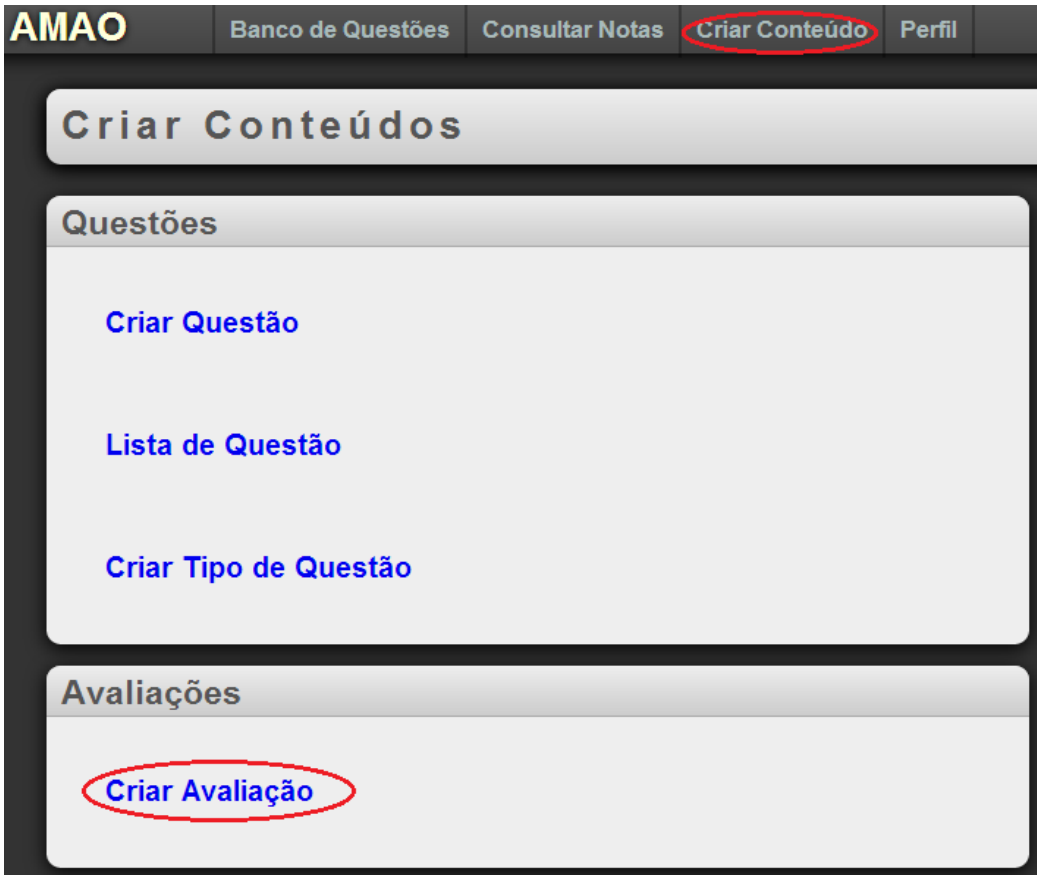

Figura 16 - Opção de menu para criar avaliação

.

Na tela de **Criar Avaliação** devem ser definidos os dados gerais da avaliação que são: título, turma, data de início e de término Figura 17.

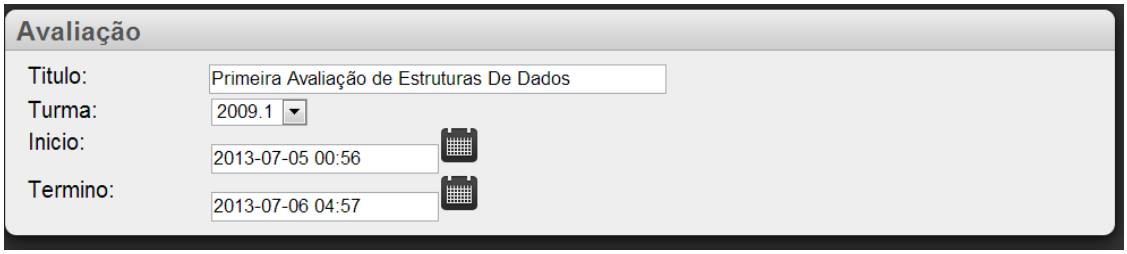

Figura 17- Tela com informações sobre a avaliação

Para facilitar a seleção das datas e horas de início e término de uma avaliação, o professor pode clicar nos ícones de calendário ao lado dos campos do formulário. Isso fará com que seja exibido um calendário onde é possível selecionar o dia, mês, ano e hora. Vide Figura 18.

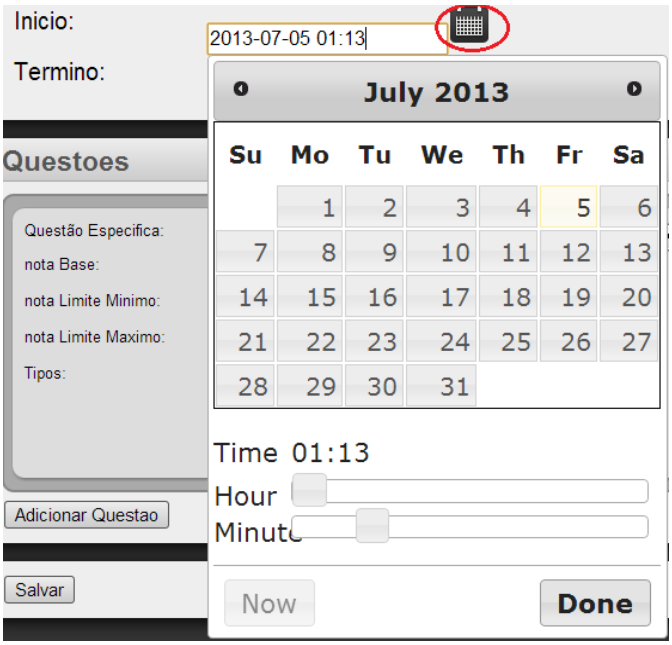

Figura 18 - Tela de seleção da data e hora da avaliação

Em seguida deve ser escolhido quais questões farão parte da avaliação. Como explicado anteriormente, observou-se que seria necessário uma maneira simples para que os professores produzissem avaliações. Assim, seria preciso um dinamismo na avaliação, onde se pudesse gerar provas com questões diferentes dentro do mesmo contexto de características e grau de dificuldade.

Para incluir uma nova questão na avaliação, existem duas formas distintas de abordagem. A seleção da questão específica dentre as armazenadas. Ou a seleção de um ou mais tipos de questão, deixando que o sistema selecione aleatoriamente questões que se enquadrem nos critérios esperados.

Para escolher uma questão específica, uma questão do campo **"Questão Específica"** deve ser selecionada conforme indicado na Figura 19.

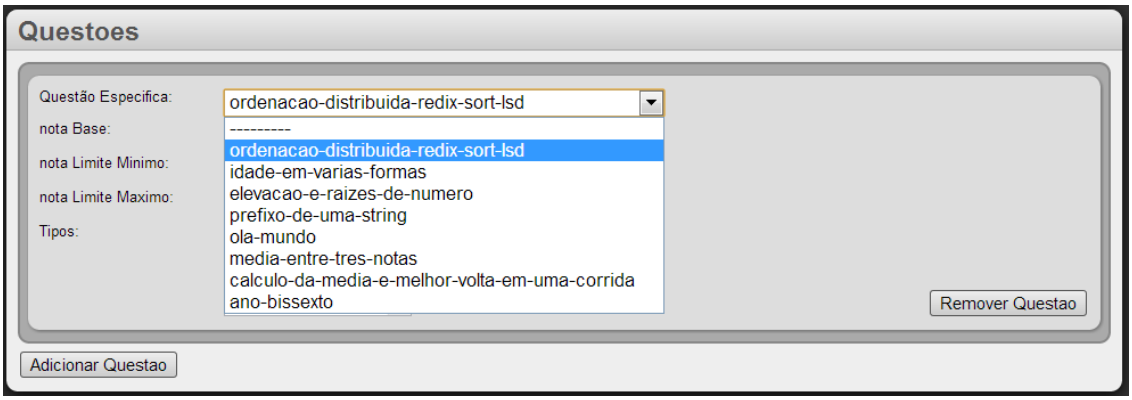

Figura 19 - Tela de seleção de questão

Ao selecionar essa opção, todos os alunos que fizerem essa avaliação terão obrigatoriamente esse exercício em suas provas.

Caso prefira deixar o AMAO encarregado de escolher a questão atual, o professor deve selecionar em quais tipos essa questão deve se enquadrar. Como pode ser observado na Figura 20. O uso da opção de selecionar os tipos faz com que, cada vez que um aluno for realizar esta avaliação, o sistema selecione aleatoriamente alguma questão do Banco que tenha os tipos selecionados.

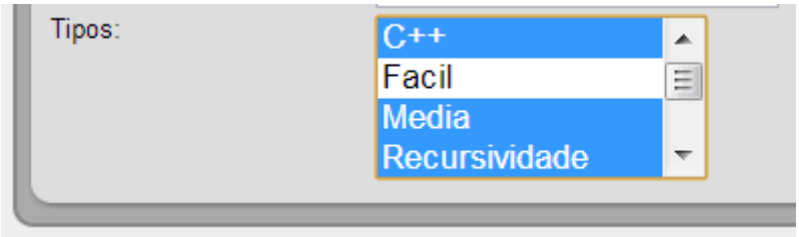

Figura 20 - Seleção de tipo de questão para indicar automaticamente uma questão de forma aleatória

Para cada questão também é necessário preencher dados sobre a sua pontuação, que consistem em três valores, a **Nota Base**, que define quanto o aluno receberá se conseguir 100% da questão, a **Nota Limite Minimo**, que é o mínimo que o aluno pode

adquirir na questão, pois é possível criar uma questão que possua valor negativo, e por fim **Nota Limite Máximo** para o valor máximo que o aluno pode conseguir na questão, pois é possível criar um questão que permite pontuação extra, que vai além da nota base (Figura 21).

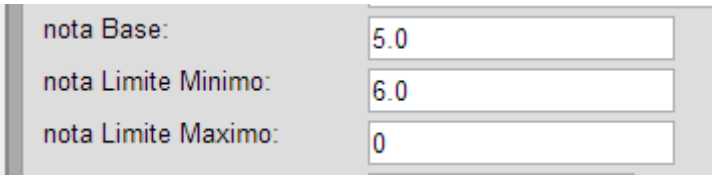

Figura 21- Definição dos pontos por questão

3.2.2.5 Revisão de Avaliação e de Questão

No ponto em que uma avaliação criada tenha sido encerrada, isto é, depois que expirar a data e o horário de término da avaliação, o professor pode consultar as respostas de cada aluno. Para isso acessa-se a página de **"Consultar Notas"** como na Figura 22.

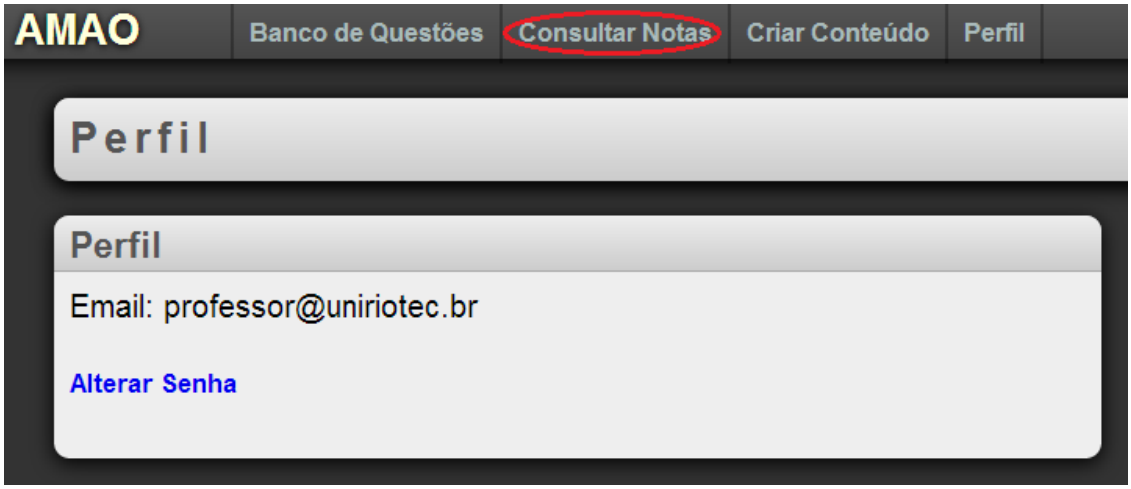

Figura 22 - Opção de menu para consulta de notas

Desta forma serão exibidas todas as avaliações já encerradas, juntamente com seus dados básicos como observado na Figura 23.

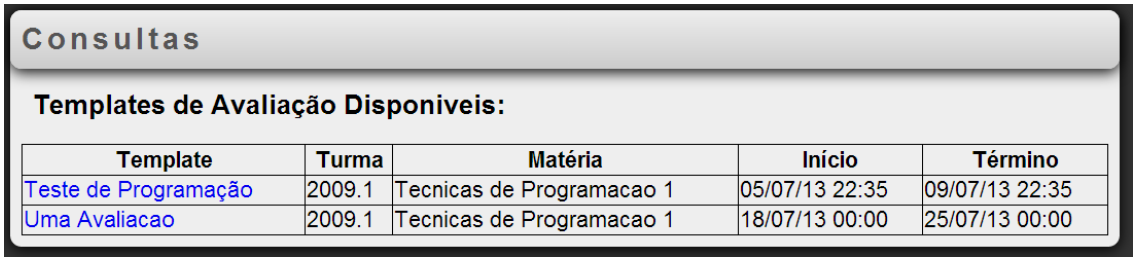

Figura 23 - Tela de consulta de avaliações de turmas

Clicando nos links dos títulos de cada avaliação, o professor poderá acessar mais dados sobre aquela avaliação.

Nesta nova tela (vide Figura 24) é encontrado um link para exibir novamente todas as avaliações de turma disponíveis com informações gerais da Avaliação de Turma selecionada, tais como seu título, nota máxima, nota mínima, e outros; e por fim uma listagem das avaliações que cada aluno fez, mostrando a matrícula do estudante e sua nota final neste teste.

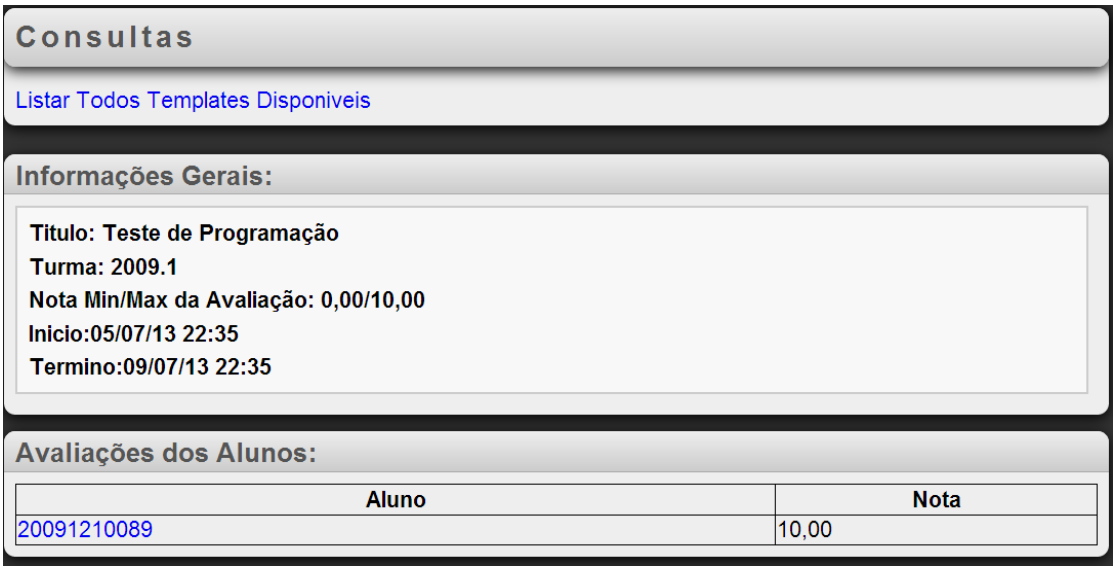

Figura 24- Tela detalhando uma avaliação de turma

É possível consultar os dados mais detalhados sobre a avaliação de cada aluno clicando no número da matrícula do estudante desejado, assim, será carregada uma nova tela, conforme a Figura 25. Em que se pode ter a visualização de cada questão do aluno e sua respectiva nota.

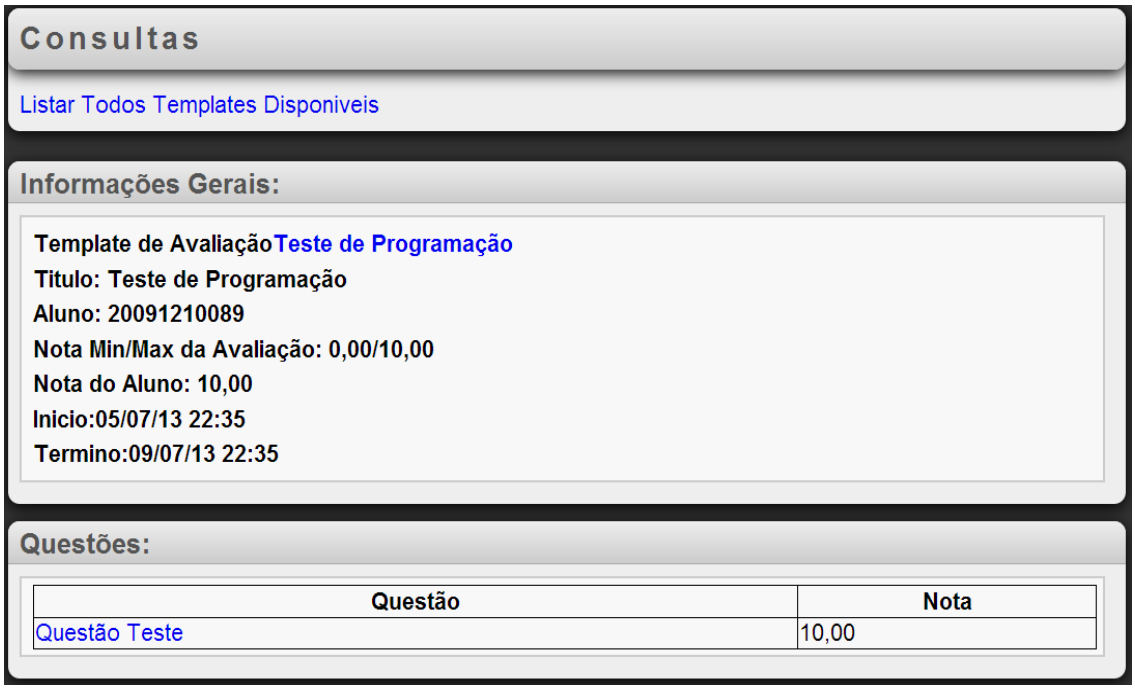

Figura 25 - Tela com visualização de cada questão do aluno

Novamente é possível um detalhamento maior clicando-se no link do titulo da questão, então é apresentado uma tela com todas as informações da questão escolhida.

As informações sobre a questão listadas são seu título, enunciado, nota mínima e máxima, nota total do aluno e os comentários de revisão do professor (Figura 26).

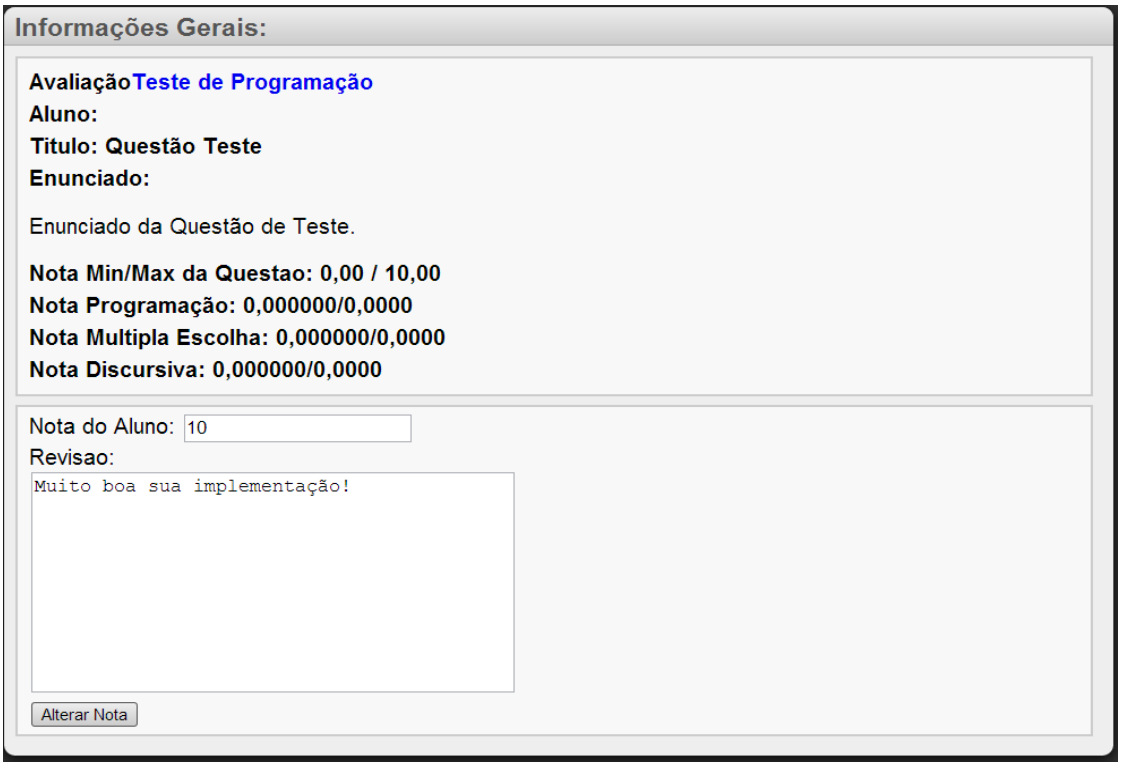

Figura 26 - Tela com Informações Gerais da questão

São disponibilizadas também as respostas do aluno, isto é, seus arquivos fonte, opções múltipla escolha, e resposta discursiva, conforme a Figura 27.

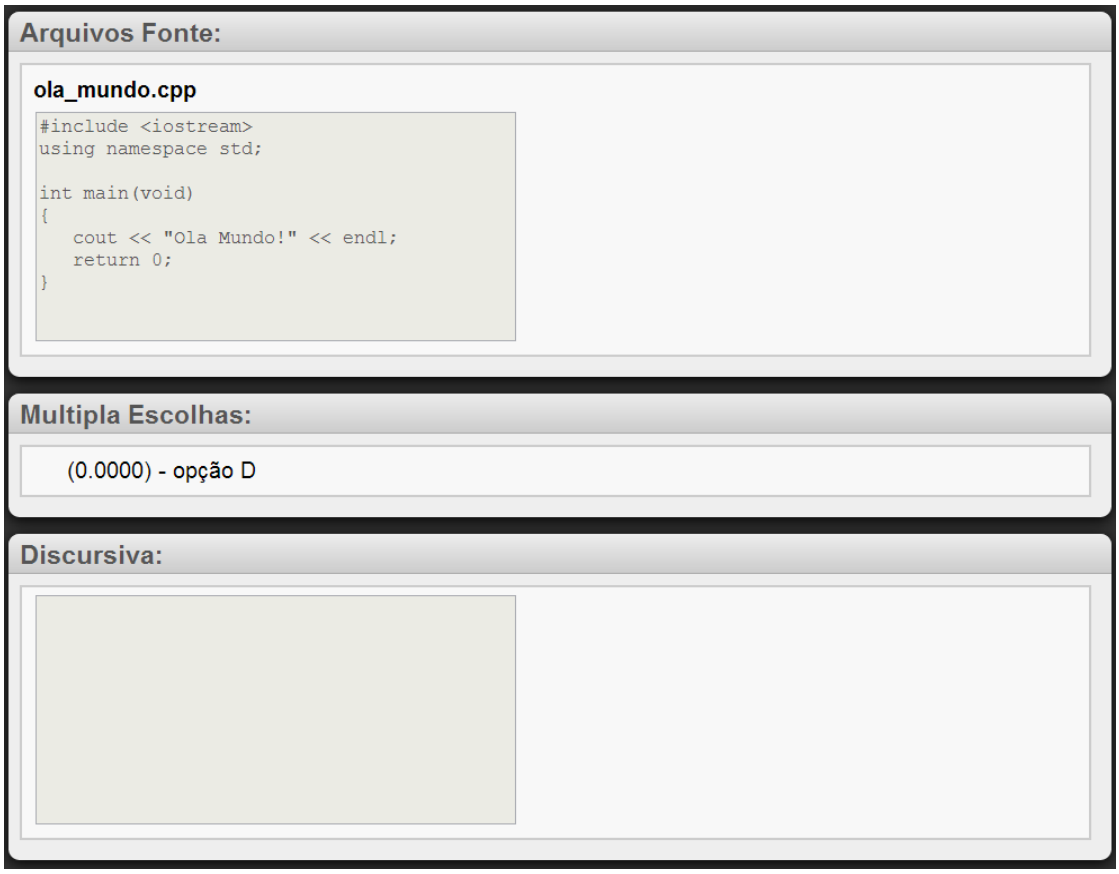

Figura 27 - Visualização da resposta do aluno

Portanto, o professor tem visão completa da resolução de cada aluno. Podendo assim, se julgar necessário, alterar a nota e deixar uma mensagem de revisão que o aluno terá acesso posteriormente.

3.2.3 Visão do Aluno

Um aluno pode ver e realizar as avaliações das quais ele faz parte, e ainda consultar revisões ou gabaritos daquelas que já foram encerradas, conforme será detalhado nas telas que se seguem.

3.2.3.1 Avaliações

Para acessar a tela de avaliações do aluno, este deve fornecer inicialmente seu login e senha (Figura 28).

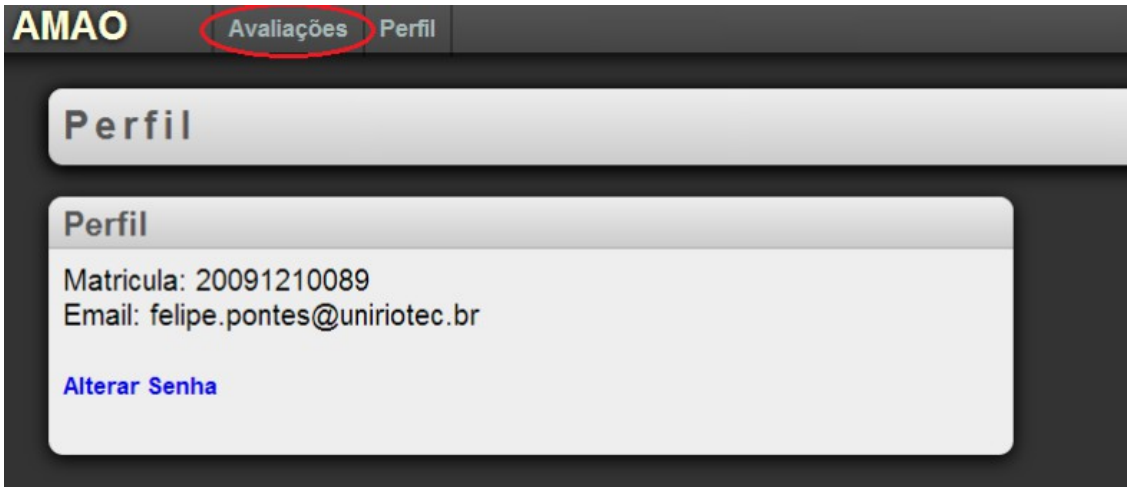

Figura 28 - Tela de acesso as avaliações do aluno

Acessando a área de avaliações pelo menu, o aluno verá uma tela onde serão listadas as avaliações divididas em 4 categorias, considerando o estado em que a avaliação se encontra conforme pode ser visto na Figura 29, e organizadas da seguinte forma:

- **Avaliações Futuras**

  As avaliações futuras são as avaliações que já foram preparadas pelo professor para a turma do aluno, porém ainda não chegou a data de realização.

  Para o aluno estas ficam listadas apenas para informação.

- **Avaliações para Iniciar**

  São as avaliações que já atingiram a data de inicio, entretanto o aluno ainda não começou a realização da prova.

  Ao escolher uma avaliação que está nesta lista, o aluno é levado para a página de realização da prova.

- **Avaliações em Andamento**

  São as avaliações que ainda tem tempo para serem realizadas e que já foram acessadas pelo aluno pelo menos uma vez.

  Ao escolher uma avaliação que está nesta lista, o aluno é levado para a pagina de realização da prova, para continuar a avaliação.

- **Avaliações Terminadas**

  As avaliações cujos prazos de término já foram ultrapassados serão listadas aqui, independente se o aluno já acessou e realizou uma questão ou não.

  Ao escolher uma avaliação que está nesta lista, o aluno é levado para uma página onde poderá consultar as questões da prova, mas não poderá mais realizá-la.

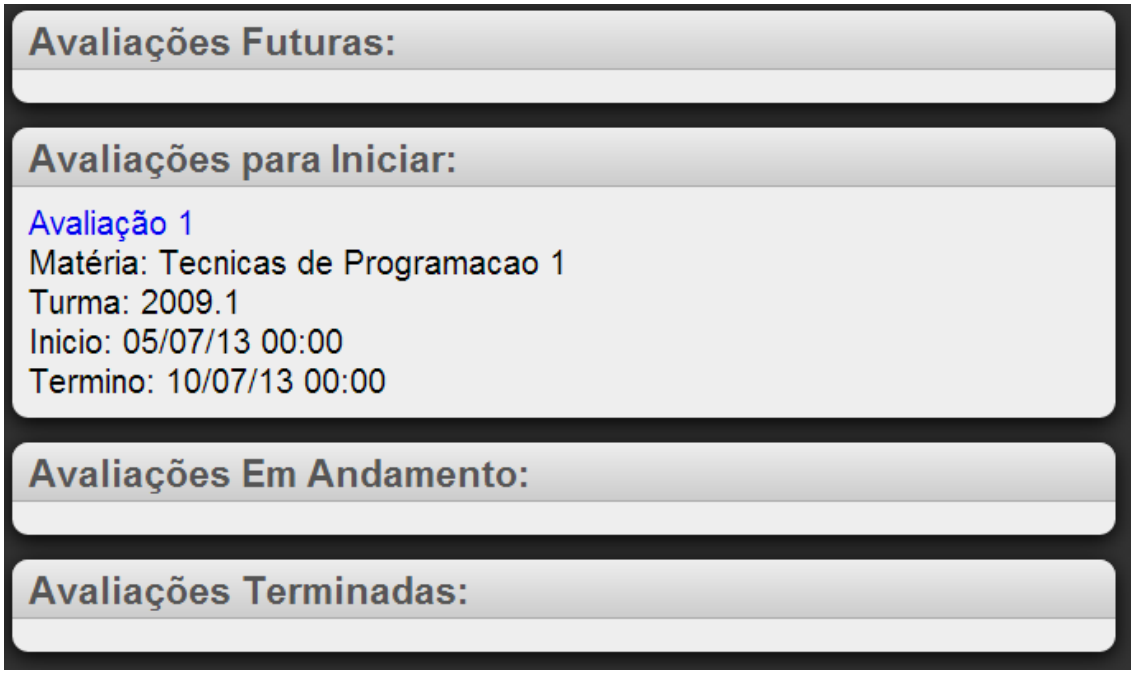

Figura 29 - Tela com listagem categorizada das avaliações

3.2.3.2 Resolução de Avaliação

Para realizar uma avaliação o aluno precisa escolher uma que esteja listada dentre as avaliações para iniciar ou em andamento. O diagrama de atividades apresentado na Figura 30 descreve o processo em que um aluno inicia uma avaliação.

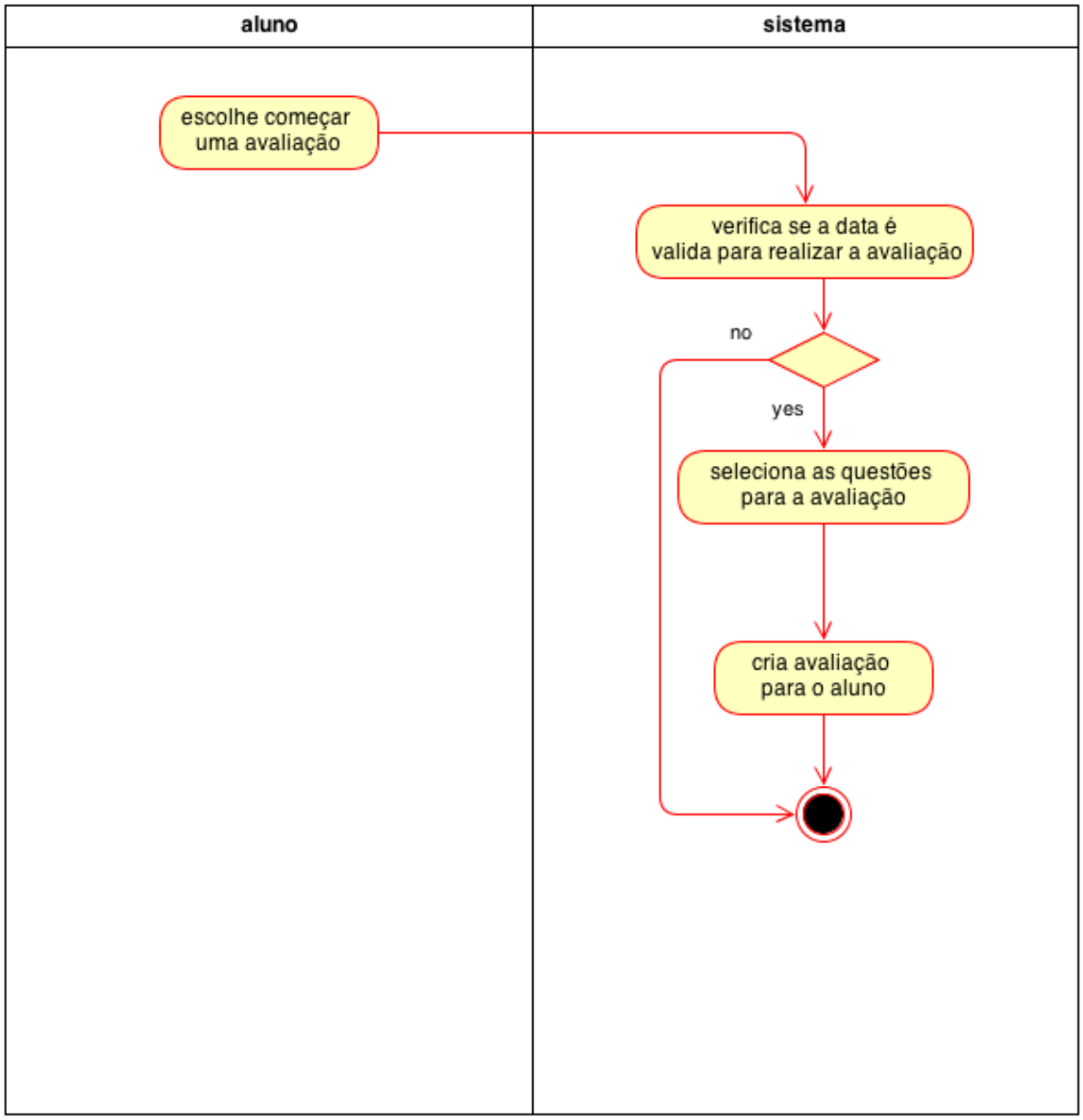

Figura 30 - Diagrama de atividades referente ao inicio da realização da avaliação pelo aluno

Para facilitar o entendimento, a etapa “selecionar as questões para a avaliação” é detalhada no diagrama de atividades na Figura 31. Vale ressaltar que esta etapa em particular permanece transparente para o usuário, correspondendo à uma lógica interna do sistema.

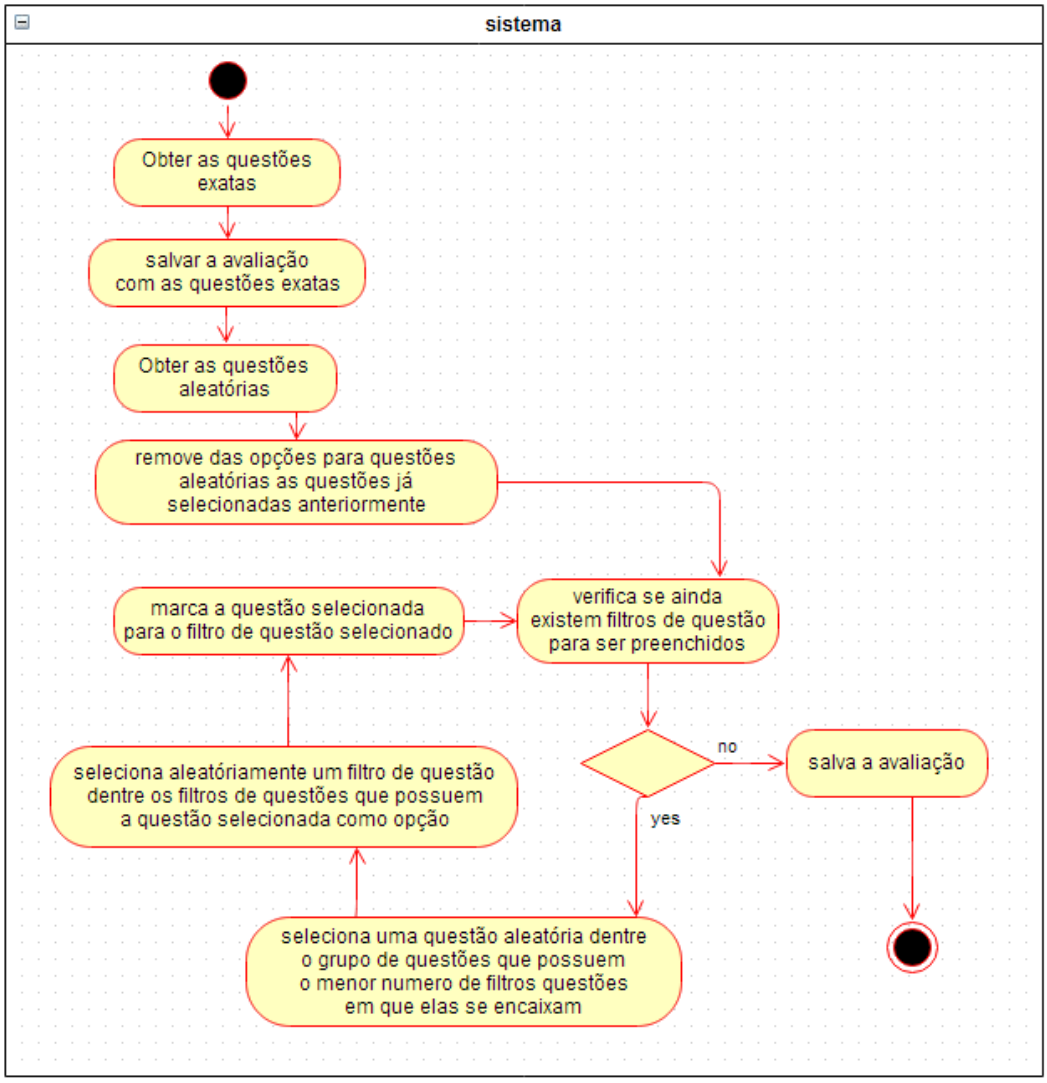

Figura 31 - Diagrama de atividades com seleção das questões para a avaliação

Uma vez iniciada uma avaliação o aluno verá uma tela semelhante a Figura 32.

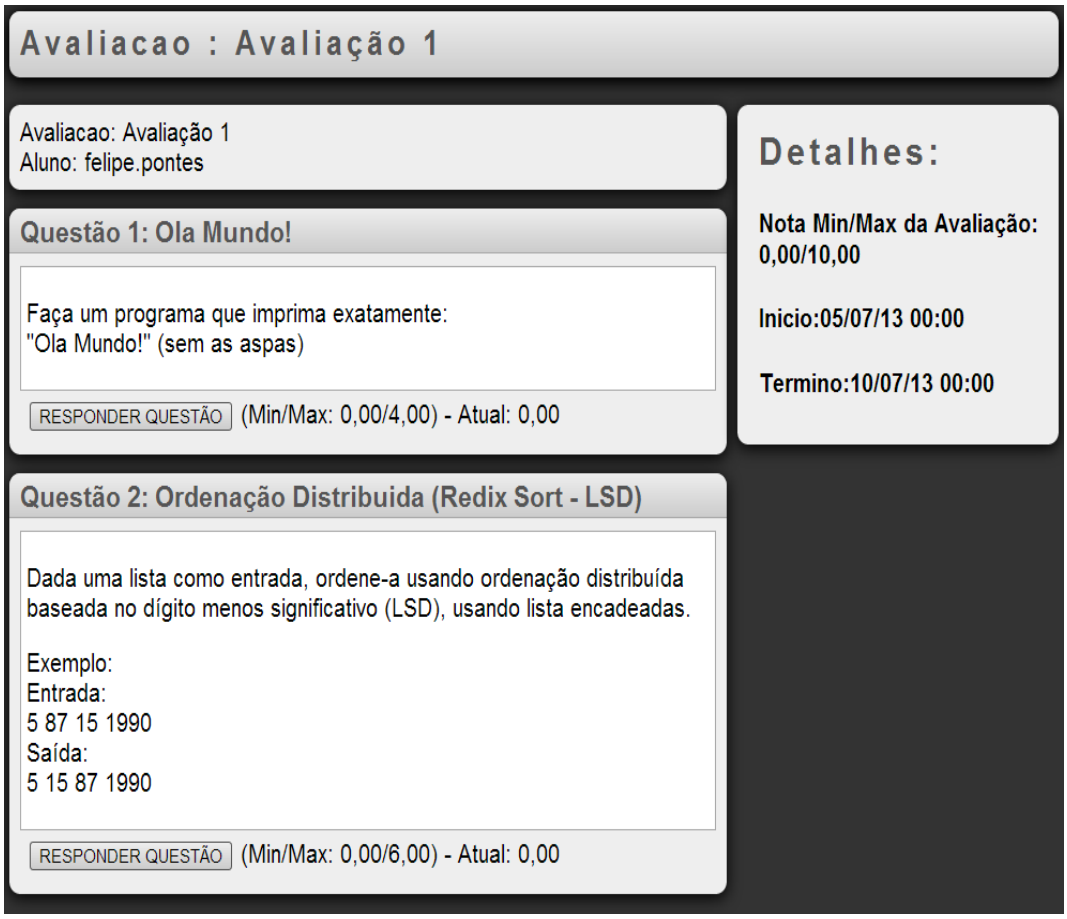

Figura 32 - Tela inicial da avaliação

Nessa página o aluno possuirá uma listagem de questões, mostrando seu enunciado, a pontuação mínima e máxima da questão e o quanto o aluno já obteve na mesma, junto com um botão para ir para a página de resolução da questão. Essa tela também possui uma caixa de detalhes informando as notas mínimas e máximas da avaliação, e a data/hora de inicio e término.

Caso o aluno acesse a página de uma avaliação já terminada, as questões possuirão um botão Gabarito que abrirá a página com o gabarito da questão, conforme Figura 33.

Figura 33 - Tela de acesso ao gabarito da questão para as avaliações já finalizadas

3.2.3.3 Página da Questão

Ao clicar no botão “RESPONDER QUESTÃO” na listagem de questões, o aluno será levado para a página onde poderá realizar a questão, um exemplo dessa tela é a Figura 34, cujos componentes serão descritos nos itens a seguir:

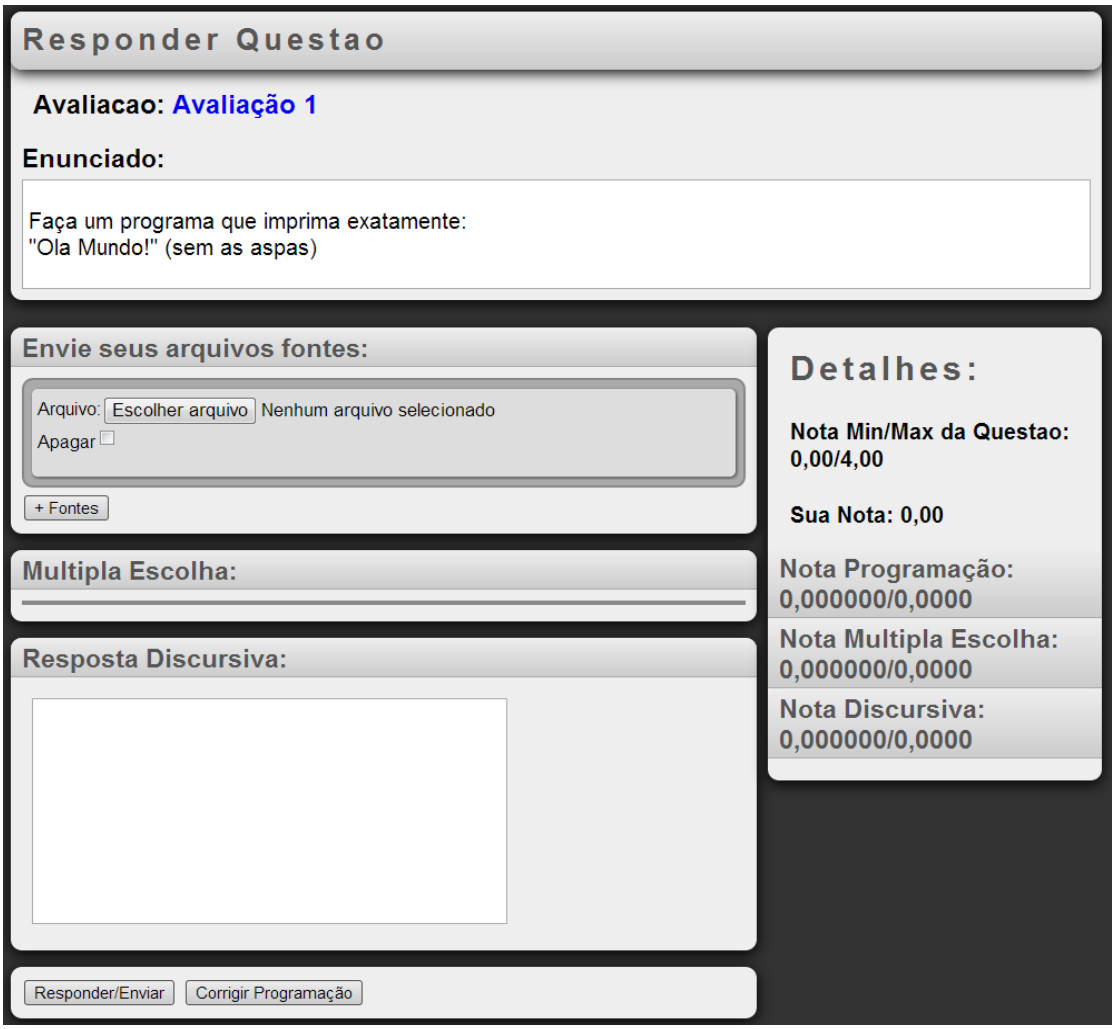

Figura 34 - Tela para responder uma questão selecionada

a) Cabeçalho

Essa área possui informações sobre a avaliação a qual a questão pertence, tais como título e enunciado (Figura 35).

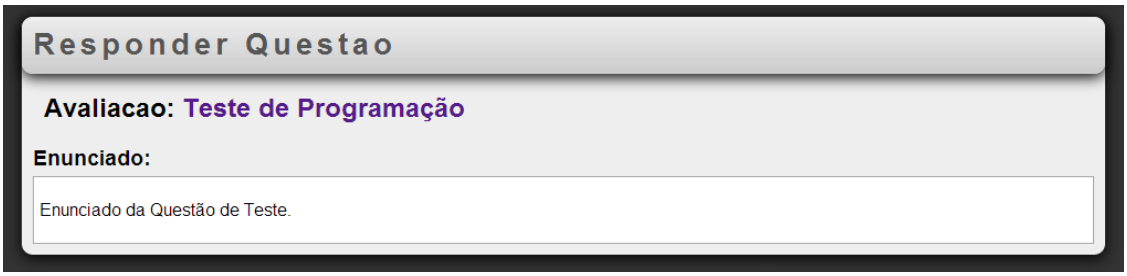

Figura 35 - Enunciado de uma questão.

b) Detalhes da Questão

Essa área possui informações sobre a nota mínima e máxima da questão, além da nota que o aluno já obteve, e mostra a pontuação dividida pelas partes que compõem a questão, tais como a programação, a múltipla escolha e a discursiva (Figura 36).

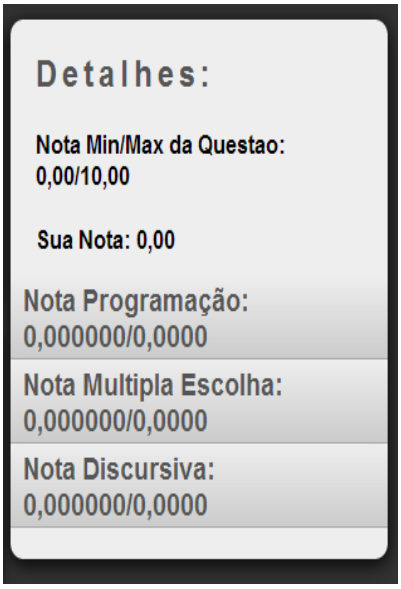

Figura 36 - Tela com detalhes da questão

c) Envio dos arquivos fontes

Essa é a área onde o aluno poderá enviar um ou mais arquivos fontes com a resposta da questão para serem compilados no servidor, aqui ele também pode apagar ou substituir arquivos já enviados (Figura 37).

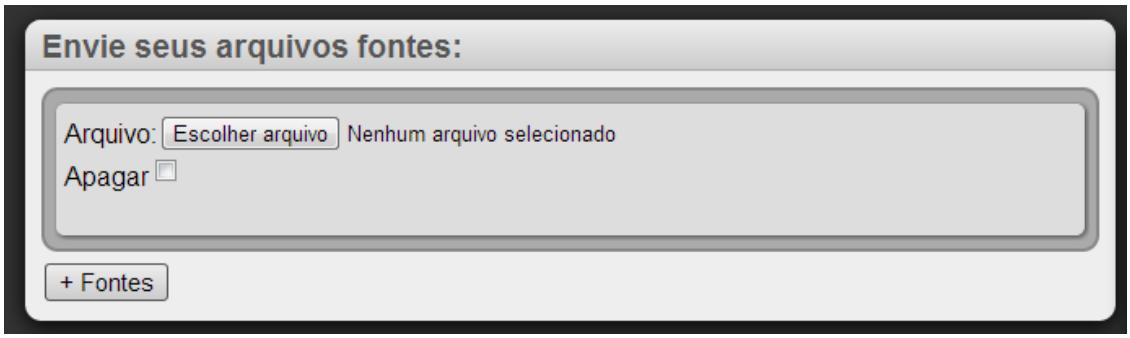

Figura 37- Tela de escolha do arquivo fonte para upload na resolução do exercício

Para adicionar um arquivo o aluno deve pressionar o botão "+ Fontes" que criará um nova caixa na listagem de arquivos, nessa caixa ele deve clicar na opção para escolher o arquivo do seu computador para enviar.

Para apagar um arquivo existente é necessário marcar a opção "Apagar" na caixa do arquivo.

d) Múltipla Escolha

Nesta são listadas as opções de múltipla escolha para o aluno marcar, caso exista múltipla escolha na questão (Figura 38).

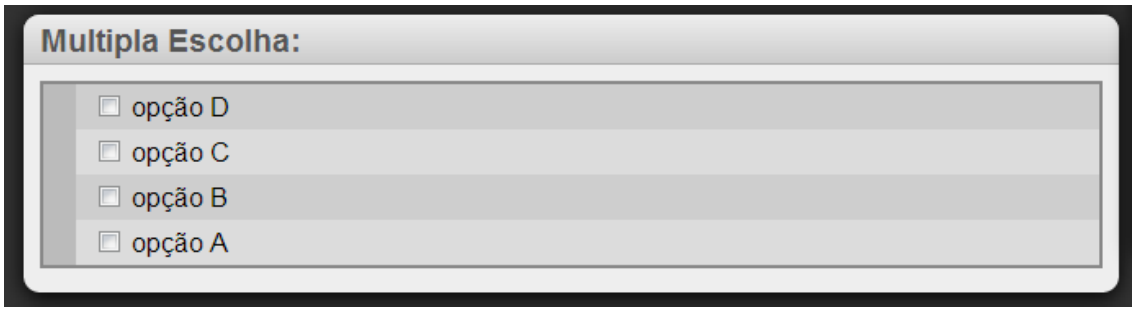

Figura 38 - Trecho de tela relativo a opção de múlitpla escolha para resolução

e) Resposta Discursiva

Essa área possui um campo de texto onde o aluno pode digitar a resposta discursiva da questão (Figura 39).

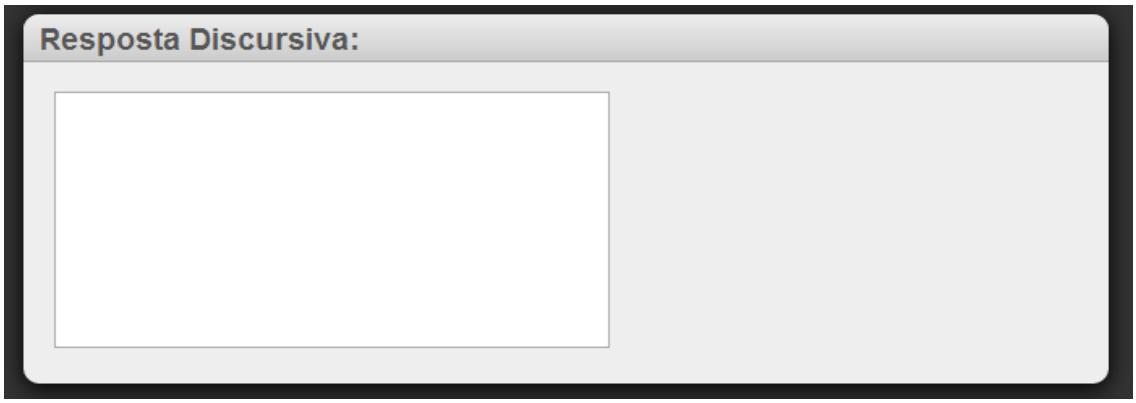

Figura 39 - Trecho de tela relativo a parte discursiva da questão

No final da página da questão existe um campo com dois botões, Responder/Enviar e Corrigir Programação (Figura 40).

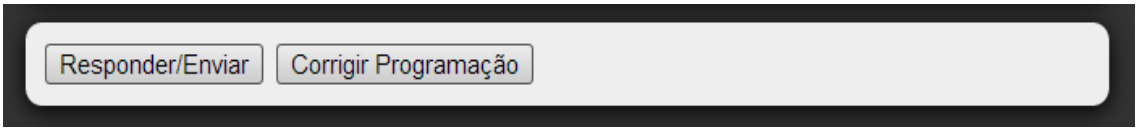

Figura 40 - Trecho de tela para corrigir e enviar questão

O botão Corrigir Programação irá realizar a correção da parte de programação da questão, e levará o aluno para uma tela de *feedback*, onde ele pode ver se o programa compilou e executou corretamente, ou se a saída gerada estava correta.

O botão Responder/Enviar irá salvar as alterações realizadas na questão, porém o aluno sempre poderá alterar as suas respostas até o fim do prazo da avaliação.

3.2.3.4 Pagina de Feedback do Corretor

Esta é a tela que será apresentada ao aluno quando ele escolher "Corrigir Programação" na página da questão, essa tela tem a informação de qual questão está sendo corrigida e o resultado da correção (Figura 41).

O resultado não é instantâneo, o aluno pode ter que esperar alguns segundos até o servidor retornar o resultado da correção, seja essa resposta de sucesso ou falha.

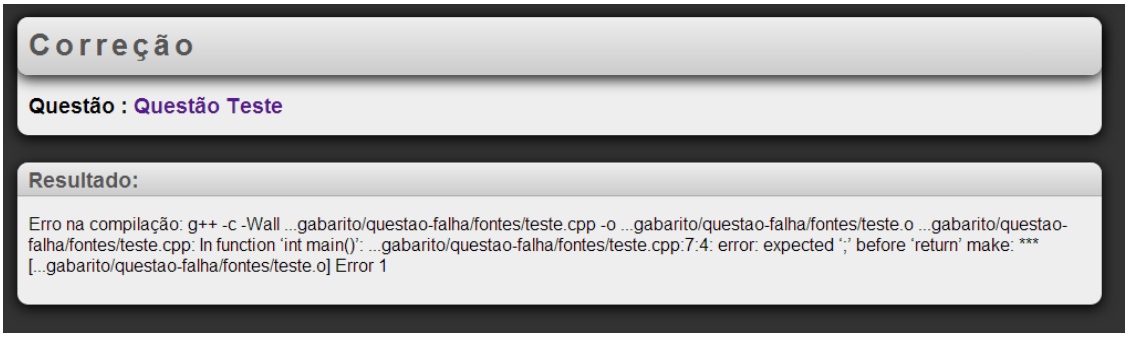

Figura 41 - Tela com exemplo de correção de uma questão

3.2.3.5 Pagina de Gabarito

A página de gabarito pode ser acessada clicando no botão gabarito que aparece na parte inferior das caixas de questões na tela de uma avaliação que já foi terminada (Figura 33).

Nesta página o aluno verá uma tela semelhante a Figura 42.

Figura 42 - Tela de gabarito da questão

Essa tela possui as informações de enunciado e nota mínima e máxima da questão, e mostra os códigos fonte do gabarito enviados pelo professor.

Neste mesmo ponto é possível consultar, após o término da prova, qual era a solução correta da questão.

4 TECNOLOGIAS E METODOLOGIAS USADAS NO DESENVOLVIMENTO

Ao longo de todo o processo de desenvolvimento do software foram selecionadas e combinadas uma grande variedade de metodologias e tecnologias. Em cada uma das etapas de trabalho foi priorizada a utilização de boas práticas e de ferramentas de software livre que são tendências de mercado.

Vale ressaltar que, devido ao grande dinamismo das soluções no segmento de software livre para desenvolvimento ágil e plataformas web, é contraprodutivo o uso de bibliografias tradicionais, pois o mercado editorial não tem como acompanhar a rápida evolução das ferramentas.

Serão discutidas nas seções seguintes as ferramentas utilizadas nas funções de gestão do projeto, controle de versionamento, utilitários de teste, aplicativos de gerência de dependências, ambiente virtual de Python, framework web com seus aplicativos adicionais, banco de dados, aplicações de depuração, broker de mensagens, bibliotecas Javascript e servidor web.

4.1 Sistema de Controle de Versionamento e de Gestão de Projeto de Software

Para viabilizar a atividade de elaboração do software com a participação de mais de um desenvolvedor atuando simultaneamente e para que fosse possível retornar a uma versão anterior caso houvesse algum erro, optou-se pela utilização da ferramenta GIT[6]. Este utilitário funciona como um controle de versões para os arquivos fonte gerados e possui uma série de recursos que facilitam o processo de desenvolvimento.

[6] http://git-scm.com/

A escolha desse sistema e não de outros como o SVN ou CVS teve como motivação algumas facilidades que o GIT oferece e que não estão presentes em uma ou em ambas das soluções concorrentes, como por exemplo:

- Facilidade em tratar o versionamento de pastas;
- Economia de espaço ao utilizar *"Branches"* (Vertentes);
- Simplicidade em renomear um arquivo;
- Rapidez; e
- Versionamento mesmo que offline.

Adicionalmente, diversas redes de desenvolvimento colaborativo tais como o GitHub, funcionam com base no sistema GIT, o que facilitaria a disseminação dos resultados da pesquisa numa etapa posterior do trabalho (CHACON, 2009).

Atuando de forma integrada com o GIT foi implementado o Trac[7], que também é uma ferramenta open source, e possui uma interface web para controle de alterações em projetos de desenvolvimento de software. Seu objetivo é ajudar o desenvolvedor a rastrear essas mudanças, acessar os registros que explicam o porquê de cada uma e qual o seu impacto no projeto como um todo .

Seus principais recursos são:

- instrumentos de Gerência de Projetos;
- sistema de Ticket e rastreamento de erros;
- linha do tempo das atividades recentes para o acompanhamento da evolução do projeto;
- integração com diversos sistemas de controle de versão como o Subversion, Git, Mercurial, Bazaar, Perforce e Darcs; e
- disponibilidade de uma ferramenta wiki para propiciar uma melhor documentação do sistema.

O software Trac tem seu uso bastante disseminado. Dentre os seus diversos usuários destaca-se, por exemplo, o Laboratório de Propulsão a Jato da NASA, que usa a ferramenta para controle de vários projetos.

[7] http://trac.edgewall.org

4.2 Ambiente Python

virtualenv

O sistema AMAO foi desenvolvido em Python e utiliza a ferramenta virtualenv[8], que possibilita a criação de ambientes isolados de Python, permitindo resolver o problema básico de controlar as dependências, versões e permissões dos módulos do projeto sem que isso possa interferir em qualquer outra aplicação ou serviço em Python na mesma máquina.

A utilidade do emprego dessa ferramenta pode ser melhor entendida considerando-se o seguinte cenário, supondo que existe uma aplicação "A" que precisa da versão 1 de um módulo fictício LibFoo, mas há uma outra aplicação "B" na mesma máquina que requer a versão 2 desse mesmo módulo.

Se as dependências de "A" e "B" (ou seja, as duas versões do módulo LibFoo) forem instaladas no local padrão da máquina, como por exemplo: **"/usr/lib/python2.7/site-packages/"**, então uma das duas aplicações(A ou B) deixaria de funcionar.

Este inconveniente é evitado com o emprego da ferramenta virtualenv, possibilitando assim, melhor controle dos pacotes Python do sistema. Já os módulos Python que são genéricos, ou seja, que não estão diretamente ligados ao AMAO, mas que são essenciais para a máquina, foram instalados no ambiente global.

O diagrama da Figura 43 ilustra como se dá essa interação.

[8] http://www.virtualenv.org

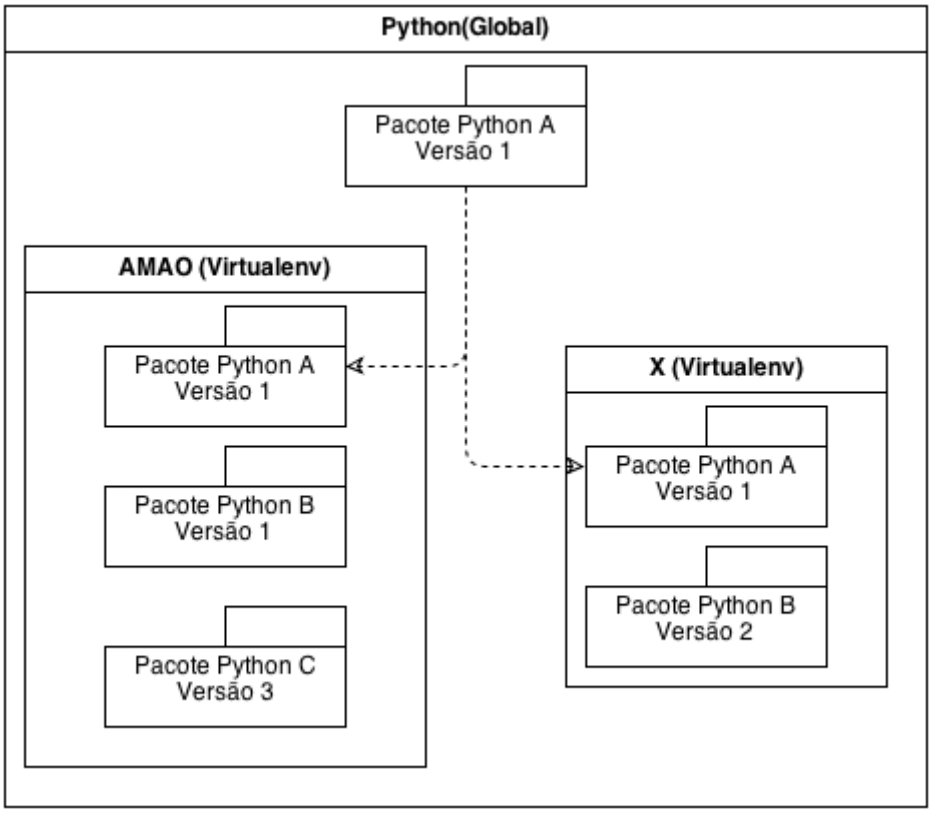

Figura 43 - Exemplo de gerenciamento de dependência empregando o Virtualenv

Nesse diagrama o Pacote Python A Versão 1 está instalado no ambiente global, logo disponível para todos os ambientes virtuais (AMAO e X).

No caso do ambiente virtual AMAO, o Pacote Python B é da Versão 1, enquanto o mesmo pacote no ambiente virtual X é da Versão 2.

Além disso o ambiente AMAO tem um pacote Python C que não existe no ambiente X.

Com isso é possível ter dois ambientes Python completamente distintos rodando na mesma máquina.

virtualenvwrapper

Essa ferramenta permite adicionar funcionalidades ao virtualenv[9]. Algumas dessas melhorias são:

- uma melhor navegação entre ambientes Python;
- fácil ativação de um ambiente; e

[9] http://virtualenvwrapper.readthedocs.org

- utilização de alguns hooks(scripts que são executados automaticamente em determinadas ações) que agilizam o desenvolvimento.

Mais informações sobre o uso desta ferramenta podem ser vistas em (PONTES, 2011).

PIP

Essa ferramenta possibilita um fácil gerenciamento das dependências em Python do projeto, instalando facilmente o que é necessário, no lugar de ter que instalar os pacotes Python manualmente, além de tratar as dependências desses mesmos pacotes.

Este utilitário também possuí a vantagem de poder gravar em um arquivo de texto todos os módulos que estão atualmente instalados. Esse arquivo de texto é normalmente chamado de **"requirements.txt"**.

Uma vez tendo em mãos o arquivo requirements.txt, pode-se utilizá-lo pelo PIP para instalar todas as dependências nele listadas com um único comando.

Utilizado em conjunto com o virtualenv, pode-se recriar um ambiente virtual Python com facilidade.

4.3 Django e Suas Ferramentas

Django[10] é uma framework web de alto nível em Python, que encoraja o desenvolvimento ágil e limpo (KAPLAN-MOSS; HOLOVATY, 2007).

Esta ferramenta foi projetado para lidar com dois desafios: os prazos apertados típicos do trabalho de redação jornalística no qual seria originalmente usado e os requisitos rígidos dos experientes desenvolvedores web que o criaram. Ela permite a construção rápida de aplicações web de alto desempenho e elegância. Atualmente a ferramenta já atingiu um bom nível de maturidade e vem sendo utilizada em sites de grandes empresas, tais como Instagram(recentemente comprada pelo Facebook por um bilhão de dolares), Mozzila, Disgus, Globo.com, The Washington Post, The Guardian, The New York Times, conforme detalhado em (LUZ, 2011), (REES, 2011) e (WILLIAMS, 2012).

Django se concentra no máximo de automatização possível e adere ao princípio DRY - *"Don't Repeat Yourself"* (não se repita) (THOMAS; HUNT, 1999).

[10] https://www.djangoproject.com

No desenvolvimento do AMAO foi usada a versão estável 1.3 do Django, que era a última disponível no momento e foram aproveitadas diversas funcionalidades já existentes na framework tais como:

- Módulo de Autenticação: **django.contrib.auth;**
- Módulo de Sessões: **django.contrib.sessions**; e
- Módulo de Administração: **django.contrib.admin**.

Isso permitiu que em pouco tempo houvesse um protótipo com algumas das funcionalidades básicas, como os CRUD (operações básicas dos modelos do sistema), utilizando a interface de administração do Django, além de um sistema de autenticação, como mostra a Figura 44. Ao longo do desenvolvimento esta interface foi substituída por uma versão mais elaborada e de acordo com a identidade visual do site.

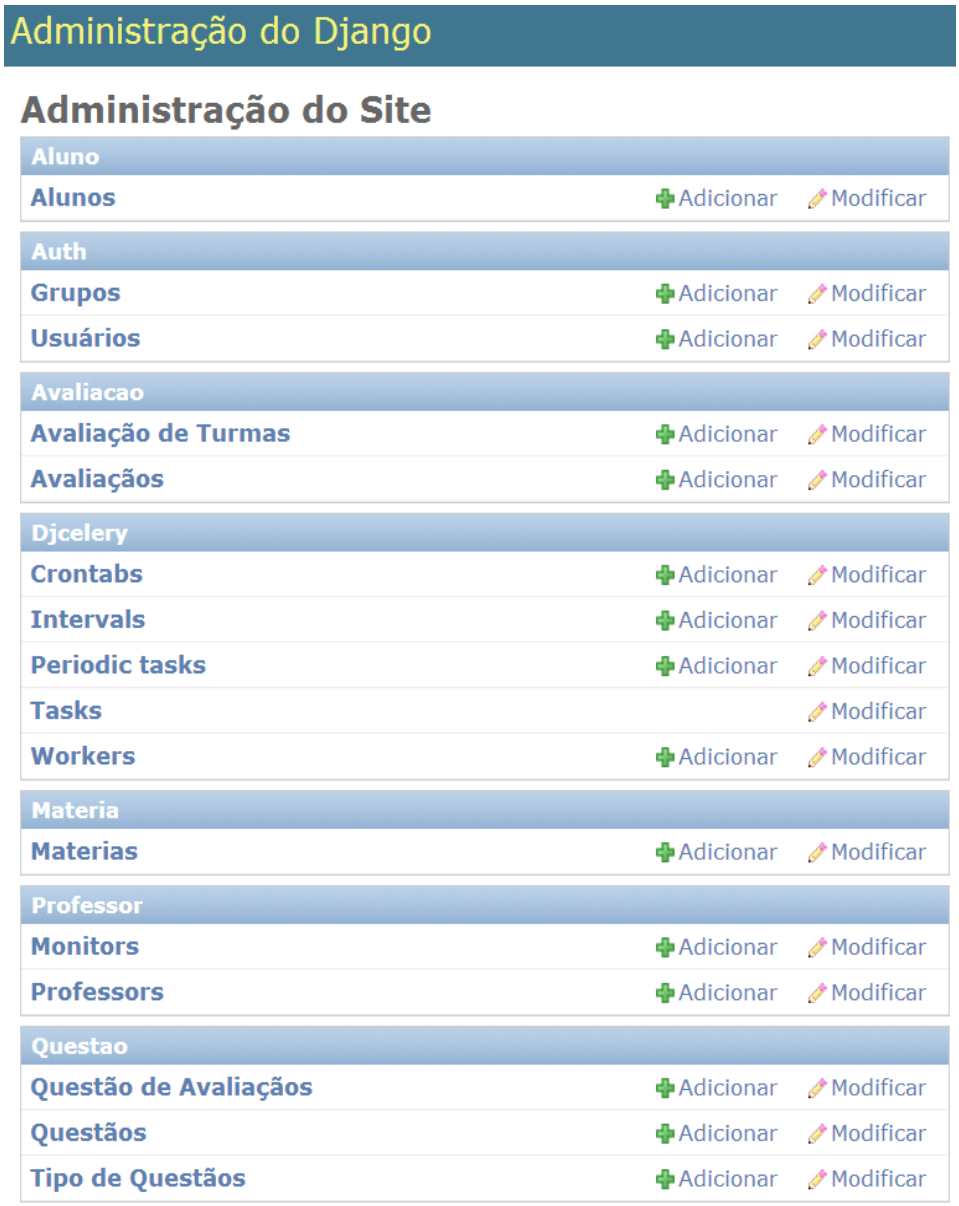

Figura 44- Versão preliminar da tela de administração do site que empregava a interface de administração Django

Werkzeug

Essa é uma biblioteca WSGI[11] (Web Server Gateway Interface) que é amplamente usada.

Um WSGI define uma interface simples e universal entre servidores web e aplicações web ou frameworks para a linguagem de programação Python, além de ser um padrão Python definido pela PEP 333 [12] conforme pode ser visto em (EDY, 2011).

[11] http://wsgi.readthedocs.or

[12] PEP (Python Enhancement Proposals) - conjunto de propostas de melhorias da linguagem Python e boas práticas sugeridas pela comunidade .

Uma das vantagens que esta biblioteca possui é a execução em modo Debug, que torna a depuração de erros possível dentro do navegador que está acessando a aplicação. A Figura 45 é um exemplo da utilização deste recurso.

Figura 45 - Exemplo de depuração empregando Werkzeug

Django-extensions

Django-extensions[13] é uma biblioteca que consiste de um conjunto de diversas aplicações customizadas que facilitam o desenvolvimento de um sistema na framework Django, dentre as mais importantes pode-se destacar:

- **runscript**: Roda um script Python com contexto e configurações do Django;

[13] http://pythonhosted.org/django-extensions/

- **validade-templates**: Verifica se os templates HTML possuiem erros de sintaxe ou de compilação; e
- **graph-models**: Permite criar um diagrama simplificado da modelagem do sistema.

Django-Annoying

Uma aplicação Django que traz diversos atalhos úteis para o desenvolvimento de sistemas web, reduzindo assim a quantidade de código escrito e tendo um melhor reaproveitamento do que já existe.

Alguns desses atalhos são:

- Decorador *render_to*;
- Decorador *signals*; e
- Função *get_object_or_None.*

django-mptt

Ferramentas que permitem a implementação de uma MPTT[14](Modified Pre-order Traversal Tree), que consiste em uma técnica de armazenar herança em um Banco de Dados.

Essa técnica foi utilizada na modelagem dos **Tipos de Questão**, em que cada tipo pode ser filho de outro tipo, criando assim uma herança.

Uma análise detalhada e didática do tema pode ser encontrada em (TULDER, 2003).

django-debug-toolbar

O django-debug-toolbar[15] é uma aplicação que habilita uma aba de ferramentas em todas as páginas do sistema como pode ser visualizado na Figura 46.

Nessa aba é possível acessar algumas informações úteis para a depuração de uma tela, algumas dessas informaçõe são:

- quais templates (HTMLs) foram carregados em uma página;
- Quais variáveis Python foram carregadas no contexto;

[14] http://django-mptt.github.io/django-mptt/
[15] https://github.com/django-debug-toolbar/django-debug-toolbar

- Sinais que foram enviados; e
- SQLs executadas e o tempo de execução de cada uma.

Essa aba só é carregada em um ambiente de desenvolvimento, portanto não aparece na aplicação em produção.

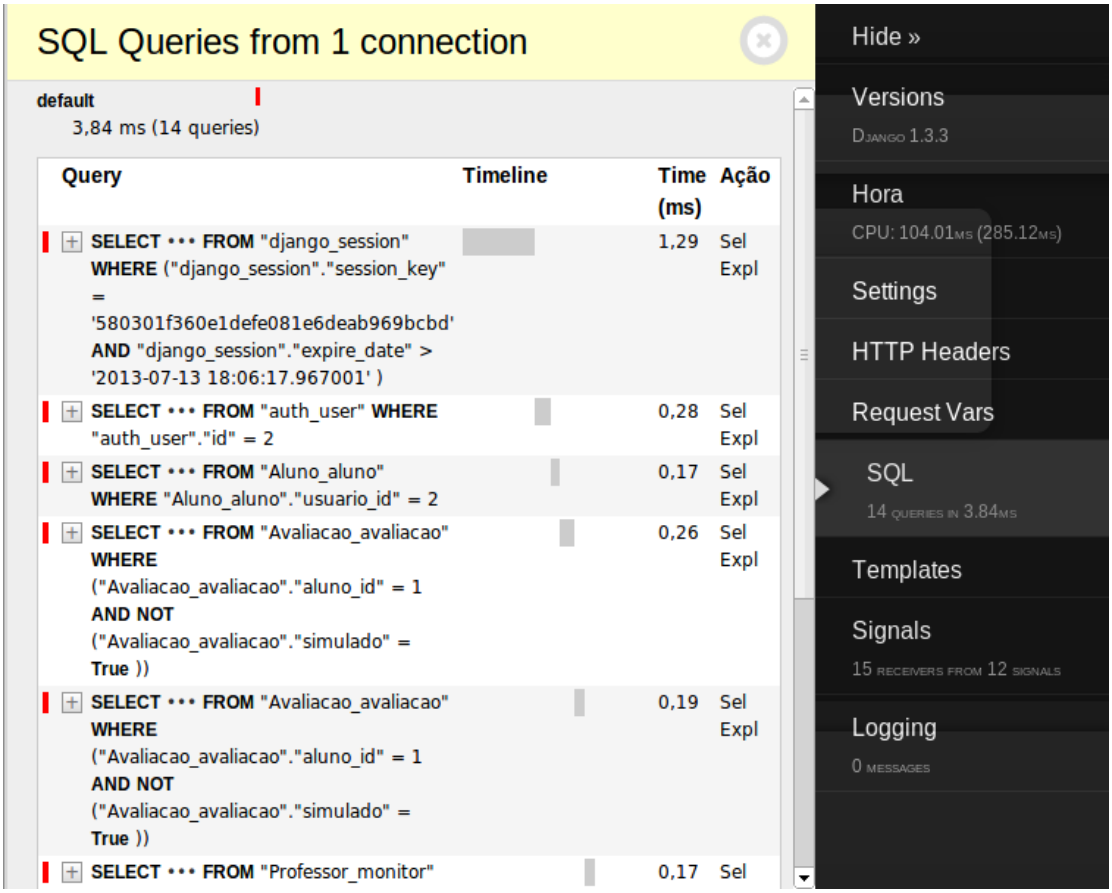

Figura 46 - Tela do django-debug-toolbar

4.4 Processos Assíncronos

Celery

Essa ferramenta é um distribuidor assíncrono de fila de processos/tarefas baseado na transmissão distribuída de mensagens. É normalmente utilizado em operações em tempo-real, porém permite a utilização de Agendamento(Scheduling). E como é feito em Python sua integração com o Django é simples.

No sistema AMAO, é responsável por enfileirar os processos de correção automática e enviar estes ao Broker de Mensagens(RabbitMQ).

Também é utilizada para o Agendamento dos processos de *Iniciar Avaliações* e *Terminar Avaliações* de acordo com as datas de início e termino destas no Banco de Dados.

Esta ferramenta possuí uma extensa documentação na integração Celery/RabbitMQ.

django-celery:

Para integrar com o Django, basta instalar esta aplicação, que permite acessar e configurar os processos/tarefas do Celery de dentro do sistema, tendo inclusive uma interface básica dentro da API de administração do Django.

RabbitMQ[16]

Este sistema é um Broker de Mensagens (Message Broker), escrito em Erlang, que implementa o padrão AMQP (Advanced Message Queuing Protocol).

Além de ser uma solução Open Source, seu servidor é construído em cima de uma framework tolerante a falhas e que permite a utilização de Clustering.

Este é responsável pela execução de modo assíncrono e distribuído dos processos de correção do AMAO, e foi escolhido no lugar de outros softwares do mesmo gênero, por ter uma configuração simples, rápida e por ser o Broker de uso recomendado pelo Celery. Um estudo mais aprofundado desta ferramenta pode ser visto em (RICHARDSON et al ,2013).

4.5 Banco de Dados

Como a framework Django utiliza um ORM(Mapeamento Objeto-Relacional) robusto, todo o acesso ao banco é feito através de classes Python, tornando o sistema independente de qual o banco de dados utilizado.

Assim para trocar qual o banco que está sendo utilizado pela framework, basta alterar a configuração para o driver do banco escolhido.

Como banco de dados para ser usado na etapa de produção, foi selecionado o PostgreSQL[17], que é um software robusto, confiável e de alta conformidade com os padrões. Outra vantagem dessa ferramenta se deve a existência de vasta documentação relativa a integração com a framework.

[16] http://www.rabbitmq.com/documentation.html
[17] http://www.postgresql.org/

No que tange ao ambiente de desenvolvimento, foi escolhido o SQLite3[18] , que é um banco de dados transacional, simples e auto-contido.

South

South[19] é uma ferramenta para fornecer migrações de banco de dados de forma simples para aplicações Django, facilitando a manutenção da sincronia entre as tabelas do banco de dados e os modelos do sistema. Uma característica importante é que o funcionamento do South é independente da base de dados escolhida.

4.6 Front-End

HTML5-Boilerplate

É um template open source para websites com suporte para HTML5 e compatível com vários navegadores, incluindo navegadores mobile.

HTML5-Boilerplate[20] inclui uma base de css, jquery, Google Analytics dentre outras estruturas prontas para acelerar o desenvolvimento front-end.

jQuery[21]

O JQuery é uma biblioteca Javascript compatível com vários navegadores para facilitar o desenvolvimento de código nesta linguagem, simplificando tarefas como manipulação do documento HTML, tratamento de eventos, animação, AJAX, dentre outras funcionalidades.

> **jQuery Ui[22]:** É um conjunto de componentes Javascript prontos, construídos tendo como base a biblioteca jQuery. São utilizados para criar interações, interface de usuário, efeitos, widgets e temas.

[18] http://www.sqlite.org/
[19] http://south.readthedocs.org/en/latest/about.html
[20] http://html5boilerplate.com/
[21] http://jquery.com/
[22] http://jqueryui.com/

4.7 Testes

Para suportar a atividade de teste do sistema, optou-se pelo uso de diversas aplicações ao invés do simples executor de testes do Django, pois este além de não ser muito rápido, é muito básico. A seguir são enumeradas as aplicações utilizadas nos testes:

Nose

No lugar do executor de testes padrão do Django, foi selecionada outra ferramenta chamada Nose[23]. Esta ferramenta trabalha estendendo o unittest[24] que é uma biblioteca padrão de testes do Python o que facilita a realização dos testes.

Além dessa biblioteca, foi utilizada uma série de plugins e aplicativos que são executados por cima da mesma, tais como:

django-nose[25]:

Possibilita rodar os testes do Nose no escopo da framework Django, aproveitando assim as configurações do projeto e outras facilidades desta.

nose-exclude[26]:

Um plugin do Nose que simplifica a exclusão de diretórios durante a execução de testes. Isto é, para que determinadas pastas sejam ignoradas em testes específicos.

rednose[27]:

Deixa as saídas dos testes coloridas, melhorando desta forma a identificação dos erros.

[23] https://nose.readthedocs.org/en/latest/
[24] http://docs.python.org/2/library/unittest.html
[25] https://pypi.python.org/pypi/django-nose
[26] https://pypi.python.org/pypi/nose-exclude
[27] https://pypi.python.org/pypi/rednose/

model-mommy[28]

Foi utilizado para tornar mais prática a criação de objetos Mocks, que são objetos falsos que simulam um comportamento esperado e controlado nos testes unitários(FREEMAN; PRYCE, 2010) .

Esta ferramenta possibilita a criação com um comando desses objetos e facilita o desenvolvimento dos testes.

django-test-utils[29]

Essa ferramenta facilita a criação de testes para o Django, pois tem comandos que para uma dada aplicação, a ferramenta cria testes unitários que devem ser utilizados como ponto de partida.

Isso ajuda a evitar que algum ponto de uma aplicação não esteja sendo coberto por testes.

4.8 Servidor Web

Nginx

Nginx[30] é um servidor HTTP e proxy reverso, bem como um servidor proxy de email IMAP/POP3.

O Nginx é um servidor web rápido, leve, e com varias possibilidades de configuração para melhor performance, ele lida com requisições Web através de uma arquitetura baseada em eventos assíncronos, que permite uma maior escalabilidade e melhor uso de memória.

4.9 SandBox

Safeexec

Para garantir que os programas executados pelo corretor automático não impactassem no resto do sistema, seja em questão de segurança ou de performance (no caso de vazamento de memória), foi utilizado esse programa como um mediador.

[28] https://github.com/vandersonmota/model_mommy
[29] http://django-test-utils.readthedocs.org/en/latest/
[30] http://wiki.nginx.org/Main

Esse programa escrito em C é uma sand-box (um ambiente controlado e restringido) leve para executar de forma segura e controlada programas advindos dos usuários.

A versão escolhida para ser utilizada pelo AMAO é uma vertente do safeexec[31] presente no sistema Mooshak[32], porém com a funcionalidade de chroot, tendo assim mais uma forma de controlar o ambiente em que será executado o código do aluno.

A relação entre as ferramentas Django, Celery, RabbitMQ e Safeexec pode ser vista na Figura 47, em que para um processo de correção é passado do Django para o Celery, que por sua vez envia o processo ao RabbitMQ Server. Esse último por fim envia o processo de correção a um cluster do Rabbit MQ que irá executar o programa de um aluno através do Safeexec, aproveitando seu ambiente seguro e restrito.

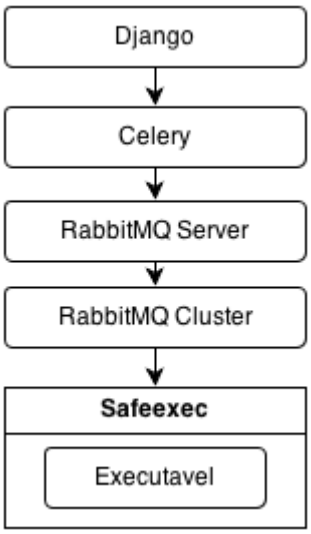

Figura 47 - Relação entre as ferramentas utilizadas

[31] https://github.com/cemc/safeexec
[32] http://mooshak.dcc.fc.up.pt/

5 INSTALAÇÃO DO SISTEMA

As etapas que compõem o processo de instalação do sistema estão divididas em instalação de desenvolvimento e instalação de produção por existirem diferenças no processo de cada uma, e as respectivas etapas estão listadas a seguir:

Instalação de Desenvolvimento

- Instalar o GIT
- Configurar a Chave RSA
- Instalar o PIP
- Instalar o Ambiente Virtual VirtualEnv
- Instalar o Ambiente Virtual VirtualEnvWrapper
- Configurar o Ambiente Virtual
- Criar e Ativar o Ambiente Virtual
- Instalar as Dependências
- Configurar o SafeExec
- Instalar o Django-Celery

Instalação de Produção

- Criar usuário AMAO no sistema
- Configurar Chave RSA
- Habilitar Login do Root
- Habilitar Serviço SSH

- Instalar o GIT
- Instalar e configurar o Nginx
- Instalar e configurar o Banco de Dados PostgreSQL
- Obter os arquivos fontes do projeto
- Instalar o PIP
- Instalar o Ambiente Virtual VirtualEnv
- Instalar o Ambiente Virtual VirtualEnvWrapper
- Configurar o Ambiente Virtual
- Criar e Ativar o Ambiente Virtual
- Instalar as Dependências
- Configurar o SafeExec
- Ativar o Ambiente de Produção
- Arrumar as pastas Media e Static
- Carregar o Django
- Sincronizar o Banco de Dados
- Instalar o Django-Celery
- Executar o Celery

A descrição completa dos procedimentos de instalação tanto do ambiente de desenvolvimento quanto do ambiente de produção está detalhada nos Anexos 1 e 2 respectivamente.

6 ORGANIZAÇÃO DO SISTEMA

6.1 Representação do Banco de Dados

Aqui serão mostradas a modelagem do banco referente a cada uma das aplicações que compõem o AMAO, deixando de fora outras modelagens ligadas a framework Django e outras ferramentas usadas no sistema (a modelagem completa do banco pode ser vista no Anexo 3).

6.1.1 Professor, Aluno, Matéria e Turma

Como estes modelos são simples, e diretamente relacionados, estes foram postos em um único diagrama (Figura 48):

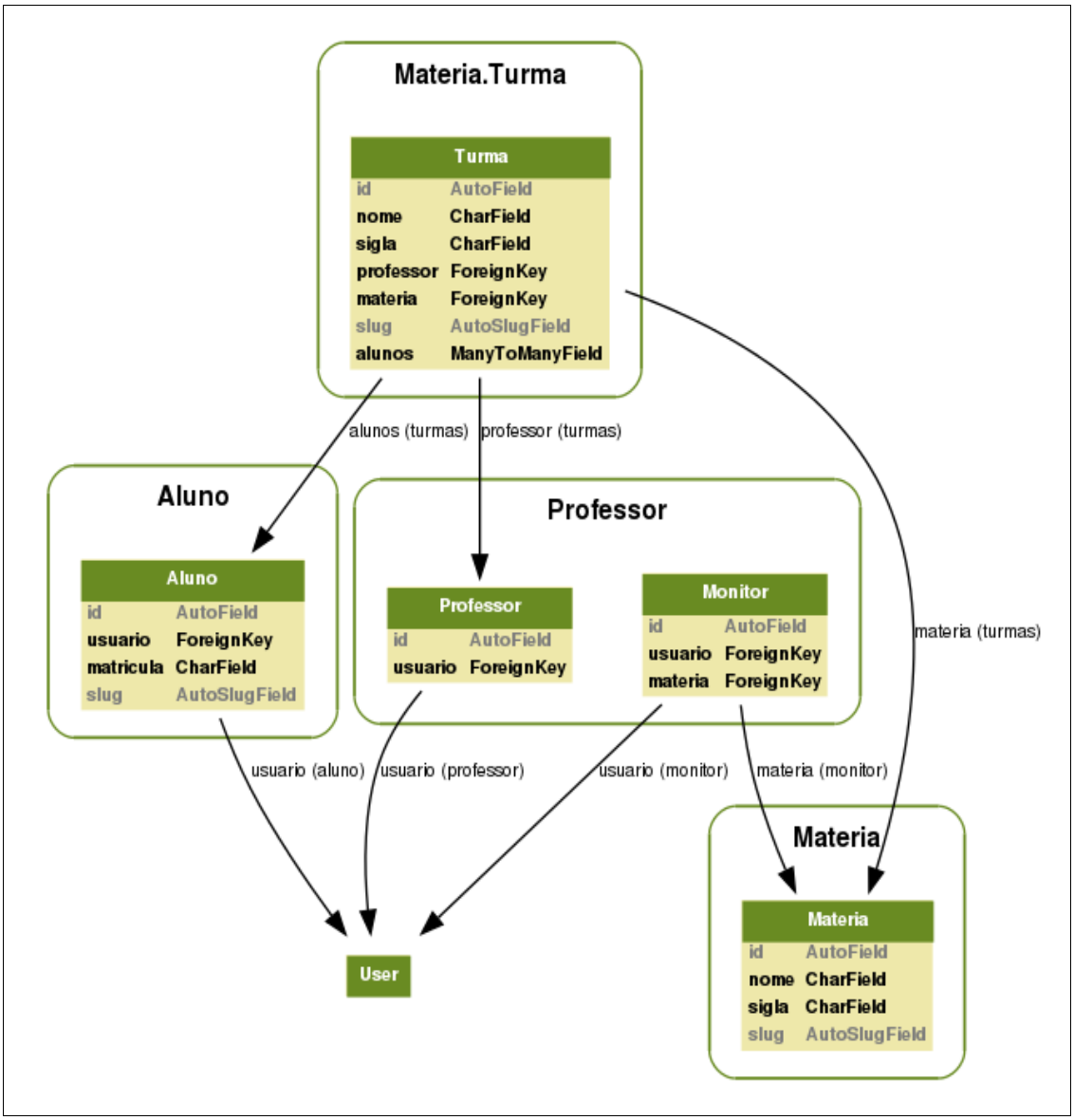

Figura 48 - Representação dos modelos Professor, Aluno, Matéria e Turma no banco de dados

6.1.2 Avaliação

A seguir, pode ser visto na Figura 49 a modelagem da aplicação de Avaliação, que possuí os modelos de Avaliação (avaliação de um aluno), Simulado (Não utilizado atualmente mas já criado para ser utilizado futuramente) e Template de Avaliação (Avaliação de uma Turma):

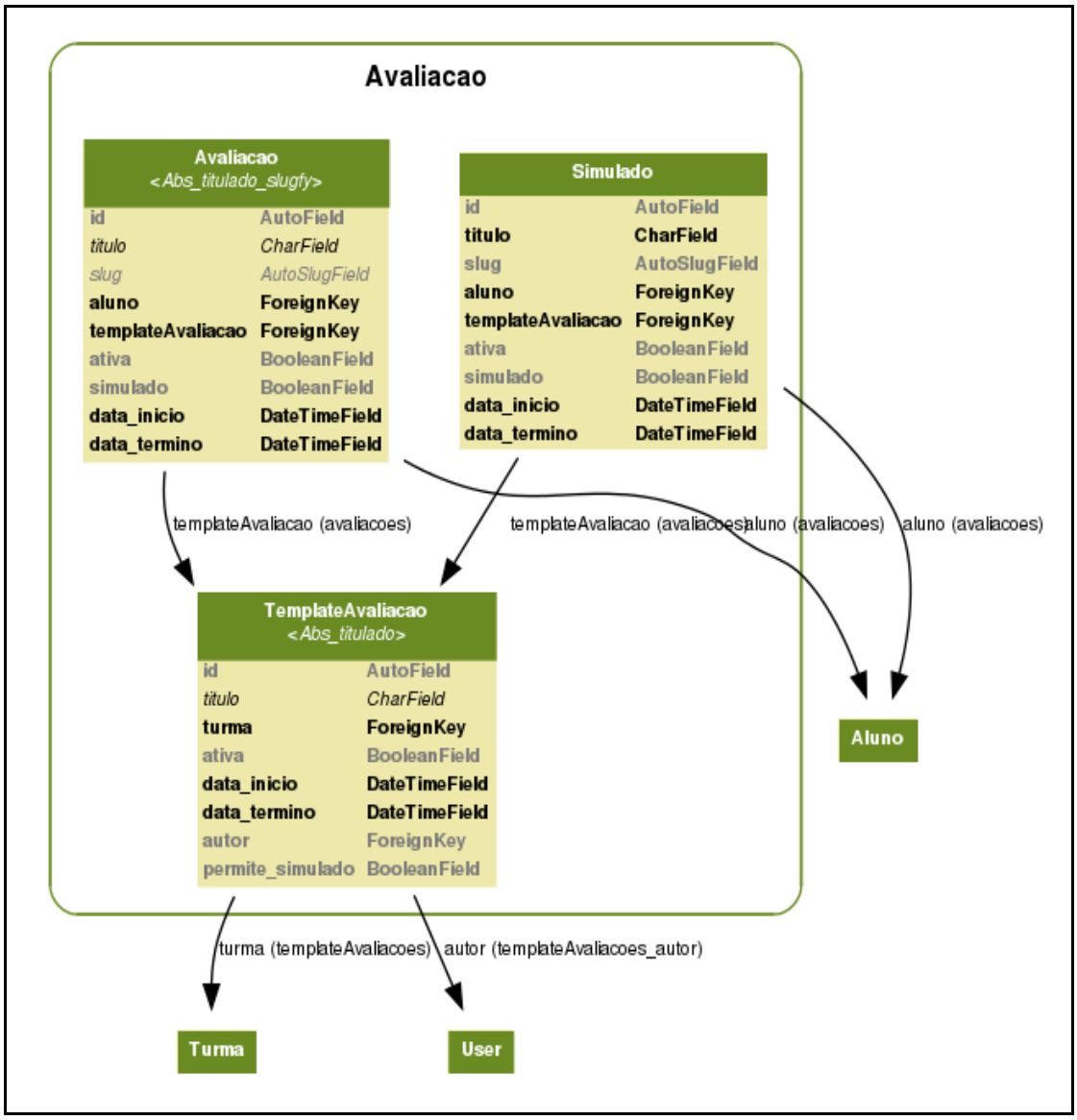

Figura 49 - Representação dos modelos Avaliação, Simulado, Template, Aluno, Turma e User

6.1.3 Questão e Corretor

Neste tópico são apresentadas as modelagens de banco das aplicações Questão e Corretor, que foram postas em conjunto já que estão diretamente ligadas.

O diagrama(Figura 50) possui diversos modelos, são eles:

- **Entrada Gabarito** Um arquivo de entrada para ser usado como gabarito em uma Questão;
- **Filtro Questão** Utilizado na criação de uma avaliação para determinar que questão será selecionada na hora de criar a avaliação de um aluno;
- **Fonte** Arquivo fonte de uma questão de um aluno;
- **Fonte Gabarito** Arquivo fonte de um gabarito de uma questão;

- **Opção Múltipla Escolha** Representa uma opção múltipla escolha em uma questão;
- **Questão** Representa uma questão do Banco de Questões;
- **Questão de Avaliação** Uma questão de um aluno em uma avaliação;
- **Retorno Correção** Os dados do retorno de uma correção, seja de uma Questão de Avaliação ou de uma Questão e
- **Tipo de Questão** É o que define em quais critérios uma Questão se enquadra;

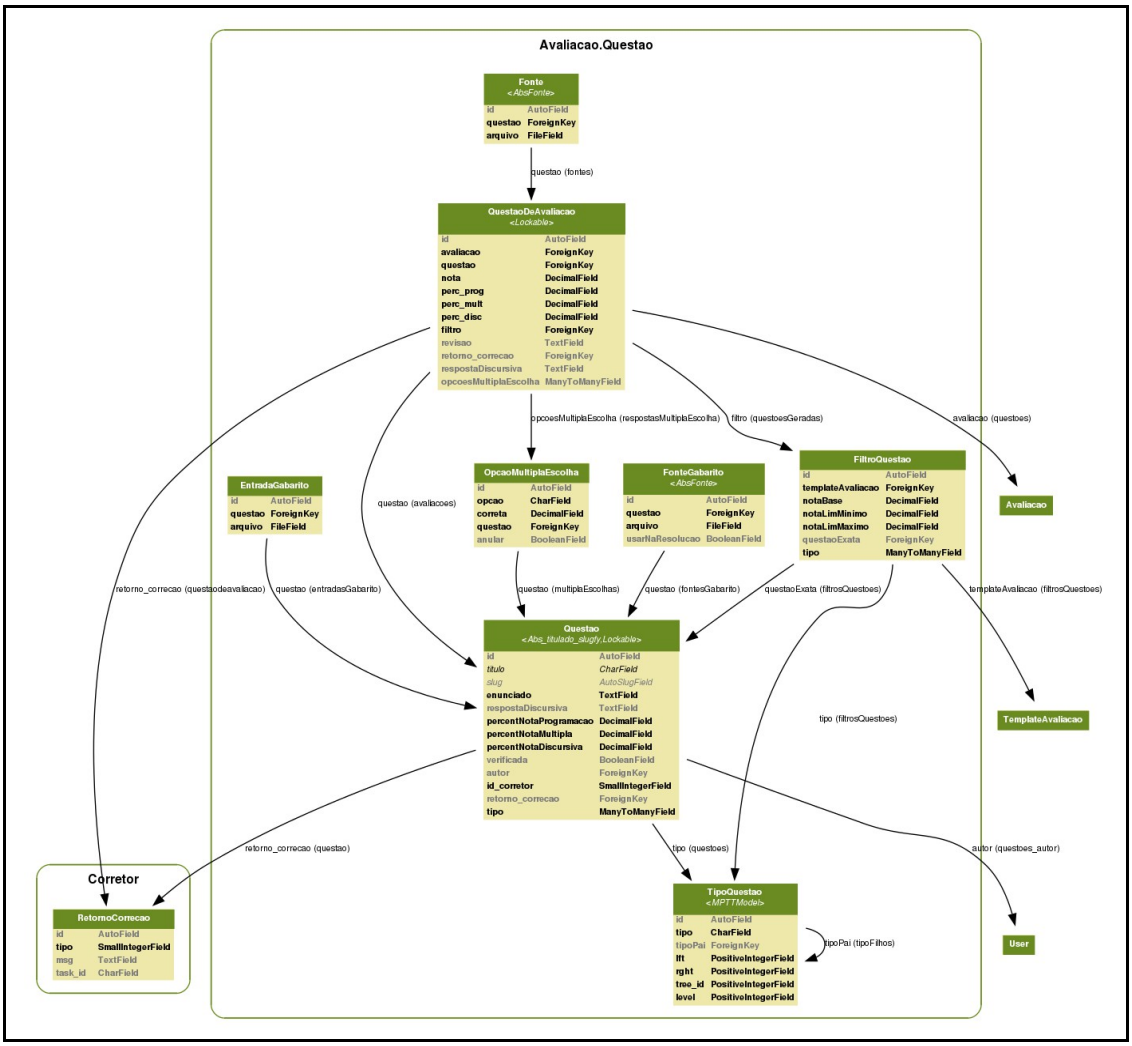

Figura 50- Representação da modelagem do banco das aplicações Questão e Corretor

6.2 Estrutura do Sistema

Esta seção tem o intuito de explicar a estruturação do AMAO, como são organizadas as aplicações Django, os módulos Python usados e como interagem os diversos serviços que compõem o sistema.

6.2.1 Estrutura do Django

O Django utiliza uma estrutura chamada MVT(Model, View e Template) que é similar a MVC(Model, View e Controller)(YATES et al, 2006) padrão.

A forma como uma aplicação Django é estruturada pode ser melhor analisada na Figura 51.

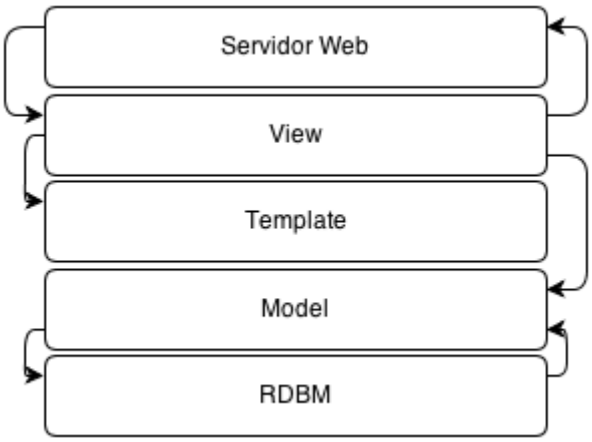

Figura 51 - Estrutura de funcionamento de uma aplicação Django

- O **Servidor Web**(Nginx) se comunica com a **View**(responsável pela exibição e controle dos dados que serão exibidos).
- A **View** se comunica então com o **Model**(representação dos modelos de dados do sistema em classes Python) e se comunica com o **Template**(HTML modelo que será alterado com base nos dados provenientes da **View**), e por fim envia o retorno ao **Servidor Web**.
- O **Model** por sua vez se comunica com o **RDBM**(Relational Database Management System)(CODD, 1999) e serve a representação do **Banco de Dados** em objetos Python.
- O **RDBM** faz a tradução da busca proveniente do **Model** para o **Banco de Dados** utilizado pelo sistema.

6.2.2 Relação entre componentes do sistema

A atual implementação do sistema AMAO emprega uma única máquina na função de servidor da aplicação propriamente dita e como servidor de Banco de Dados. Nesta mesma máquina funcionam também outros serviços tais como o servidor web e o Message Broker.

Os serviços relacionados com o gerenciamento de projeto de software (TRAC) e de versionamento (GIT) por sua vez, estão hospedados em outra máquina.

No esquema da Figura 52 está representado a forma como cada um dos componentes do sistema se relacionam.

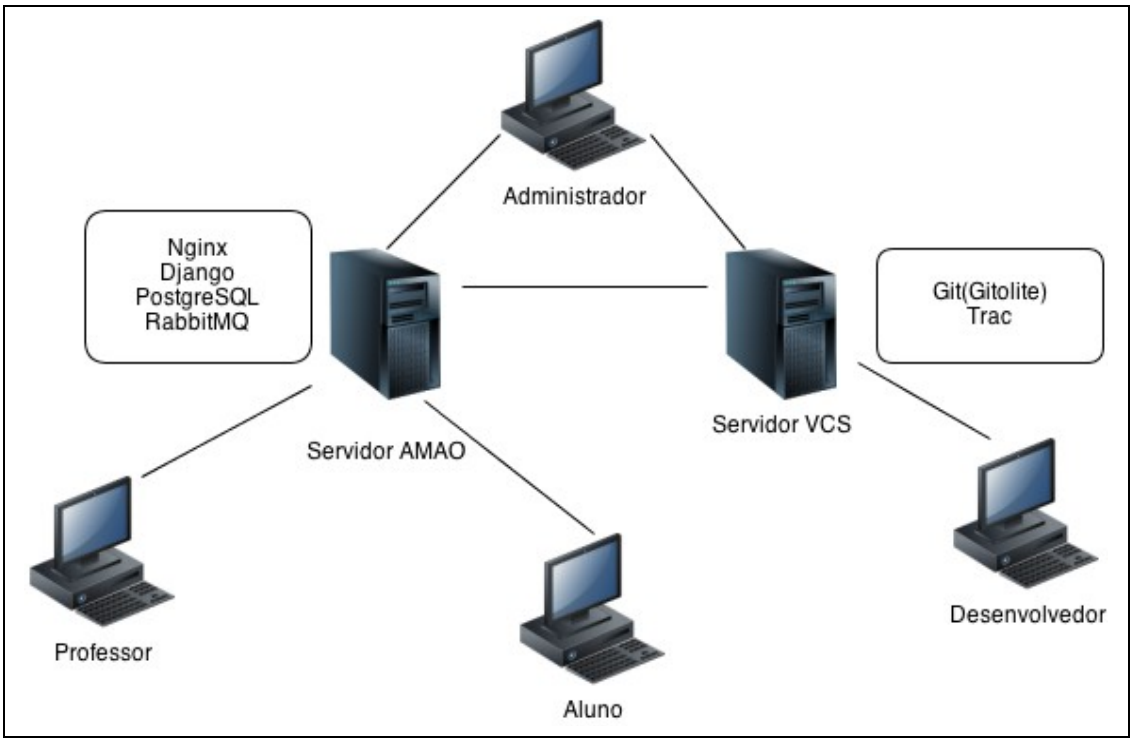

Figura 52 - Relacionamento dos componentes do Sistema AMAO

Descrição de cada um dos elementos:

Servidor AMAO

Nesse servidor rodam os seguintes serviços:

- Servidor Web(Nginx)
- Django(Sistema AMAO)
- Bando de Dados(PostgreSQL)
- Message Broker(RabbitMQ)

Servidor VCS - Version Control System

Esse servidor de controle de versionamento hospeda:

- Servidor GIT(Utilizando o Gitolite[33])
- Servidor Trac

Administrador

Estação de trabalho responsável por administrar ambos os servidores(Servidor AMAO e VCS).

Desenvolvedor

Representa uma estação de trabalho de um desenvolvedor, de onde este envia suas alterações diretamente ao Servidor VCS.

Professor e Aluno

Ambos são estações de onde o Usuário pode acessar o sistema AMAO em produção.

Desde os primeiros estágios do desenvolvimento existiu uma grande preocupação com a escalabilidade, de forma a viabilizar a utilização do AMAO em contextos reais e com uma grande quantidade de alunos realizando avaliações de programação simultaneamente. Neste caso pode-se utilizar a configuração sugerida na Figura 53.

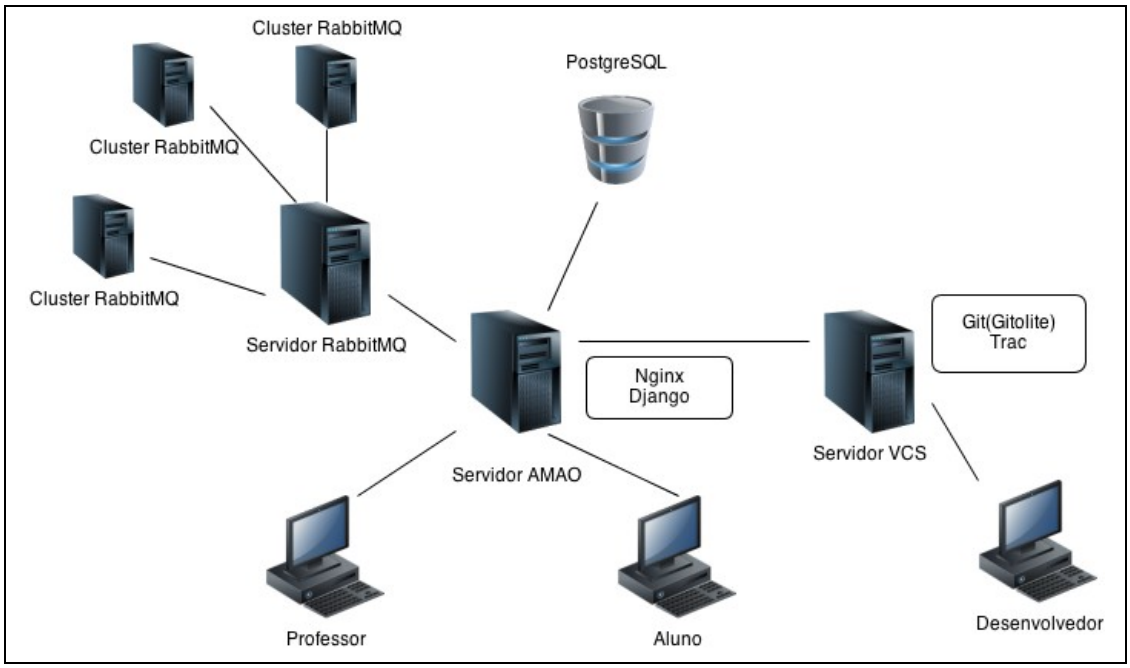

Figura 53 - Esquema de configuração sugerida para uso do sistema AMAO em grande escala

[33] http://gitolite.com/gitolite/

Nessa organização cada serviço pode ser separado em diversos servidores, tornando assim todo o sistema mais rápido.

Além disso pode-se utilizar o **Message Broker**(RabbitMQ) no modo **Servidor → Clusters**[34]. Assim, se houverem 100 correções para serem realizadas ao mesmo tempo, o **Servidor RabbitMQ** irá distribuir entre os **Clusters RabbitMQ** esses processos (ou seja, as correções dos alunos), tornando a execução dos mesmos muito mais rápida.

Outra abordagem que tornaria o ambiente mais rápido é a utilização de um servidor dedicado ao **Banco de Dados**(nesse caso o PostgresSQL).

Nesta representação a figura do Administrador foi retirada, pois considera-se que cada um dos diferentes servidores(VCS, AMAO, RabbitMQ e PostgreSQL) tenham seus próprios responsáveis e a representação destes poluiria essa figura.

[34]Desta forma, no lugar de executar os Workers no servidor, cada maquina cluster tem seus próprios Workers que irão executar os processos enviados pelo servidor.

7 TRABALHOS FUTUROS

Nesta seção serão tratados os possíveis desdobramentos do projeto, funcionalidades, melhorias para o sistema, e vantagens que estas podem trazer.

Análise do código fonte do aluno

Uma característica que pode evoluir no sistema de correção automática é a análise do código fonte durante a correção, de modo que a pontuação não seja atribuída apenas devido ao resultado final estar correto ou não, mas também pelas características do código .

O acréscimo desta funcionalidade tratará o caso, por exemplo, que um aluno fique com nota zero em uma questão que estava correta em quase todos os aspectos exceto por um detalhe, ou no caso em que receba nota máxima mesmo tendo utilizado implementação fraca.

Simulado

Outra funcionalidade que permitiria um maior aproveitamento do sistema por parte dos alunos seria a possibilidade do mesmo poder criar um simulado. Onde um simulado seria uma avaliação que o próprio aluno cria para si, para poder estudar a disciplina utilizando as ferramentas de correção e o banco de questões que o sistema já oferece.

Questões que não sejam de programação

A possibilidade de criar questões que não são de programação ampliaria a área de atuação do sistema que poderia ser usado para outras disciplinas ou áreas que não requerem programação, ou mesmo para avaliações de programação que não utilizam implementação por parte dos alunos.

Integração com Moodle

Uma continuidade interessante para o projeto é a integração com o sistema Moodle.

Tal integração pode ser feita através do Web Service API do Moodle (MOODLE, 2013)[35]. Com isso, pode-se integrar alunos cadastrados neste sistema diretamente no AMAO, além dos cursos, professores e principalmente as avaliações.

Histórico de Submissões

Uma funcionalidade importante para o professor avaliar a resposta do aluno é um histórico de submissões, registrando todos os envios para o sistema, permitindo o professor analisar a evolução na resolução do exercício.

Configurações e Parâmetros para execução

Adicionar opções de configuração para a execução dos exercícios para permitir um maior controle do professor sobre o exercício, tais como tamanho máximo do arquivo, limite de quantidade de memória, limite de tempo de execução e definir os parâmetros de compilação.

Sistema de Dicas

[35] http://docs.moodle.org/dev/Web_services_API

Uma funcionalidade que pode auxiliar o aluno durante a resolução dos exercícios é um sistema de dicas, nesse sistema o professor pode cadastrar os resultados de falhas prováveis de ocorrer na questão e relaciona-los a uma dica sobre o seu possível erro, quando o aluno enviar uma resposta que resulte em algum dos erros cadastrados o sistema apresentará a dica relacionada.

Gerar Relatório de Tuma

Um requisito que pode aumentar a praticidade e a utilização do sistema é a possibilidade de gerar relatórios em CSV sobre o desempenho das turmas, com as notas dos alunos nos exercícios, para facilitar a transferência destes dados para outros ambientes.

Software Livre

Uma proposta futura é de colocar o projeto como Software Livre (Open Software), com o intuito de possibilitar uma maior disseminação do uso sem custos desta ferramenta pela comunidade acadêmica em geral.

Distribuição

Para distribuir melhor o software, e fazer com que este tenha mais notoriedade e colaboração, seria adequado coloca-lo no GitHub[36] (Uma das maiores redes de colaboradores de projeto de Código Aberto que utiliza repositórios Git).

Outra facilidade do GitHub é que este possui seu próprio sistema, similar ao Trac, para gerenciar as tarefas do projeto, além de wikis para facilitar a documentação.

Licença

[36] https://github.com/

Como licença seria utilizada a MIT[37], que de forma sucinta diz que qualquer um pode utilizar o software e seus documentos associados sem qualquer custo (sejam esses para fins lucrativos ou não) contanto que seja sempre distribuído junto o documento original da licença MIT.

Essa licença, bem como a BSD Clause 3[38], é conhecida por ser menos restritiva que outras como a GNU GPL[39].

[37] http://opensource.org/licenses/mit-license.php
[38] http://opensource.org/licenses/BSD-3-Clause
[39] http://www.gnu.org/licenses/gpl.html

8 CONCLUSÃO

O presente trabalho discute a importância de uma plataforma de auxílio à criação, correção e a resolução de avaliações de programação. O objetivo foi propor um sistema, denominado AMAO, de autocorreção, escalável, modularizado, fazendo uso exclusivamente de softwares livres, boas práticas de desenvolvimento e tecnologias que atualmente são utilizadas por empresas de referência no mercado de Tecnologia da Informação.

Num estágio inicial, com o intuito de identificar as características desejáveis na ferramenta proposta, foram levantadas as principais métricas usadas em sistemas autocorretores.

Num segundo momento foram selecionadas e estudadas as diversas bibliotecas e utilitários usados ao longo do processo de desenvolvimento do software. Em cada uma das tarefas envolvidas optou-se pelo uso de recursos que são tendência no segmento de software livre. Em virtude dessa estratégia conseguiu-se ao fim do projeto acumular uma significativa experiência na combinação de tecnologias que têm sido bastante demandadas no mercado, mas que ainda não são discutidas em profundidade no contexto acadêmico.

Os mais importantes pontos observados foram: que a utilização de métodos simples e mais conhecidos de correção automática proporcionaria maior usabilidade do sistema como um todo; o uso deste sistema em disciplinas de correção pode ajudar a reduzir a carga de trabalho de professores e melhorar a experiência prática do aluno; e ainda que a utilização de serviços dedicados à execução em paralelo das correções facilita a manutenção deste processo.

Diante do que foi apresentado, pode-se concluir que o sistema resultante desta pesquisa pode representar um auxílio efetivo ao processo de aprendizado em

disciplinas de programação. Tais benefícios se devem a facilidade com que os professores podem aplicar avaliações e corrigir as mesmas em um prazo mais curto, com menos trabalho repetitivo, auxiliando inclusive através do *feedback* sobre a compilação e execução do gabarito fornecido pelo professor. Por parte do aluno, identifica-se um ganho no que se refere ao modo em que este realiza sua avaliação de programação, permitindo um trabalho mais ágil e dinâmico, mais próximo ao efetivo processo de elaboração de códigos e possibilitando que o aluno dirija seu foco para a lógica da programação propriamente dita. Adicionalmente, o *feedback* proporcionado pelo sistema para o aluno ainda durante as avaliações corresponde a um importante recurso didático, facilitando o processo de aprendizagem.

REFERÊNCIAS BIBLIOGRÁFICAS

Ala-Mutka, K. (2004) Problems in learning and teaching programming – a literature study for developing visualizations in the Codewitz-Minerva project . Codewitz needs Analysis Literature Study, p.1-10.

Almeida, V.C.(2005), Proposta de Implementação de Métricas de Complexidade em um Avaliador Automatizado de Programas – VDSP. Monografia de diplomação do curso de Sistemas de Informação da PUC-MG. Arcos.

Arnow, D. & Barshay, O.(1999), "On-line programming examinations using WebToTeach" In*: Proceedings of the 4th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education*. Krakow, Poland, June 27 – 30, p. 21-24.

Benford, S.D., Burke, E.K., Foxley, E., Gutteridge, N.H., Higgins, C., & Mohd Zin, A. (1994) "Software support for automated assessment and administration", *J. Res. Comput.Edu* , p. 121-131.

Chacon, S.(2009) "Why Git is better than X". Disponível em: <http://thkoch2001.github.io/whygitisbetter/#> Acesso em 30 jul. 2013.

Cheang, B., Kurnia,A., Lim, A. & Oon,W.(2003) "On automated grading of programming assignments in an academic institution", *Computers & Education* , 41, p. 121-131, Elsevier Ltda.

Codd, E.F.(1999), "A relational model of data for large shared data banks", *Communications of the ACM*, vol. 13, issue 6, p. 377-387, New York, NY, USA: ACM.

Daly, C.(1999), "RoboProf and an introductory computer programming course" In: Proceedings of the 4th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, Krakow, p. 155–158.

De Campos, C. P. & Ferreira, C. E.(2004), "BOCA: um sistema de apoio a competições de programação". In: *Workshop de Educação em Computação*, Salvador, BA: Anais do Congresso da SBC, 11 p.

Douce, C., Livingstone, D., & Orwell, J.(2005), "A Technical Perspective on ASAP – Automated System for Assessment of Programming" In*: Proceedings of 9th International Computer Assisted Assessment,*. Leicestershire, UK, June 5 – 6, p. 1-13.

Eby, P.J.(2011) *PPP-333: "*Python Web Server Gateway Interface v1.0". Disponível em: <http://www.python.org/dev/peps/pep-0333/> Acesso em 20 jul. 2013.

Foxley, E. et al. (2001) “The CourseMaster CBA system: Improvements over Ceilid” In: *Proceedings of the Fifth International Computer Assisted Assessment Conference*, Loughborough University, UK, July 2 – 4, p. 189–201.

Freeman, S. E Pryce, N. (2010), Growing Object-Oriented Software,Guided by Tests, USA, New York, NY: Addison-Wesley, 349 p.

Halstead, M.H. (1977) Elements of Software Science (Operating and programming systems series), USA, New York, NY: Elsevier Science Inc.

Higgins, c.a., Gray, G., Symeonidis, P., Tsintsifas, A. (2005) “Automated Assessment and Experiences of Teaching Programming”, *ACM Journal of Educational Resources in Computing,* Vol. 5, No. 3, p. 1-21.

Hunt, A., Thomas, D. (1999) The Pragmatic Programmer: From Journeyman to Master. USA: Addison-Wesley Professional, 352 p.

Hyvönen, J. & Malmi, L. (1993) “TRAKLA – A system for teaching algorithms using email and a graphical editor” In: *Proceedings of the HYPERMEDIA Conference.* Vaasa, Finlândia, p. 141-147.

Jackson, D., Usher, M. (1997) “Grading student programs using ASSYST” In: *Proceedings of the 28th SIGCSE Technical Symposium on Computer Science Education*, San Jose, CA, USA, p. 335–339

Kaplan-moss, J., Holovaty, A.(2007) “The Definitive Guide to Django: Web Development Done Right”. USA: Apress, 447 p.

Kjollerstrom, B. & Martensson, M. (2009) “Assessment: The Key to Changing the Way We Learn”, *CAL-Laborate* , vol. 3, p. 17-20, Sweden: UniServe Science,.

Luck, M. & Joy, M.S. (1999) “A secure on-line submission system”, *Softw. – Pract.Exper*, vol. 29, issue 8, p. 721-740, John Wiley & Sons, Ltd.

LUZ, E.B. (2011) “Python e django na globo.com.” [Slides da Palestra]. Disponível em: <http://www.slideshare.net/ricobl/python-e-django-na-globocom>. Acesso em 3 ago. 2013.

Martins, V.F., Fernandes, D., Grégio, A. (2009) “Homemade Sandbox: Como construir rapidamente um analisador de Malware para responder à sua vítima”. 14ª Reunião do Grupo de Trabalho em Segurança (GTS). São Paulo, SP, [Slides da Palestra]. Disponível em: <ftp://ftp.registro.br/pub/gts/gts14/06-Homemade-sandbox.pdf>. Acesso em 3 ago. 2013.

McCabe, T.J. (1976) “Automated Assessment and Experiences of Teaching Programming”, *IEEE Transactions on Software Engineering,* Vol. SE-2, NO.4, p. 308-320..

Moodle (2012) “Question Bank.”Disponível em: <http://docs.moodle.org/24/en/Question_bank> Acesso em 30 jul. 2013.

Moodle (2013) “Web Services API.”. Disponível em: <http://docs.moodle.org/dev/Web_services_API> Acesso em 30 jul. 2013.

Moreira, M.P. & Favero, E.L.(2009) “Um Ambiente para Ensino de Programação com Feedback Automático De Exercícios” In: *XXIX Congresso da Sociedade Brasileira de Computação*, Bento Gonçalves, RS: UFRGS, p. 429-438.

Pontes, F.A. (2011) “Usando VirtualEnvWrapper.” Disponível em: <http://www.arruda.blog.br/programacao/python/usando-virtualenvwrapper>. Acesso em 3 ago. 2013

Prior, J.C. (2003) “Online assessment of SQL query formulation skills” In*: Proceedings of the Fifth Australasian Conference on Computing Education*. Adelaide, Australia: Australian Computer Society, Inc, p. 247-256.

Queirós, A.R.P, Leal, J.P. (2012) “PETCHA: a programming exercises teaching assistant” In: *Proceedings of the Annual Conference on Innovation and Technology in Computer Science Education - ITiCSE*, p. 192–197.

Rees, R. (2011) “Large django sites at mozilla - Andy McKay”. Disponível em: <http://reinout.vanrees.org/weblog/2011/06/06/large-mozilla-sites.html>. Acesso em 3 ago. 2013.

Richardson A. et al (2013) “Introduction to RabbitMQ - An open source message broker that just works.” Google London. London, UK, 2008. [Slides da Palestra]. Disponível em: <http://www.rabbitmq.com/resources/google-tech-talk-final/alexis-google-rabbitmq-talk.pdf>. Acesso em 2 ago..

Saikkonen, R., Malmi, L., Korhone, A. (2001) “Fully automatic assessment of programming exercises” In: *Proceedings of the 6th annual conference on Innovation and technology in computer science education*. New York, NY, USA: ACM, p. 133-136.

Sukkarieh, J.Z., Pulman, S.G., Raikes, N. (2003) “Auto-Marking: Using Computational Linguistics to Score Short, Free Text Responses” In: *Proceedings of the 29th Annual Conference of the International Association for Educational Assessment*, 15 p.

Tobar, C. M, Rosa, J.L.G.,Coello, J.M.A. & Pannain, R. (2001) “Uma Arquitetura de Ambiente Colaborativo para o Aprendizado de Programação” In: *XII Simpósio Brasileiro de Informática na Educação*, Vitória, ES: UFRGS, p. 244-252.

Tulder, G. (2003), “Storing Hierarchical Data in a Database.” Disponível em: <http://www.sitepoint.com/hierarchical-data-database-2/> Acesso em 10 jul. 2013.

Wang, T. et al. (2011) “Ability-training-oriented automated assessment in introductory programming course”, *Computers & Education*, vol. 56, issue 1, p. 220-226, Oxford, UK: Elsevier Science Ltd,.

Williams, C. (2012) “Building with Django: In Good Company”. Disponível em: <http://www.fiveq.com/blog/programming/building-django-good-company/>. Acesso em 3 ago. 2013.

Yates, c. et al. (2006) Expert Spring MVC and Web Flow. USA: Apress,. 424 p.

ANEXO 1: INSTALAÇÃO DE DESENVOLVIMENTO

O processo de instalação do sistema para um local de desenvolvimento (descrito antes em linhas gerais), será agora detalhado considerando-se o sistema operacional: **Ubuntu 12.04 Desktop 64Bits**

1 Instalar o GIT

É necessário ter o GIT instalado para poder instalar algumas aplicações que estão em repositórios no GitHub.

sudo apt-get install git-core git-guigit-doc

2 Configurando Chave RSA

O acesso ao repositório GIT requer a criação de uma chave SSH para o usuário.

Assim, na maquina de trabalho deve ser executado:

ssh-keygen -t rsa -C "exemplo_email@exemplo.com"

Durante o processo de instalação será perguntado o nome e local do arquivo, neste ponto deve-se apenas apertar [Enter] para o instalador criar na pasta padrão (~/.ssh/id_rsa.pub).

Ao pedir a passphrase deve ser colocado uma senha, que será utilizada para enviar ou alterar algo no repositório GIT do servidor.

A chave RSA que está no arquivo **id_rsa.pub** deve ser então enviada ao administrador do repositório do AMAO para ele adicionar a mesma.

3 Instalando o PIP

PIP é um programa que controla dependências de forma simples.

sudo apt-get install python-setuptools python-dev build-essential

sudo easy_install pip

4 Instalando o Ambiente Virtual

VirtualEnv:

pip install virtualenv

VirtualEnvWrapper:

pip install virtualenvwrapper

5 Configurando o Ambiente Virtual

Criar pasta para os virtuaisEnvs:

$mkdir ~/.virtualenvs

Configurar o .bashrc:

gedit ~/.bashrc

Colocar no final do mesmo:

export PIP_RESPECT_VIRTUALENV=true

export WORKON_HOME=~/.virtualenvs

source /usr/local/bin/virtualenvwrapper.sh

export VIRTUALENVWRAPPER_VIRTUALENV_ARGS='--no-site-packages --distribute'

Após salvar o bashrc é preciso fechar e abrir uma nova janela de terminal(ou deslogar e logar) para as serem executados os scripts do wrapper ou deve se executar:

source ~/.bashrc

6 Criando e Ativando o Ambiente Virtual

Para criar o ambiente para o sistema:

mkvirtualenv AMAO

Para ativar o mesmo basta fazer:

workon AMAO

Após ser executado o comando acima, o terminal ficara parecido com:

(AMAO)usuario@computador:~$

7 Instando as Dependências

Dentro da pasta do projeto(rais) executar o comando:

pip install -r etc/requirements.txt

Para tornar o arquivo **manage.py** executável:

chmod +x manage.py

8 Instalando Outras Dependências

Para instalar as outras dependências é preciso rodar o comando:

sudo pip install venv-dependencies; sudo apt-get install graphviz-dev; sudo apt-get install python-pygraphviz

Relacionar outras bibliotecas no virtualEnv do AMAO:

workon AMAO

link_venv.py "pygraphviz"

9 Configurando o SafeExec

Compilar o SafeExec no servidor, para isso:

cd (...)amao/AMAO/safeexec

make

10 Instalando o Django-Celery e o Celery Beat

Para instalar o RabbitMQ:

sudo apt-get install rabbitmq-server

Para rodar o celery:

python manage.py celeryd -l info

Para executar o Celery-Beat:

python manage.py celerybeat

ANEXO 2: INSTALAÇÃO DE PRODUÇÃO

O processo de instalação do sistema para um local de produção (descrito antes em linhas gerais), passos a seguir foram feitos executados nos sistemas operacionais: **Ubuntu 10.10 Server 64 Bits, Ubuntu 12.04 Server 64 Bits, Ubuntu 10.10 Desktop 64 Bits.**

A senha utilizada como senha padrão será a mesma para tudo durante esse processo para facilitar o mesmo.

1 Criando Usuário Amao

Primeiramente deve-se logar como o usuário root no sistema, em seguida abrir uma janela de terminal e digitar:

useradd -d /home/amao -m amao

passwd amao

Ao pedir a senha deve ser colocada a senha padrão.

Em seguida é necessário inserir o usuário no grupo de SUDO:

adduser amao sudo

Dessa forma quando acessar o servidor será usado o usuário amao, e caso precise virar root basta usar o comando:

sudo su

2 Configurando Chave RSA

O acesso ao repositório GIT requer a criação de uma chave SSH para o usuário.

Assim, na maquina de trabalho deve ser executado:

ssh-keygen -t rsa -C "amao@uniriotec.br"

Durante o processo de instalação será perguntado o nome e local do arquivo, neste ponto deve-se apenas apertar [Enter] para o instalador criar na pasta padrão (~/.ssh/id_rsa.pub).

Ao pedir a passphrase deve ser colocado uma senha, que será utilizada para enviar ou alterar algo no repositório GIT do servidor.

A chave RSA que está no arquivo **id_rsa.pub** deve ser então enviada ao administrador do repositório do AMAO para ele adicionar a mesma.

3 Habilitando Login do Root

Abrir uma janela de terminal e digitar:

sudo passwd root

Isso pedirá uma senha para o usuário root, essa senha deve ser a mesma do usuário amao.(senha padrão)

4 Habilitando Serviço SSH

Para habilitar o ssh:

sudo apt-get install openssh-server openssh-client

Instalado o servidor ssh os próximos passos já podem ser realizados por uma conexão ssh com o servidor, ex:

ssh root@amao.uniriotec.br

VIM

É preciso instalar o vim para poder editar os arquivos pelo SSH.

apt-get install vim

5 Instalando o GIT

É necessário ter o GIT instalado para poder instalar algumas aplicações que estão em repositórios no GitHub.

sudo apt-get install git-core git-guigit-doc

6 Instalando Nginx

Para instalar o Nginx:

apt-get update

apt-get upgrade

apt-get install nginx

Em seguida, para rodar o servidor(pela primeira vez):

/etc/init.d/nginx start

Para configurar o Nginx corretamente:

vim /etc/nginx/sites-available/amao

No novo arquivo criado, colocar o seguinte conteúdo:

```
server {
        listen 80;
        server_name localhost;
        location / {
            # host and port to fastcgi server
            fastcgi_pass 127.0.0.1:8000;
            fastcgi_param PATH_INFO $fastcgi_script_name;
            fastcgi_param REQUEST_METHOD $request_method;
            fastcgi_param QUERY_STRING $query_string;
            fastcgi_param CONTENT_TYPE $content_type;
            fastcgi_param CONTENT_LENGTH $content_length;
            fastcgi_pass_header Authorization;
            fastcgi_intercept_errors off;
            fastcgi_param REMOTE_ADDR $remote_addr;
            }
        location /media  {
            root /var/www/medias/AMAO;
        }
        location /static  {
            root /var/www/medias/AMAO;
        }
}
```

Acessar a pasta:

cd /etc/nginx/sites-enabled/

Rodar o comando:

ln -s ../sites-available/amao

Retirar a configuração que vem por padrão no nginx:

rm default

7 Instalando o Banco de Dados

Para instalar o PostgreSQL:

apt-get install postgresql-8.4 postgresql-server-dev-8.4

Em seguida definir a senha para o usuário postgres:

passwd postgres

A senha é a padrão.

Para criar o usuário para o django:

sudo -u postgres createuser -P django_login

A senha é a padrão.

Usar o sudo nesse caso por conta do -u

Mudar para o usuário postgres:

su postgres

Entrar na shell:

psql template1

Para criar o banco de dados:

CREATE DATABASE django_db OWNER django_login ENCODING 'UTF8';

Para sair do shell utilize **q**, e ctrl+D para sair do usuário postgres.

Para alterar as configurações de permissão do postgres:

vim /etc/postgresql/8.4/main/pg_hba.conf

Na linha:

local all postgres ident

Colocar abaixo dela a seguinte:

local django_db django_login md5

Reiniciar o PostgreSQL:

/etc/init.d/postgresql restart

Instalar a engine de conexão:

apt-get install python-psycopg2

8 Obtendo os Fontes

Para obter o projeto deve se ir para a pasta onde este ficará:

mkdir -p /var/www/

cd /var/www

Clonar a pasta:

sudo -H -u amao git clone -b deploy git@arruda.blog.br:amao

A senha necessária para esse procedimento, novamente, é a padrão.

9 Instalando o PIP

PIP é um programa que controla dependências de forma simples.

sudo apt-get install python-setuptools python-dev build-essential

sudoeasy_install pip

10 Instalando e configurando o Ambiente Virtual

VirtualEnv:

pip install virtualenv

VirtualEnvWrapper:

pip install virtualenvwrapper

Criar pasta para os virtuaisEnvs:

$mkdir ~/.virtualenvs

Configurar o .bashrc:

vim ~/.bashrc

Colocar no final do mesmo:

export PIP_RESPECT_VIRTUALENV=true

export WORKON_HOME=~/.virtualenvs

source /usr/local/bin/virtualenvwrapper.sh

export VIRTUALENVWRAPPER_VIRTUALENV_ARGS='--no-site-packages --distribute'

Após salvar o bashrc deve se executar:

source ~/.bashrc

Para criar o ambiente para o sistema:

mkvirtualenv AMAO

Para ativar o mesmo basta fazer:

workon AMAO

Após ser executado o comando acima, o terminal ficara parecido com:

(AMAO)usuario@computador:~$

11 Instando as Dependências

Dentro da pasta do projeto(rais) executar o comando:

pip install -r etc/requirements.txt

Para tornar o arquivo **manage.py** executável, deve ser executado:

chmod +x manage.py

Por fim instalar no virtualenv:

pip install psycopg2

12 Configurando o SafeExec

Compilar o SafeExec no servidor:

cd /var/www/amao/AMAO/safeexec

make

Alterar o dono para ser do usuário amao:

chown amao:amao /var/www/amao/AMAO/safeexec/safeexec

13 Ativando o Ambiente de Produção

Adicionar ao .bashrc uma variável de ambiente para dizer ao Django que está em um ambiente de produção:

vim ~/.bashrc

E incluir no final do mesmo:

export AMAO_ENV='PROD'

alias clear_pyc='find ./ -name ".pyc" | xargs rm -v'*

Para as alterações surtirem efeito:

source ~/.bashrc

14 Arrumando Media/Static

Criar as pastas de media e estáticos:

mkdir -p /var/www/medias/AMAO/media

mkdir -p /var/www/medias/AMAO/static

Popular a pasta de arquivos estáticos com o conteúdo do projeto e das aplicações do Django, no ambiente virtual do sistema, e na pasta do projeto deve se executar o seguinte comando:

./manage.py collectstatic

15 Carregando o Django

Instalar a biblioteca que permitirá a execução do Django:

pip install flup

Para carregar o Django, deve se executar o seguinte comando:

python manage.py runfcgi host=127.0.0.1 port=8000 --settings=settings pidfile=/var/www/amao/AMAO/amao_server.pid

Para automatizar o processo de executar e para o Django deve se criar 2 scripts. Primeiro para executar:

vim /var/www/amao/AMAO/run_server.sh

Dentro do arquivo:

/var/www/amao/AMAO/manage.py runfcgi host=127.0.0.1 port=8000 --settings=settings pidfile=/var/www/amao/AMAO/amao_server.pid

O arquivo para parar o servidor:

vim /var/www/amao/AMAO/stop_server.sh

Com esse conteúdo:

kill `cat /var/www/amao/AMAO/amao_server.pid`

16 Sincronizando o BD

Sincronizar o banco de dados:

./manage.py syncdb

O super-usuario deve ser amao, com email amao@uniriotec.br**.**

A senha deve ser a padrão.

Algumas aplicações não são sincronizadas, elas devem ser migradas uma-a-uma(bug com PostgreSQL).

./manage.py migrate Materia

./manage.py migrate Professor

./manage.py migrate Aluno

./manage.py migrate Materia.Turma

./manage.py migrate Avaliacao

./manage.py migrate Avaliacao.Questao

./manage.py migrate Corretor

./manage.py migrate djcelery

17 Instalando o Django-Celery

Como o Django-Celery já foi instalado anteriormente junto com as dependências, é necessário instalar apenas o RabbitMQ:

sudo apt-get install rabbitmq-server

Criar o arquivo de configurações no seguinte caminho:

vim /etc/default/celeryd

Colocar o seguinte conteúdo:

 # Name of nodes to start, here we have a single node

CELERYD_NODES="w1"

 # Where to chdir at start.

CELERYD_CHDIR="/var/www/amao/AMAO/"

 # Python interpreter from environment.

ENV_PYTHON="/root/.virtualenvs/AMAO/bin/python"

 # How to call "manage.py celeryd_multi"

CELERYD_MULTI="$ENV_PYTHON $CELERYD_CHDIR/manage.py celeryd_multi"

 # How to call "manage.py celeryctl"

CELERYCTL="$ENV_PYTHON $CELERYD_CHDIR/manage.py celeryctl"

 # Extra arguments to celeryd

CELERYD_OPTS="--time-limit=300 --concurrency=8"

 # Name of the celery config module.

CELERY_CONFIG_MODULE="celeryconfig"

 # %n will be replaced with the nodename.

CELERYD_LOG_FILE="/var/log/celery/%n.log"

CELERYD_PID_FILE="/var/run/celery/%n.pid"

 # Workers should run as an unprivileged user.

CELERYD_USER="root"

CELERYD_GROUP="root"

AMAO_ENV="PROD"

 # Name of the projects settings module.

export DJANGO_SETTINGS_MODULE="settings"

Em seguida:

vim /etc/init.d/celeryd

E em seu conteúdo coloca-se o conteudo do arquivo Celeryd[40]

Alterar o arquivo para ser executável:

sudo chmod +x /etc/init.d/celeryd

Também será necessário criar as seguintes pastas:

mkdir -p /var/log/celery

mkdir -p /var/run/celery

18 Configurando o CeleryBeat

Criar o arquivo de configurações no seguinte caminho:

vim /etc/default/celerybeat

Colocar nele o seguinte conteúdo:

 # Where the Django project is.

CELERYBEAT_CHDIR="/var/www/amao/AMAO/"

ENV_PYTHON="/root/.virtualenvs/AMAO/bin/python"

 # Path to celerybeat

CELERYBEAT="$ENV_PYTHON $CELERYBEAT_CHDIR/manage.py celerybeat"

 # %n will be replaced with the nodename.

CELERYBEAT_LOG_FILE="/var/log/celerybeat/celerybeat.log"

CELERYBEAT_PID_FILE="/var/run/celerybeat/celerybeat.pid"

[40] Disponível em: http://svn.arruda.blog.br/trac/AMAO/wiki/Celeryd_conteudo

 # Workers should run as an unprivileged user.

CELERYBEAT_USER="root"

CELERYBEAT_GROUP="root"

 # Extra arguments to celerybeat

CELERYBEAT_OPTS="--schedule=/var/run/celerybeat-schedule"

AMAO_ENV="PROD"

 # Name of the projects settings module.

export DJANGO_SETTINGS_MODULE="settings"

Em seguida cria-se o arquivo:

vim /etc/init.d/celerybeat

Coloca-se o conteúdo do arquivo do Celerybeat[41].

Em seguida se altera o arquivo para ser executável:

sudo chmod +x /etc/init.d/celerybeat

19 Executando o Celery

Para executar o celery:

sudo /etc/init.d/celeryd start

Para ver o status do celery:

sudo /etc/init.d/celeryd status

Para parar o celery:

sudo /etc/init.d/celeryd stop

Os comandos funcionam da mesma forma para o **Celery Beat**, ex:

sudo /etc/init.d/celerybeat start

[41] Disponível em: http://svn.arruda.blog.br/trac/AMAO/wiki/Celerybeat_conteudo

ANEXO 3: REPRESENTAÇAO COMPLETA DO BANCO DE DADOS

Abaixo se encontra o diagrama completo com a representação do banco de dados do sistema, a imagem do diagrama ampliado se encontra em: https://docs.google.com/file/d/0BxjQaemeNb8EaEk4QlVXcTNLLTQ/edit?usp=sharing

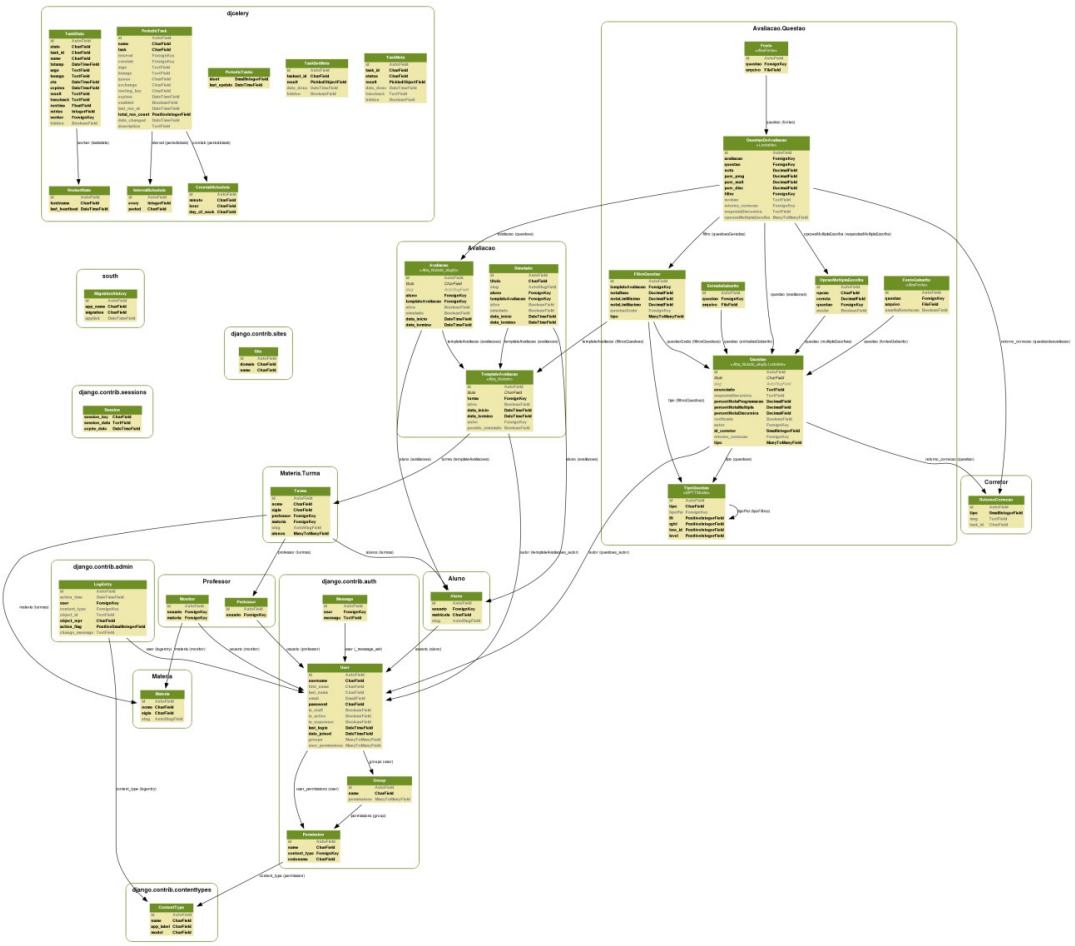

Figura 54 - Diagrama completo com a representação do banco de dados.