UNIVERSIDADE FEDERAL DO ABC

Sistema PGC

Manual do desenvolvedor

Renan Bernardo – Bernardo_mec@live.com

28/08/2014

Esse manual tem como objetivo auxiliar os próximos estagiários a entender a lógica empregada nesse sistema para realizar manutenções e melhorias

Conteúdo

Introdução

O Ruby on Rails é uma linguagem orientada a objetos utilizada em aplicações para a WEB com banco de dados.

Ela tem como base o Ruby, onde foi desenvolvida a Gem (API) Rails, que oferece as funções que facilitam a programação para a WEB. Caso não conheça a linguagem, procure pelo livro Pragmatic Agile Web Development with Rails (encontrado na internet em .PDF e na pasta "Manuais" desde projeto). Os capítulos introdutórios dão uma boa noção sobre seu uso.

Um dos maiores problemas que encontrei sobre o Rails foi a incompatibilidade entre suas versões. Ele é constantemente atualizado e entre as versões 3.2 e 4, por exemplo, muitos métodos foram adicionados e outros removidos, então as vezes encontrava material ou Gems que não eram compatíveis com a versão que eu usava e o programa não funcionava, quando me garantiam que iria dar certo... Por isso recomendo fortemente usar em seus projetos as versões Ruby 1.93 e Rails 3.2.8, para não correr o risco de desenvolver um sistema funcional em localhost, mas que apresenta erros no deploy e consultar sempre o site http://guides.rubyonrails.org/ antes de instalar uma Gem. Ele fornece informações sobre os métodos da linguagem e em quais versões ele é funcional.

A programação em Rails utiliza o padrão MVC: Model, View e Controller. Esse padrão divide o código em camadas para facilitar a programação e futuros debugs.

Model: Contém as regras do jogo.

Controller: Define o comportamento da aplicação.

View: Apresenta as informações ao usuário.

Mais informações sobre esse padrão podem ser encontradas em http://www.macoratti.net/vbn_mvc.htm.

Instalação do Ruby on Rails

1) Acesse o site http://rubyinstaller.org/downloads/ e baixe a versão 1.93 do Ruby. Na mesma página, baixe o pacote Development Kit compatível com essa versão.
2) Instale o Ruby.
3) Descompacte o Development Kit dentro do diretório do Ruby, com o nome DevKit
4) Abra a prompt de comando do Windows e entre no diretório do DevKit
5) Digite "ruby dk.rb init" no terminal para gerar o arquivo "config.yml", no diretório do DevKit
6) Abra o arquivo acima e escolha em qual(is) versão do Ruby o DevKit será instalado (Apague as que não serão utilizadas). Caso não haja nenhuma outra versão do Ruby instalada, nesse arquivo conterá apenas a versão 1.93, então não é necessário alterá-lo.

7) Digite "ruby dk.rb install" para completar a instalação.
8) Ainda no terminal/prompt de comando, digite "gem install rails –v3.2.8" e aguarde a instalação. Pode levar vários minutos até que seja concluído.
9) Acesse http://dev.mysql.com/downloads/connector/c/ e baixe a versão 6.0.2 do pacote. Faça o download do arquivo no formato .zip. Caso você tenha acesso ao servidor do BCC, basta copiar a pasta C:\\ mysql-connector
10) Descompacte o arquivo na raiz C:\\ do seu computador com o nome "mysql-connector". Caso você tenha copiado a pasta do servidor, no passo acima, cole-a no mesmo diretório e com o mesmo nome.
11) Para instalar a Gem Mysql2, responsável por conectar a aplicação ao banco de dados, acesse o terminal e digite gem install mysql2 --platform=ruby -- '--with-mysql-dir="C:\mysql-connector"'
12) Entre na pasta C:\\mysql-connector\lib e copie todos os arquivos libmysql para a pasta Bin, localizada dentro do diretório do Ruby

Caso tudo tenha dado certo, o Rails estará instalado e pronto para uso.

Iniciar um projeto

1) Abra o terminal/prompt de comando e entre dentro do diretório onde o projeto em Ruby on Rails será instalado.
2) Digite "rails new [nome do projeto] –d mysql. Aguarde a instalação. O Ruby irá criar os arquivos do projeto dentro da pasta escolhida. O comando –d indica que o projeto irá utilizar o banco de dados MySQL
3) Ainda no terminal, entre dentro da pasta do projeto recém criado e teste a aplicação digitando "rails s". O navegador será aberto, digite localhost:3000 (porta padrão para a ambiente de desenvolvimento). Você será redirecionado à sua aplicação, ainda vazia.

Instalando e preparando a IDE para programar

Existem diversar IDEs para trabalhar com o RoR. Eu utilizei o NetBeans. Por padrão, ele não é compatível com essa linguagem, mas no próprio site do desenvolvedor é possível baixar o pacote de compatibilidade.

1) Acesse o site https://netbeans.org/downloads/, baixe o pacote de instalação e instale-o. Pode ser necessária também a instalação do pacote Java.
2) Acesse http://plugins.netbeans.org/PluginPortal/ e utilize os filtros para procurar o plugin do Ruby on Rails. Faça o download do pacote e instale-o utilizando a opção "Plugins", localizado dentro do menu "Ferramentas", no Netbeans . Tive problemas durante a instalação, caso também ocorra, instale um por um todos os arquivos da pasta, atentando-se aos erros. Por exemplo, você tenta instalar o arquivo1 e retorna um erro dizendo que o arquivo2 não foi encontrado. Instale o arquivo2 e em seguida tente instalar o instalar o arquivo1, se retornar um erro dizendo que não foi possível porque não encontrou o arquivo3, instale-o e repita o processo até que todos eles sejam instalados.

3) A instalação desse plugin adiciona as opções generate, bundle, console entre outras, à IDE do NetBeans.
4) Instale as Gems que liberam o Debug aos seus projetos. Abre o terminal/prompt de comando do Windows e digite gem install debugger e em seguida gem install ruby-debug-ide.
5) O sistema se encontra hospedado dentro do servidor do BCC, na pasta E:\\RailsProjects\Graduation. Copie ela para sua pasta de projetos do NetBeans.
6) Ainda dentro da prompt de comando do Windows, entre dentro da pasta do projeto e digite "bundle install" e "bundle update" para instalar em seu computador todas as Gems utilizadas no projeto.

Procure o livro Pragmatic Agile Web Development with Rails, ou o site http://guides.rubyonrails.org/v3.2.18/ para aprender a linguagem. Recomendo que faça alguns exemplos para adquirir experiência.

IMPORTANTE: O Rails em alguns casos NÃO RECONHECE LETRAS COM ACENTOS!!!! Demorei muito tempo para descobrir isso. Caso esse tipo de caractere precise ser colocado em uma *hash*, utilize a *Gem Brazilian Rails*. O principal sinal de que esse erro ocorreu será o seguinte aviso na tela:

COMO CORRIGIR:

1) **Isole o erro:** Esse passo é muito trabalhoso. O Rails não avisa onde ocorre você cometeu esse erro, ele pode estar em qualquer Controller, View ou Helper, mesmo que não esteja sendo executado na ação que você está trabalhando no momento. Quando acontecia, eu verificava todos os arquivos que tinha modificado naquele dia e checava se haviam acentos, então trocava o nome para um qualquer e tentava executar novamente.
2) Utilize a Gem Brazilian Rails: Abra o arquivo Configuration/locales/PT-BR.yml. Esse arquivo contém as traduções de expressões/palavras para português.
3) Digite logo abaixo da ultima tradução informada [identificação da tradução]: [texto traduzido ou com acentos]
4) Na sua variável/hash que apresenta erro, passe o parâmetro t(:[identificação da tradução])

Um exemplo dessa correção encontra-se em ProjectsController#recusabanca: Ao passar para o flash[:notice] o texto "Houve um erro na sua solicitação". O sistema apresentou a mensagem de erro durante os testes, pois o Rails não soube processar o "ç" e o "ã". Ao utilizar a tradução, ele funcionou corretamente.

Para evitar esse tipo de problema, teste sua aplicação sempre que precisar utilizar acentos e faça o procedimento acima caso ocorram erros.

Erro

Em ProjectsController#recusabanca

```
    flash[:notice] = "Houve um erro no envio da avaliação!"
  else
    flash[:notice] = "Houve um erro na sua solicitação."
  end
```

Solução

Em ProjectsController#recusabanca

```
    flash[:notice] = t(:analise_pelo_orientador)
  else
    flash[:notice] = t(:erro_recusa_banca)
  end
```

Em Configuration/locales/PT-BR.yml

```
arquivo_vazio: O relatório não foi anexado.
erro_atualizar_projeto: Não foi possível atualizar o projeto.
arquivo_vazio: Essa ação não pôde ser executada. Por favor anexe o relatório
analise_pelo_orientador: Seu projeto foi encaminhado ao orientador para análi
erro_recusa_banca: Houve um erro na sua solicitação.
avaliacao_salva: Avaliação salva com sucesso.
erro_avaliacao: Houve um erro no envio da avaliação!
```

A parte técnica do sistema se encontra como comentários no código. Visualize os arquivos para mais detalhes sobre a lógica empregada.

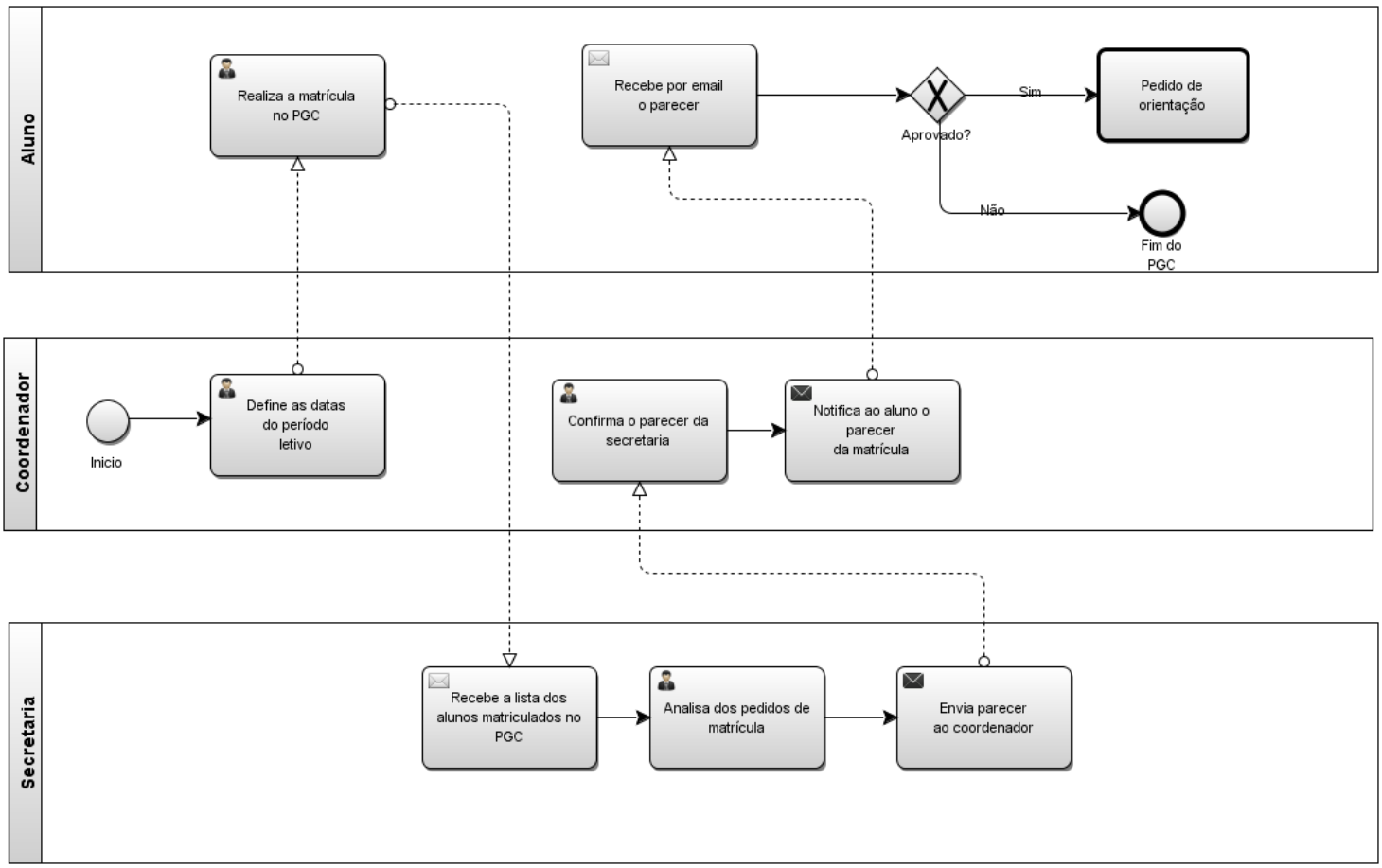
O sistema

Ele estava pela metade quando me passaram essa tarefa. Eu não tinha experiência com o Rails nem com programação orientada à objetos. Durante meu trabalho, me pediram para acrescentar funções, remover outras.... Eu optei por não remover nada do que já estava feito, apenas colocá-lo como comentário, porque eu poderia usar como base para aprender a linguagem.

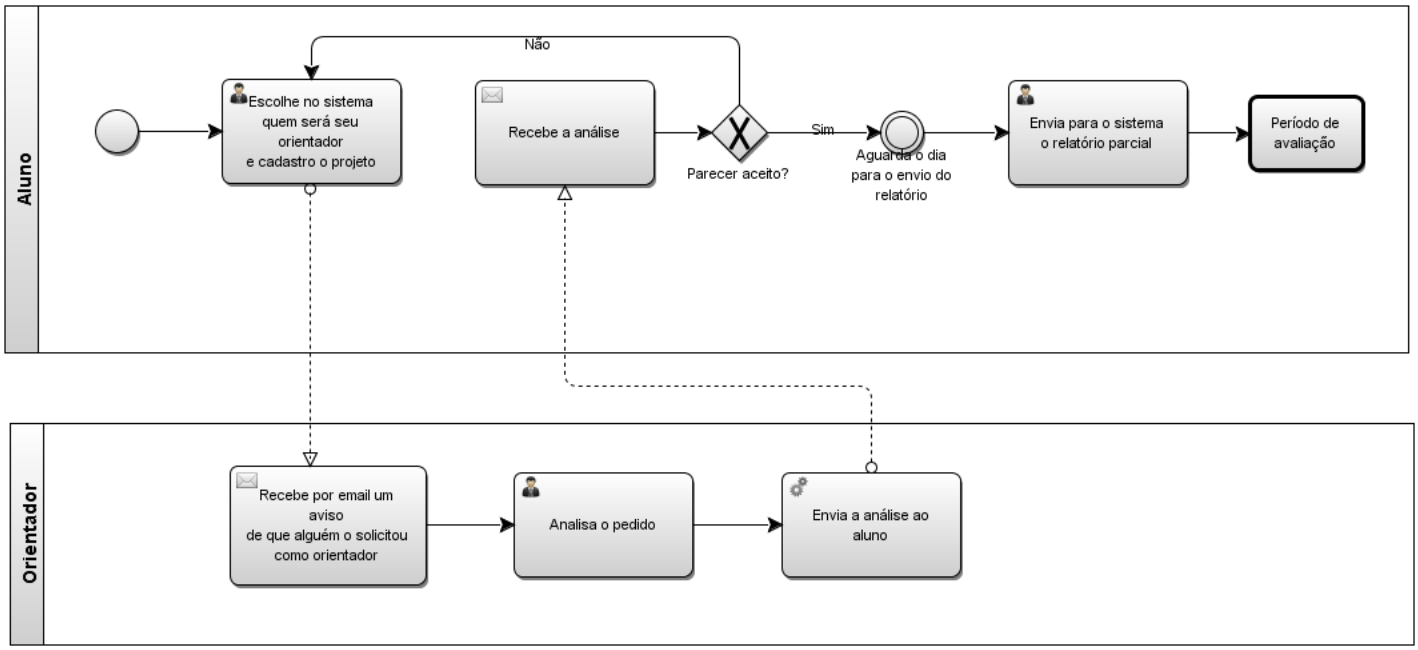
Para que o sistema funcione corretamente, pelo menos um dos usuários deve ser definido como sendo do tipo Secretaria. Ele possui permissão para alterar todos os outros. Quando o sistema for iniciado, cadastre o primeiro usuário como Professor e então altere manualmente no banco de dados para Secretaria (altere o User.role). Cadastre em seguida outro e utilize o usuário de secretaria para trocar o tipo para Coordenador, assim o sistema pode iniciar normalmente.

Fluxogramas

Matrícula no PGC



Pedido de Orientação



Período de avaliação

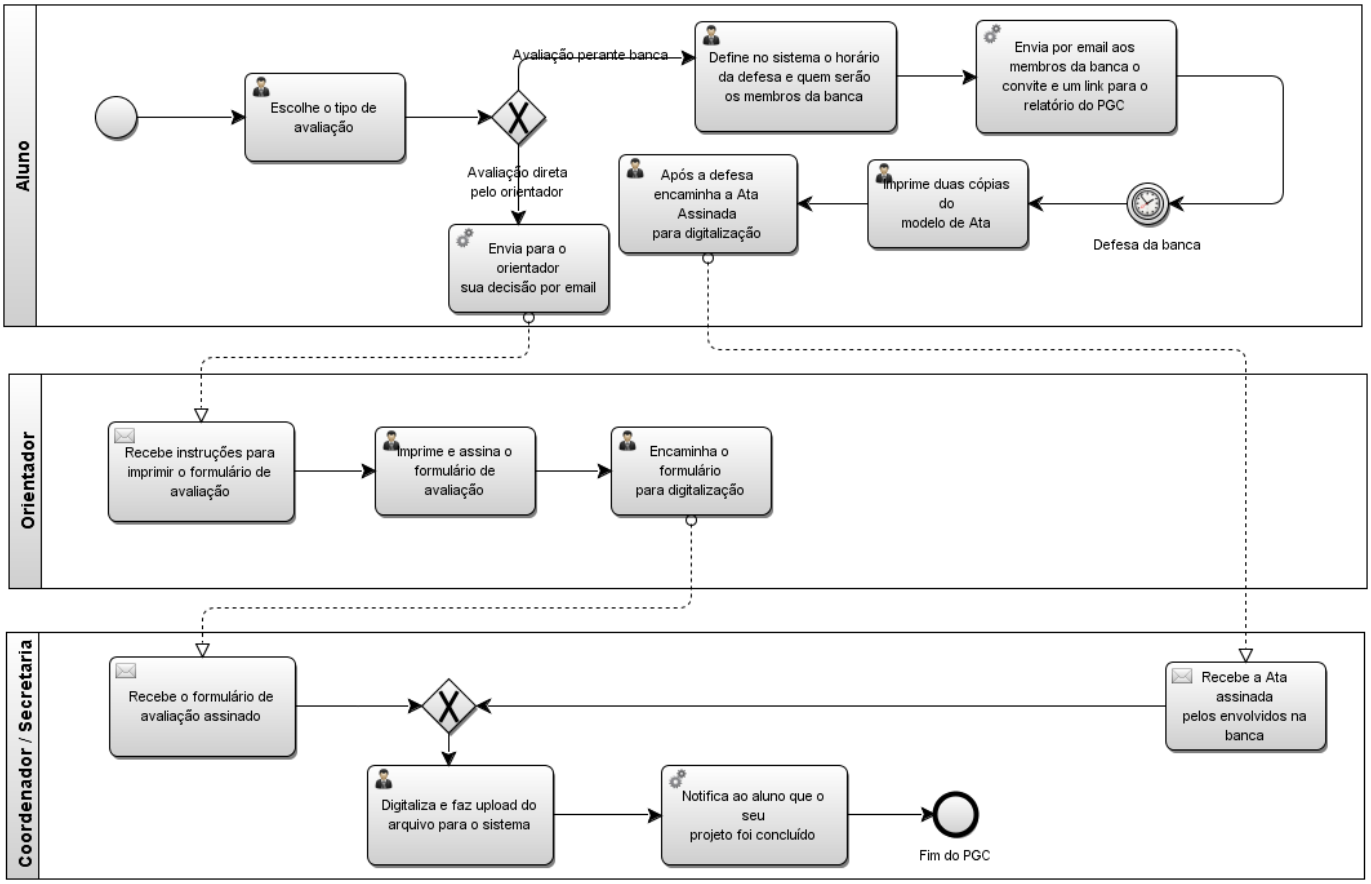
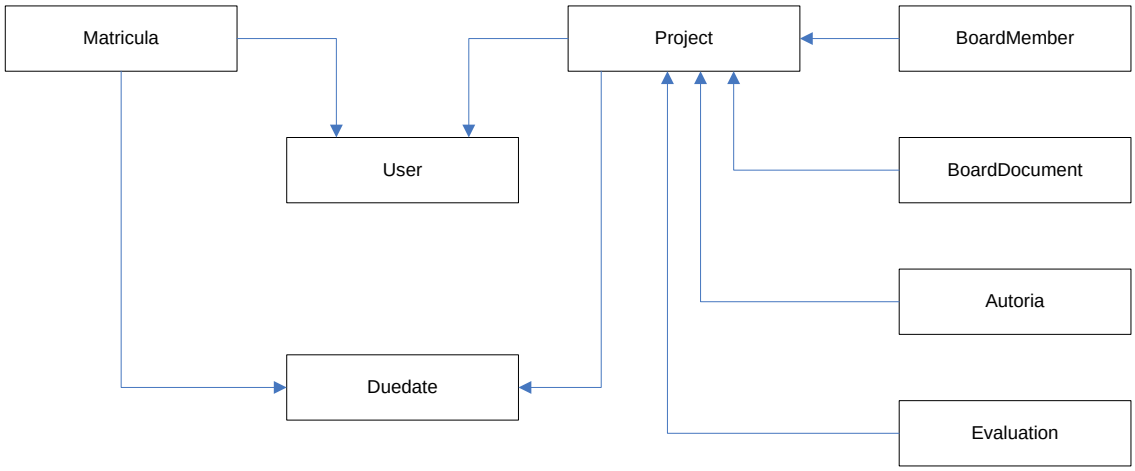


Diagrama de classes



Passo a passo dos processos do PGC

1) O cadastro no sistema

As operações de login são realizadas pela Gem Devise: Com ela é possível fazer cadastro e login no sistema. Ela utiliza o model "User".

As views relacionadas ao login estão na pasta "Views -> Device" e utiliza o registration_controller.

Quando um usuário se loga no sistema, o objeto "current_user" armazena suas informações e permite acesso a elas. No model, algumas opções podem ser definidas removendo os comentários (#).

confirmable: O sistema será aberto ao usuário apenas se ele confirmar sua conta através de um link que recebe por email

lockable: Bloqueia o usuário se ele fizer muitas tentativas mal sucedidas de login.

Timeoutable: O sistema efetua o logoff se o usuário ficar muito tempo ocioso

Recoverable: permite a recuperação da senha

Rememberable: permite lembrar a senha em um cookie do navegador, para o usuário não precisar informá-la sempre que realizar o login

Trackable: gera informações de rastreamento, como o contador do número de logins, IPs etc

Ainda no model, observe que as validações bloqueiam tentativas de cadastro cujo RA seja repetido, não seja um número ou tenha tamanho diferente de 8 se o usuário for um aluno e exige que seja informado o nome.

Ao efetuar alguma mudança nesses parâmetros, o servidor precisa ser reiniciado. Para mais informações sobre a Gem Devise, consulte seu manual clicando aqui.

Mais informações

Quando um usuário se cadastra como professor, ele ainda não possui acesso ao sistema, mesmo quando confirma o email. O coordenador ou algum usuário do tipo secretaria deve confirmá-lo como orientador antes que suas funções sejam liberadas. Isso será descrito com mais detalhes aqui.

2) O coordenador do curso define um período letivo

Todas as informações sobre os períodos letivos estão armazenados no model "duedate".

Como o sistema seria testado, me pediram para remover as datas limites!

Esse model foi gerado utilizando o Scaffold e adaptado para impedir entradas inválidas. No site, ele pode ser acessado no menu esquerdo "Definir período letivo" quando o usuário logado é um coordenador.

3) Aluno solicita matrícula no PGC

Assim que o período de matrícula se inicia, o aluno pode solicitar o cadastro no sistema pelo menu Matrícula. Ações relacionadas às matrículas estão no model "matricula".

3.1) Pedido de matrícula: É apresentado ao aluno algumas informações sobre os critérios para que seu pedido seja aceito e ele deve informar em qual PGC deseja se matricular.

Ações utilizadas nesse processo
MatriculasController#pedido
MatriculasController#solicita

3.2) Análise da matrícula: A secretaria recebe uma lista com as solicitações de estágio recebidas. Essa lista pode ser acessada pelo menu "Analisar pedidos de matrícula". Assim que o parecer é definido, o coordenador precisa confirmar a análise e então o parecer é enviado por email ao aluno.

Ações utilizadas nesse processo (tanto para a secretaria como para o coordenador)
MatriculasController#confirma
MatriculasController#update

4) Realizar o pedido de orientação

Assim que o pedido de matrícula é aprovado, o aluno é notificado por email e o sistema é aberto para que ele solicite um orientador e inicie um projeto. O menu "Pedido de Orientação" estará disponível para o aluno assim que houverem matriculas dele na situação 2.

4.1)Criação do projeto: O aluno acessa o sistema e faz o pedido de orientação a um dos professores cadastrados no sistema.

Ações utilizadas nesse processo
ProjectsController#new
ProjectsController#create

4.2) Aceite do pedido: O orientador recebe por email uma mensagem avisando que o aluno solicitou orientação à ele. Deve acessar o sistema e responder se aceita ou não.

Ações utilizadas nesse processo
ProjectsController#accept
ProjectsController#update

Não é mais necessária a aprovação do projeto pelo coordenador após o aceite do orientador. O projeto é atualizado para o status 2 quando o pedido é deferido, caso contrário para -1 e ele estará apto a fazer um novo pedido, desde que esteja dentro do prazo.

5) Avaliação parcial

Assim que o orientador aceita o pedido de orientação, ele estará apto a preencher o formulário de avaliação parcial. O menu "Realizar Avaliação Parcial do Orientador" está disponível para orientadores sempre que houverem projetos no status 2 relacionados à eles.

Ações utilizadas nesse processo
ProjectsController#evaluate
EvaluationsController#new
EvaluationsController#create

A opção "O aluno está apto a continuar com o projeto?" é utilizada pelo sistema para casos onde o aluno não pode prosseguir com o projeto, porém, ele ainda não será avaliado.

5.1.1) Aluno não apto a continuar: Esse processo pode acontecer em diversos casos, como por exemplo o aluno não comparecer as reuniões e desistir do projeto.

O status do projeto é atualizado para 8, sem que o aluno envie o relatório. O próximo procedimento é o mesmo que a avaliação direta pelo orientador.

5.1.2) Aluno apto a continuar: Muda o status do projeto para 7, permitindo que o aluno envie para o sistema o relatório parcial do projeto.

6) Envio do relatório parcial do projeto

Assim que o orientador envia para o sistema o formulário de avaliação parcial, o aluno recebe um email solicitando o envio do relatório parcial do projeto e o menu "Enviar relatório" estará disponível.

Ações utilizadas nesse processo
ProjectsController#sendfile
ProjectsController#update

Mais informações

O relatório do projeto será salvo em **Project.relatorio** utilizando a Gem Carrierwave. O *Model Project* foi configurado para permitir apenas arquivos no formato .PDF

7) Avaliação

Assim que ele envia o relatório, o menu "Opções de avaliação" estará disponível. Nesse menu é possível escolher entre avaliação seja direta pelo orientador ou por defesa de banca.

Ações utilizadas nesse processo
ProjectsController#banca

7.1) Avaliação direta pelo orientador: O aluno optou por ser avaliado pelo orientador. Ao selecionar essa opção, o orientador recebe um email informando sobre a decisão e que deve acessar o sistema para enviar imprimir o formulário de avaliação, assinar e entregar para digitalização. O status do projeto será atualizado para 8 e o menu "Gerar formulário de avaliação parcial" ficará disponível para ambos. Ele é gerado em formato .PDF utilizando a *Gem Prawn* (O manual desta *Gem* se encontra na pasta "Manuais" desde projeto).

Ações utilizadas nesse processo
ProjectsController#recusabanca
ProjectsController#view as PDF

Após devidamente assinado, ele deve ser encaminhado à secretaria acadêmica para digitalização. Quando houver projetos no status 8 o menu "Enviar formulário de avaliação" ficará disponível para usuários do tipo Secretaria ou Coordenador. Ele deve enviar o formulário em formato .PDF e em seguida o conceito. Após o envio o status do projeto é alterado para 6 e o aluno recebe uma mensagem avisando que o mesmo está concluído.

Ações utilizadas nesse processo
ProjectsController#index_avaliacao
ProjectsController#formulario_avaliacao
ProjectsController#envia_formulario

Mais informações

O formulário de avaliação é anexado ao banco de dados em **Project.form_avaliacao** utilizando a Gem Carrierwave.

7.2) Avaliação perante banca: O aluno optou por ser avaliado pela banca. Esse processo é obrigatório para alunos matriculados no PGC III e, caso contrário, opcional.

Ao escolher essa opção, ele será redirecionado a uma página onde deverá informar ao sistema quem serão os membros da banca avaliadora e onde ela ocorrerá.

Os membros da banca são armazenados no Model BoardMember. Ele foi gerado utilizando o Scaffold e configurado para receber apenas emails válidos e checar a presença das informações (cheque o Model para mais detalhes). Também é criado o Model BoardDocument, que armazena informações sobre a defesa, a Ata digitalizada e o conceito. Para evitar muitas alterações no código que já estava pronto, utilizei esse modelo para armazenar também os conceitos de alunos que foram avaliados pelo orientador.

Ações utilizadas nesse processo
ProjectsController#editbanca
ProjectsController#update

7.2.1) Gerar o modelo de Ata: Assim que o formulário de composição de banca for preenchido, o status do projeto é atualizado para 4 e a opção "Gerar modelo de Ata" ficará disponível no menu esquerdo para o aluno e para o orientador do projeto. Nesse menu é possível gerar o modelo de Ata de defesa para o projeto e revisar as informações, caso sejam necessárias alterações no local.

Ações utilizadas nesse processo
ProjectsController#gerarata
BoardDocumentsController#edit
BoardDocumentsController#update
BoardDocumentsController#atamanual as PDF

Assim que as informações sobre a defesa da banca são fornecidas, a opção "visualizar modelo de Ata" ficará disponível. Ele é gerado utilizando a *Gem Prawn* (O manual desta *Gem* se encontra na pasta "Manuais" desde projeto). O status do projeto é alterado **apenas** quando a Ata digitalizada é enviada ao sistema.

7.2.2) Envio da Ata digitalizada: Após a defesa da banca, a Ata deve ser encaminhada para digitalização. O Menu "Enviar Ata digitalizada" estará disponível para usuários do tipo Secretaria ou Coordenador, onde ele deverá selecionar anexar o arquivo em formato .PDF e informar o conceito. Nesse ponto existem 3 caminhos diferentes:

7.2.2.1) Aluno reprovado na banca: Essa opção provavelmente nunca será utilizada, porém está disponível para o caso de acontecer. Como o aluno foi reprovado na banca, ele não irá enviar para o sistema o relatório final, então é solicitado à secretaria/coordenador que informe quem foram os membros que participaram da defesa para registro. O status do projeto é atualizado para 9 assim que essas informações são passadas.

Ações utilizadas nesse processo
ProjectsController#atamanual
BoardDocumentsController#update
ProjectsController#enviabanca

7.2.2.2) Aluno matriculado no PGC I ou PGC II: A defesa da banca não é obrigatória para esses casos, logo não é necessário o envio do formulário de

autoria. Assim que a Ata for enviada, o status do projeto é atualizado para 12 e um email é enviado ao aluno solicitando o envio do relatório final, com as modificações feitas pela banca. O menu "Enviar relatório final" ficará disponível para ele no menu esquerdo. Ao acessar essa opção com um projeto no status 12, é solicitado apenas o relatório e os membros da banca. Após o envio dessas informações, o aluno recebe um email informando a conclusão do projeto e o status é alterado para 6.

Ações utilizadas nesse processo
ProjectsController#atamanual
BoardDocumentsController#update
ProjectsController#enviarelatorio
ProjectsController#concluir_pgc

7.2.2.3) Aluno matriculado no PGC III: Para concluir o Projeto de Graduação em computação é necessário preencher os formulários de autoria e de publicação. Essas informações são armazenadas no Model Autoria. Assim que a Ata é enviada ao sistema, o status do projeto é alterado para 10 e um email é enviado ao aluno solicitando o envio do relatório final, com as modificações feitas pela banca. O menu "Enviar relatório final" ficará disponível para ele no menu esquerdo. Ao acessar essa opção com um projeto no status 10, todos os formulários são solicitados. Após o envio dessas informações, o aluno recebe um email informando a conclusão do projeto e o status é alterado para 6.

Ações utilizadas nesse processo
ProjectsController#atamanual
BoardDocumentsController#update
ProjectsController#sendfinalfile
ProjectsController#concluir

Outras informações do sistema

Alterar o menu lateral

O menu lateral contém os links para os usuários acessarem as funções do sistema. Ele se encontra na *View Application.html.erb*, dentro da pasta Layouts.

Dentro do menu, cada item deve estar dentro da tag "li". Se ele estiver disponível, deve ser apresentado um link ao usuário, caso contrário, ele deve aparecer opaco, como mostra o exemplo abaixo.

```
<%if existe_projetos_aptos_para_banca? %>

        <li><%= link_to "Opções de avaliação" , "/pgc/banca" %> </li>

        #caso a comparação retorne verdadeiro, é apresentado o link

 <%else%>

        <li><h1>Opções de avaliação</h1></li>
```

```
        #caso contrário, o texto aparece opaco.
<%end%>
```

Note também que cada tipo de usuário do sistema possuí um menu próprio.

Prevenção de erros

Bloqueio a conteúdo indevido: Você notará as seguintes linhas em muitos métodos:

```
  if possui_acesso?(params[:id])
    #código
  else
   render "notifications/entrada_incorreta", alert: "!"
  end
```

Essa função foi criada para evitar que usuários acessem conteúdo indevido, ela se encontra em Application_controller.rb.

Deve ser passado nos parâmetros da função o ID do projeto. Caso a função retorne *true* o usuário pode seguir com sua solicitação, caso contrário é redirecionado à NotificationsController#entrada_incorreta e a ação é interrompida.

Criar avisos para os usuários do sistema

Usuários do tipo Coordenador podem criar avisos que serão vistos no home Page de todos que acessarem o sistema. O link "novo aviso" estará disponível na página principal quando o coordenador fizer o login no sistema.

Esse modelo foi criado utilizando o scaffold e não necessitou de configurações adicionais.

Mailers

Mailers são actions do Ruby on Rails para o envio de emails.

O mailer utilizado no sistema do PGC é o "*UserMailer*". Seu *controller* se encontra na pasta *app/mailers/user_mailer.rb*, suas *views* na pasta *views/user_mailer* e suas configurações estão nos arquivos *configuration/application.rb* e *configuration/environm ent/development.rb*

Instalando um mailer: abra o terminal/prompt de comando e digite "rails generate mailer [nome do mailer]". Caso haja um mailer instalado, não é necessária a criação de outro, a não ser que o endereço do remetente precise ser outro ou por outro motivo.

Configurando o mailer: Acesse o arquivo configuration/application.rb e acrescente a seguinte linha:

```
  #action mailer
```

config.action_mailer.default_url_options = { :host => "[nome do servidor]" }

Caso o servidor seja o do BCC, o servidor será http://cmcc-bcc.ufabc.edu.br

IMPORTANTE: Caso você esteja editando o projeto no seu computador, troque-o para localhost:3000 (ou a porta que você esteja utilizando). O nome do servidor é utilizado pela *Gem Devise* no envio dos emails de cadastro, recuperação de senhas e afins, então essa campo deve ser preenchido

Acesse o arquivo configuration/environment/development.rb e acrescente a seguinte linha:

```
 #devise mailer
 config.action_mailer.default_url_options = { :host => '[nome do servidor]' }
#mailer config gmail
 config.action_mailer.delivery_method = :smtp
 config.action_mailer.smtp_settings = {
  :address            => "smtp.gmail.com",
  :port               => 587,
  :domain             => "smtp.gmail.com",
  :user_name          => '[login de email em uma conta google]',
  :password           => '[senha do email]',
  :authentication     => 'plain',
  #:tls =>true,
  :enable_starttls_auto => true


  }
```

No sistema, o email utilizado foi **cmccbcc@gmail.com** cuja senha é **ufabcadmin,** atente-se ao nome do servidor.

Caso esses arquivos sejam alterados, o servidor de aplicação precisa ser reiniciado. Vide "Fazendo alterações no projeto do servidor".

Enviando emails: Em seu controller, chame o seguinte método:

UserMailer.[identificação do email] ([objetos enviados ao método, caso sejam necessários]).deliver

Acesse o controller do Mailer (app/mailers/user_mailer.rb) e acrescente as linhas abaixo dentro da classe:

```
def [identificação do email]([objetos enviados ao método])

    #acrescente aqui suas variáveis

    mail(:to => "#{[nome do destinatario]} <#{[email do destinatário]}>",
    :subject => "[título da mensagem]")

end
```

Em seguida, crie dentro da pasta views/user_mailer/ um arquivo chamado [nome do método].html.erb. Dentro desde arquivo, escreve o email.

Para mais informações sobre os Mailers, clique aqui.

Listar projetos

Esse menu está disponível para usuários do tipo Secretaria e Coordenador. Ao escolher o período, é apresentada uma lista com todos os projetos que ocorreram, sendo possível também acessar mais informações.

Ações utilizadas nesse processo
ProjectsController#periodo
ProjectsController#index_coordenador

Gerar relatório de bancas

Esse menu está disponível para usuários do tipo Secretaria e Coordenador. Ele retorna um relatório em .PFD contendo todos os projetos que foram avaliados perante banca.

Ações utilizadas nesse processo
DuedatesController#relatorio
DuedatesController#relatorio_de_banca

Gerar relatório de conceitos

Esse menu está disponível para usuários do tipo Secretaria e Coordenador. Ele gera um relatório com os conceitos dos projetos concluídos e apresenta o uma lista com o status atual dos projetos pendentes.

Ações utilizadas nesse processo
ProjectsController#relatorio
DuedatesController#show

Gerar relatório de matrículas

Esse menu está disponível para usuários do tipo Secretaria e Coordenador. Ele gera um relatório com as matrículas efetuadas no período letivo.

Ações utilizadas nesse processo
DuedatesController#relatorio_de_matricula
DuedatesController#view_relatorio_de_matricula

Alternar entre as views de Coordenador e Orientador

O projeto antigo não previa que um coordenador também pode ser um orientador. Para evitar alterações significativas no que já estava pronto foi definida a coluna "Coordenador" no Model User. Caso ela seja verdadeira o usuário é um coordenador e pode alternar entre as opções de coordenador e orientador.

Para efetuar essa troca, assim que o usuário se logar no sistema o link "alternar tipo de usuário" estará disponível na parte superior da página. Ao clicar o tipo de usuário e trocado para Coordenador caso o atual seja Orientador e vice-versa.

Ações utilizadas nesse processo
UsuariosController#change

Editar usuários

Usuários do sistema podem ser editados pela secretaria ou pelo coordenador.

Ações utilizadas nesse processo
UsuariosController#show
UsuariosController#edit
UsuariosController#update

Liberar acesso de orientador ao sistema

Quando um orientador realiza o cadastro no sistema, ele precisa ter sua identidade confirmada pelo coordenador. Quando existem análises pendentes o menu "Analisar pedidos de cadastro de orientador" estará disponível.

Ao realizar o cadastro, o orientador é definido como sendo do tipo "Professor". Esse procedimento procura por todos os usuários desse tipo e, sendo aprovados, os altera para "Orientador", podendo assim acessar as funções do menu esquerdo.

Ações utilizadas nesse processo
NotificationsController#confirma
NotificationsController#validar

Colocar o projeto no servidor

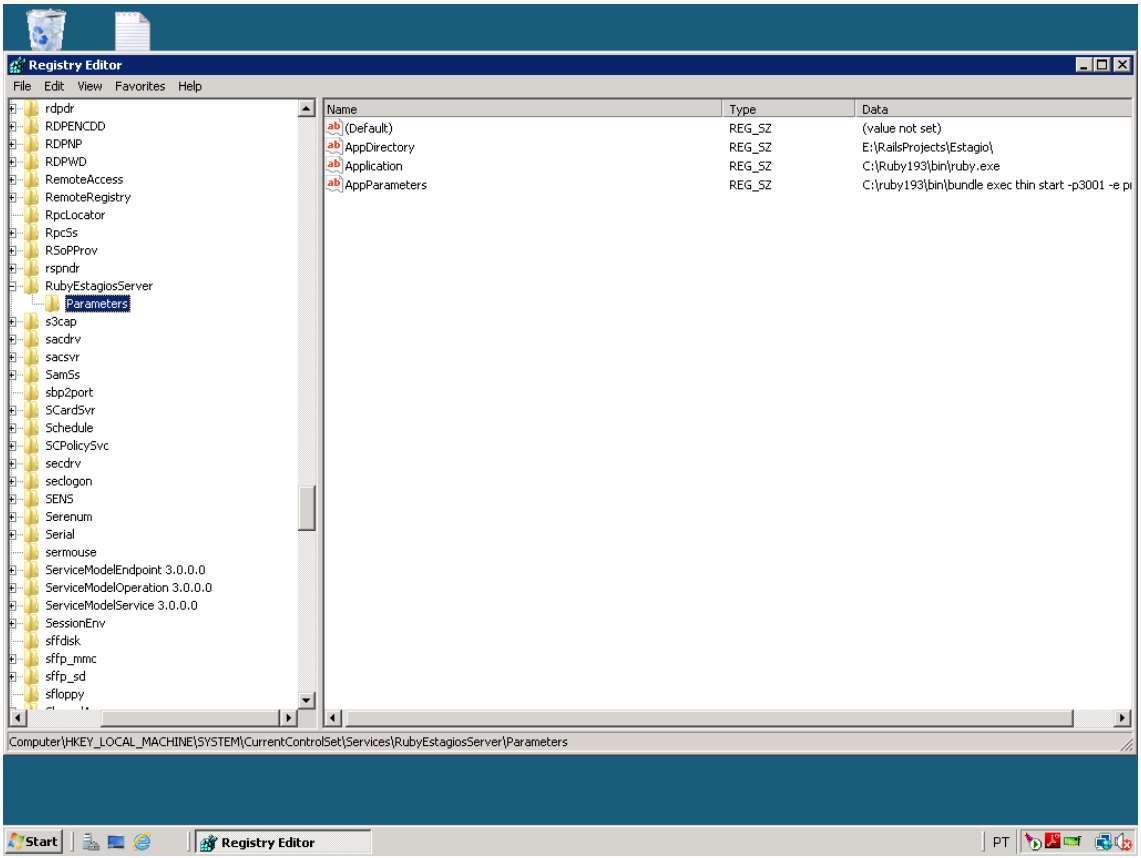
A partir desse ponto, vou considerar que você já possui os conhecimentos em Ruby on Rails para prosseguir sem muito detalhamento. Esse procedimento é utilizado para a CRIAÇÃO DE UM NOVO PROJETO, informações sobre como editar o projeto existente se encontram aqui.

1) Instale a Gem thin em seu computador e coloque-a no Gemfile do projeto
2) Acesse o servidor cmcc-bcc.ufabc.edu.br através do programa "Conexão de área de trabalho remota", localizado no menu iniciar => todos os programas -> acessórios. Caso não tenha um login e uma senha, peça para a professora Drª Juliana Braga criar um para você.

3) Copie a pasta do seu projeto para dentro do servidor, no diretório E:\\RailsProjects. Pasta copiar e colar.
4) Aperte com o botão direito em cima do ícone da prompt de comando e escolha ‘‘executar como administrador’’
5) Entre dentro da pasta do projeto e digite ‘‘bundle install’’ para instalar no servidor possíveis Gems que você tenha utilizado em seu projeto, mas que não estão instaladas no servidor
6) Ainda dentro do terminal, entre no diretório system32, dentro da pasta do Windows
7) Digite sc create [NomeDoServiço] binPath= "c:\windows\system32\srvany.exe" DisplayName= "[Nome do Serviço]". O nome do serviço deve ser algo que você possa identificar, por exemplo Ruby[nome do seu projeto]Server.
8) Digite regedit no menu iniciar e execute o programa. Localize seu service na pasta [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\]
9) Aperte com o botão direito e clique em new -> key. Renomeie a pasta para ‘‘Parameters’’
10) Dentro dessa pasta, crie os seguintes strings: substituindo o endereço se necessário.

    "Application"="C:\Ruby193\bin\ruby.exe"
    "AppDirectory"="[diretório onde seu projeto está]"
    "AppParameters"="C:\Ruby193\bin\bundle exec thin start -p [um numero de porta] -e[‘‘development’’ ou ‘‘production’’, informe qual banco de dados seu projeto vai usar]"

    Segue um exemplo desse passo.



11) Digite services no menu iniciar
12) Procure seu service criado acima e clique em "iniciar"
13) Abra o navegador e digite localhost:[porta escolhida]. veja se sua aplicação será iniciada. Caso tenha acontecido algum erro, abra a prompt de comando, entre dentro do diretório do projeto e digite "rails s" ou "bundle exec thin start", caso apareça algum erro, corrija-o, reinicie o service e tente novamente
14) Quando o projeto puder ser acessado dentro do servidor pelo navegador, Solicite ao NTI que libere uma porta para acesso global, liberando no firewall. Por exemplo a porta 3002 ou outra que não esteja sendo utilizada.
15) Assim que o acesso for liberado, dentro do servidor, entre na pasta C:\\program files (x86)\Apache software Foundation\apache 2.2\conf
16) Abra o arquivo httpd.conf e digite abaixo de um campo onde contem linhas similares à essas o seguinte texto:

ProxyPass /[o scope que você utilizou nas rotas do seu projeto] http://localhost:[porta utilizado no deploy da qual o NTI liberou o acesso]/[o scope que você utilizou nas rotas do projeto]

ProxyPassReverse /[o scope que você utilizou nas rotas do seu projeto] http://localhost:[porta utilizado no deploy]/[o scope que você utilizou nas rotas do projeto]

Exemplo:

ProxyPass /estagios http://localhost:3001/estagios

ProxyPassReverse /estagios http://localhost:3001/estagios

17) Reinicie o Apache e veja se sua aplicação pode ser acessada de fora do servidor (digitando cmcc-bcc.ufabc.edu.br/[scope do seu projeto])

Fazendo alterações no projeto do servidor

Caso seja necessário atualizar algum dos projetos hospedados no servidor, siga as instruções abaixo.

1) Digite services no menu iniciar
2) Encontre todos os serviços que utilizam o Ruby e desative-os (atualmente, esses serviços são "meu serviço" e "Ruby Estagios Server".
3) Clique com o botão direito em cima da barra de tarefas e escolha "Task Manager"
4) Na guia "process" Mude o filtro para mostrar todos os processos em andamento e procure pelos que se chamam "ruby.exe", encerre todos.
5) Copie os arquivos modificados para suas respectivas pastas.
6) Entre novamente em Services e ative os serviços listados acima.

IMPORTANTE: Caso os arquivos alterados sejam Views, Controllers, pequenas alterações nos Models, que não exigem que o servidor de aplicação seja reiniciado ou novas migrações, não é necessário interromper os serviços, apenas substitua os arquivos em suas respectivas pastas.

NUNCA substitua inteiramente a pasta do projeto pela sua! Muitos arquivos são de configurações e já estão prontos, a mudança da pasta inteira pode acarretar em perda de funcionalidades.

Caso você tenha instalado *Gems* que não estavam nesse projeto, substitua também o arquivo *gemfile.lock* e em seguida instale-a no servidor. Caso o sistema apresente algum erro durante sua iniciação, abra o prompt de comando, acesse a pasta do projeto e execute o "bundle exec" e "bundle update".

O banco de dados do sistema

Todas as informações sobre o sistema estão armazenadas no banco de dados "Graduation_development" dentro do servidor cmcc-bcc.ufabc.edu.br.

Entre no servidor pelo terminal de acesso remoto e abra o software MySQL Workbench. O banco está em "Local Instance MySQL55", utilize a senha "ufabc" para ter acesso às informações.

Gems úteis

Debugger + ruby-debug-ide = Usadas para permitir o debug

Devise: Usado para criar logins

gem "prawn", "~> 1.0.0.rc1" : Para gerar documentos em PDF

Carrierwave : Usado para fazer upload de arquivos

brazilian-rails : Permite traduções em português

thin: Utilizado para o deploy da aplicação (colocá-la na internet para acesso externo)

A Gem Carriewave – upload de arquivos

A *Gem Carrierwave* é utilizada para receber arquivos anexos como os formulários digitalizados e a Ata do Projeto de Graduação.

Instalação:

Abra o prompt de comando do Windows e digite "gem install carrierwave". Em seguida, ainda dentro da Prompt de comando, entre dentro da pasta do projeto e digite "rails generate uploader [Nome do arquivo que será carregado]".

Configurações presentes no sistema PGC

O uploader foi chamado "relatorio" e seu arquivo de configuração pode ser encontrado em app/uploaders/relatório_uploader.rb. Ele foi configurado para salvar os arquivos enviados ao sistema na pasta "upload" na raiz do sistema. (dentro do servidor cmcc-bcc.ufabc.edu.br, no diretório E:\\RailsProjects\Graduation). Caso seja necessário reiniciar o banco de dados, **RECOMENDO** que faça uma cópia desses arquivos.

No *model Project*, observe que foi definido as colunas "relatorio" e "form_avaliacao" entre "mount_uploader" e "*RelatorioUploader*". Essa configuração define que o campo é reservado para um arquivo. Para que eles pudessem ser salvos, foi colocado em *attr_accessible* o [nome do campo] e também [nome do campo]_cache, por exemplo [:relatório] e [:relatorio_cache]. Essa configuração é obrigatória.

Criando novos espaços para arquivos

Para criar um novo espaço para armazenar arquivos, utilize a migração do *Rails* para gerar uma nova coluna no banco de dados com o tipo string. Siga o exemplo acima para configurar o *model* e especificar que ele é reservado a arquivos. Nos formulários da view, utilize o campo *file_field* :[nome do arquivo]e *hidden_field* :[nome do arquivo]_cache, assim como no exemplo abaixo, retirado da view project#sendfinalfile

```
<%= project_form.file_field :relatorio %>

<%= project_form.hidden_field :relatorio_cache %>
```

Mais informações

Para mais informações, acesse o manual do desenvolvedor da *Gem* em https://github.com/carrierwaveuploader/carrierwave.

Brazilian Rails – Tradução de testes e padrões do português

A Gem Brazilian Rails é utilizada para traduzir textos e alterar padrões. Os arquivos de tradução encontram se no arquivo Configuration/locales/ em extensões .yml.

Nesse arquivo é possível traduzir modelos, mensagens de erro, escolher padrões de dias/hora entre outras.

Models utilizadas e suas descrições

Aluno_projeto

Esse modelo tinha sido criado para relacionar projetos aos alunos. Não é mais utilizado.

Anexo

Esse modelo era utilizado para armazenar arquivos em anexo. Não é mais utilizado.

Autoria

Modelo utilizado para armazenar informações do formulário de autoria e de declaração de orientação.

Declaração_autoria: Variável booleana que o aluno declarou estar ciente sobre a autoria.

Banca_[1,2]: Armazena quem foram realmente os membros da banca. Para o caso de ter participado um professor suplente.

Autorização: Variável que indica qual foi a resposta para a autorização de publicação

1 = Liberar o conteúdo dos arquivos para acesso público.

2 = Liberar o conteúdo dos arquivos somente para a comunidade da Universidade.

3 = Reter o conteúdo dos arquivos por motivos de patente, publicação e/ou direitos autorais.

4 = Liberar o conteúdo de alguns arquivos para acesso público, restringir o conteúdo de outros arquivos para acesso somente da comunidade da Universidade e/ou reter o conteúdo de alguns arquivos por motivos de patente, publicação e/ou direitos autorais.

Observacao: registra alguma observação que o aluno tenha informado durante o preenchimento dos formulários.

Declaração_final: Variável booleana que declara que o aluno leu e está de acordo com todas as informações passadas.

BoardDocument:

Armazena informações sobre a defesa da banca e os conceitos.

Approval: Armazena a situação do aluno. [aprovado/reprovado]

Mark: Armazena o conceito do projeto

Data/location/time/city: Armazena informações sobre onde ocorreu a defesa da banca. Essas colunas são redundantes, pois já são utilizadas no Model Project. Como o código já estava funcionando, optei por não remover.

Tipo: Armazena qual é o tipo de Ata. A opção de Ata automática foi removida do sistema porque não seria viável sua aplicação.

Atadigital: Armazenava a Ata digitalizada com a Gem Paperclip. Devido à problemas com caracteres alfanuméricos e espaços, deixei de utilizar essa gem. Colunas não utilizadas.

Ata_digitalizada: Armazena a Ata digitalizada com a Gem Carrierwave.

Ata_enviada: registra que a Ata foi enviada com sucesso. Criei essa coluna para evitar erros com o download.

Notification

Armazena os avisos que o coordenador deixa para os usuários do sistema

User

Email: Armazena o Email dos usuários.

Encrypted_password: Contém a senha criptografada do sistema.

Reset_password_token: Quando o aluno solicita a recuperação de senha, esse token é gerado com um valor aleatório, enviado por email ao usuário como um link e é usado como confirmação de que a pessoa realmente é solicitou a mudança

Reset_password_sent_at: Armazena a data em que a solicitação de uma nova senha foi gerada

Remember_created_at: A gem permite que o usuário armazene a senha criptografada em um cookie do navegador, para que ele não precise digitá-la sempre que acessa o sistema.

Sign_in_count: Registrador que conta quantas vezes o usuário se locou no sistema

Confirmation_token: Ao registrar no sistema, o token é gerado com um valor aleatório e enviado por email ao usuário como um link. O usuário só terá acesso ao sistema se clicar no link e o token do email corresponder com esse

Confirmed_at: Armazena a data em que o usuário foi confirmado

Confirmed_sent_at: Armazena a data em que o pedido de confirmação foi enviado

Unconfirmed_email: quando é solicitado uma atualização do email do usuário, essa campo armazena temporariamente o novo email e envia um novo token para ele. O email será trocado apenas quando o token for confirmado.

Name: Armazena o nome do usuário

Role: Armazena o tipo de usuário [Aluno, Orientador, Coordenador, Secretaria]

Location: Armazena o local do usuário

RA: Armazena o RA do aluno

Coordenador: Variável booleana que informa ao sistema se ele é ou não um coordenador.

Bloqueado: Informa se o usuário está bloqueado ou não

Duedate

Orientation_start: No projeto antigo não existia o período de matrícula: o aluno já solicitava o pedido de orientação e essa era a data de início desse pedido. Me foi pedido que, antes de solicitar orientador, o aluno deveria fazer a matrícula no PGC. Essa coluna não é mais utilizada.

Orientation_end: Data limite para a entrega do formulário de aceite do pedido de orientação.

Evaluation_start: No projeto antigo, o orientador só poderia preencher a avaliação parcial a partir dessa data, mas o sistema foi atualizado e ele pode enviar a avaliação a qualquer momento. Essa coluna não é mais utilizada.

Evaluation_end: Data limite para a entrega da avaliação parcial do orientador.

Board_start: No projeto antigo o aluno poderia marcar o dia da defesa da banca apenas para entre um intervalo definido. A data de início foi removida. Essa coluna não é mais utilizada.

Board_end: Data limite para envio do formulário de composição da banca examinadora.

Trimestre: Registra informações sobre qual é o perído letivo (1°, 2° ou 3° quadrimestre)

Terminado: Não encontrei nenhuma utilização pelo antigo estagiário dessa coluna.

Finalizado: Informa se o período terminou ou não.

Quantidade_de_atuas: Cada Ata possui um número próprio. Esse registrador armazena o número de Atas que foram geradas no período.

Matricula_inicio: Data de início do período de matrícula.

Matricula_fim: Data de encerramento do período de matrícula.

Prazo_ata: Data limite para o envio da Ata digitalizada pela secretaria.

Prazo_relatorio: Data limite para o aluno enviar o relatório final do projeto.

Prazo_conceito: Data limite para o orientador enviar o conceito, caso o aluno opte por não defender a banca.

Matrícula

Aluno_id: registra o ID do aluno que efetuou a matrícula

Situação: Registra qual é a situação atual do aluno. Estados possíveis:

0 = "Aguardando análise da secretaria" – O aluno acabou de realizar o cadastro

1 = "Aguardando análise do coordenador" – O aluno foi aprovado pela secretaria e aguarda a confirmação do coordenador

2 = "Aluno buscando orientador" – O coordenador já confirmou a análise e o aluno está apto à procurar orientador

3 = "Aguardando análise do coordenador" - O aluno foi reprovado pela secretaria e aguarda a confirmação do coordenador

4 = "Pedido indeferido" – O coordenador já confirmou a análise da secretaria

5 = "Pedido deferido" – O aluno encontrou um orientador

6 = "Não encontrou orientador dentro do prazo" – O pedido foi indeferido porque o aluno não encontrou um orientador dentro do prazo informado pelo coordenador.

Caso seja necessário criar mais estados, atualize a função ‘‘retorna_situação’’, localizada em ‘‘Application_controller’’

Observação: Mensagem que a secretaria deixou para o coordenador durante a análise

Duedate: armazena o ID do período atual

Projects

Orientador_id: armazena o ID do orientador do projeto

Title: Título do projeto

Descricao: Descrição do projeto informada pelo aluno quando fez o pedido de orientação.

Course: Em qual PGC o aluno se matriculou. [PGC I, PGC II, PGC III]

Pdf: Estava no projeto antigo, não encontrei finalidade ela. Essa coluna não é mais utilizada.

Status: informa o estado atual do projeto

0 = "Aguardando aprovação do orientador" – O aluno acabou de criar um projeto novo

1 = "Aguardando aprovação coordenador" – No sistema antigo, o coordenador também deveria aprovar o projeto, mas foi retirado quando o período de matrícula foi criado. Esse estado não é mais utilizado.

2 = "Aguardando avaliação parcial do orientador" – O orientador aceitou o pedido do aluno e o sistema está aguardando o envio do relatório parcial

3 = "Aguardando escolha da avaliação" – O sistema está aguardando o aluno responder se a avaliação do projeto será por banca ou avaliação direta

4 = "Aguardando elaboração/envio do modelo de Ata" – Aguardando o aluno ou orientador gerar o modelo de Ata de enviar à secretaria para digitalização.

5 = "Aguardando envio da Ata digitalizada" -

6 = "Projeto concluido" – O projeto foi terminado com sucesso

-1 = "Recusado pelo orientador" – O orientador recusou o pedido do aluno

-2 = "Recusado pelo coordenador" – Esse estado também não é mais utilizada pelo mesmo motivo descrito acima.

-4 = "Aguardando avaliaçao do projeto" – Não encontrei aplicação para esse estado. Também foi removido

-5 = "Aguardando aprovação da ata pelos membros da banca" – No projeto antigo, o sistema permitia que ele fosse realizado em grupos, porém, como foi necessário armazenar informações de projetos, essa opção foi removida. Estado não utilizado.

-3 = "Aguardando confirmação dos membros do grupo" – Estado não utilizado pelo mesmo motivo descrito acima

-7 = "Reprovado na avaliação parcial" – Informa ao sistema que o aluno não está apto a continuar com o projeto

8 = "Aguardando envio do conceito" – O aluno informou ao sistema que a avaliação será pelo orientador. Aguardando o envio do conceito digitalizado

7 = "Aguardando envio do relatório" - Aguardando o aluno enviar para o sistema o relatório parcial.

9 = "Reprovado" – O aluno foi reprovado na banca

10 = "Aguardando envio do relatório final" - Aguardando envio do relatório final, apenas alunos do PGC I e PGC III

11 = "Reprovado" – Reprovado por algum outro motivo

12 = "Envio do relatório final" – Aguardando envio do relatório final, apenas alunos do PGC I e PGC II

Caso haja necessidade de criar mais campos, atualize as funções ‘‘retorna_status’’, localizadas em ‘‘application_controller’’ e ‘‘application_helper’’.

Observation: Armazena as observações que o orientador informou durante a análise do pedido de orientação.

File: Utilizada no projeto antigo para armazenar arquivos com a Gem Paperclip. Removi essa coluna pois essa Gem gerava problemas caso o arquivo anexado tivesse espaços ou caracteres alfanuméricos. Essa coluna não é mais utilizada

Groupstate: Utilizada no projeto antigo para informar ao sistema que o projeto era em grupo. Essa opção também foi removida.

Board_location: Armazena em que Universidade ocorrerá a defesa da banca.

Board_time: Armazena o horário da defesa.

Numero_da_ata: O número da Ata é gerada automaticamente com a função gera_numero_da_ata, em ApplicationController.

Board_location_2: Armazena em qual sala ocorreu a defesa da banca

Avaliacao: Utilizada no projeto antigo para armazenar arquivos com a Gem Paperclip. Removi essa coluna pois essa Gem gerava problemas caso o arquivo anexado tivesse espaços ou caracteres alfanuméricos. Essa coluna não é mais utilizada.

Relatorio: Armazena o relatório do projeto anexado com a Gem Carrierwave

Form_avaliacao: Armazena o formulário de avaliação anexado com a Gem Carrierwave

Relatorio_enviado: Variável booleana que informa ao sistema que o relatório do projeto com enviado com sucesso. Criei essa coluna porque tive problemas para saber se realmente o arquivo tinha sido anexado.

Defender_banca: Utilizada para informar ao sistema que o aluno optou por não defender a banca.

Tipo_avaliacao: Utilizada para separar quais informações serão mostradas serão mostradas em ProjectsController#show

BoardMember

Instituicao: Utilizado para armazenar de qual Universidade é o membro da banca

Tipo: Informa o tipo de membro [Presidente da Banca, Titular, Suplente].

Project_id: Armazena o ID do projeto do qual o membro está participando

Referências

http://www.mohanarun.com/how-to-install-mysql-adapter-ruby-gem-in-windows/

http://simplesideias.com.br/configurando-ruby-rails-mysql-e-git-no-windows

http://misheska.com/blog/2013/03/10/using-pik-to-manage-multiple-versions-of-ruby-on-windows/

https://github.com/oneclick/rubyinstaller/wiki/Development-Kit

http://stackoverflow.com/questions/19014117/ruby-mysql2-gem-installation-on-windows-7

http://stackoverflow.com/questions/11252199/rails-deployment-environment-on-windows

http://everything-ruby.blogspot.com.br/2011/04/installing-windows-server-2008-ee-with_10.html

http://everything-ruby.blogspot.com.br/2011/04/installing-windows-server-2008-ee-with.html

http://stackoverflow.com/questions/9530554/how-can-i-change-thins-configuration

https://www.digitalocean.com/community/tutorials/how-to-deploy-rails-apps-using-unicorn-and-nginx-on-centos-6-5

http://ruby-on-rails-tutorials.blogspot.com.br/2013/10/how-to-set-up-production-mode-in-rails.html