



3. Rails Básico

x. Cadastro das horas trabalhadas

i. Começamos pelo controlador. time_logs_controller.rb

```
1. class TimeLogsController < ApplicationController
  before_filter :setup_variables
  # GET /time_logs/new
  # GET /time_logs/new.xml
  def new
    @time_log = TimeLog.new
    @time_log.user = @current_user
    @time_log.task_date = Date.today
    respond_to do |format|
      format.html # new.html.erb
      format.xml { render :xml => @time_log }
    end
  end
  # GET /time_logs/1/edit
  def edit
    @time_log = TimeLog.find(params[:id])
    params[:project_id] = @time_log.task_type.project_id
  end
  # POST /time_logs
  # POST /time_logs.xml
  def create
    @time_log = TimeLog.new(params[:time_log])
    respond_to do |format|
      if @time_log.save
        flash[:notice] = 'Registro de trabalho criado com sucesso.'
        format.html { redirect_to :action => "new", :project_id =>
@time_log.task_type.project_id }
        format.xml { render :xml => @time_log, :status =>
:created, :location => @time_log }
      else
        format.html { render :action => "new" }
        format.xml { render :xml => @time_log.errors, :status =>
:unprocessable_entity }
      end
    end
  end
  # PUT /time_logs/1
  # PUT /time_logs/1.xml
```



UNIVERSIDADE FEDERAL DO PIAUÍ
CENTRO DE CIÊNCIAS DA NATUREZA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

```
def update
  @time_log = TimeLog.find(params[:id])
  respond_to do |format|
    if @time_log.update_attributes(params[:time_log])
      flash[:notice] = 'Registro de tempo atualizado com
sucesso.'
      format.html { redirect_to :action => "new", :project_id =>
@time_log.task_type.project_id }
      format.xml { head :ok }
    else
      format.html { render :action => "edit" }
      format.xml { render :xml => @time_log.errors, :status =>
:unprocessable_entity }
    end
  end
end

# DELETE /time_logs/1
# DELETE /time_logs/1.xml
def destroy
  @time_log = TimeLog.find(params[:id])
  @time_log.destroy
  flash[:notice] = 'Registro removido com sucesso'
  respond_to do |format|
    format.html { redirect_to :action => "new" }
    format.xml { head :ok }
  end
end

private
def setup_variables
  @time_logs = TimeLog.find :all, :conditions => {:user_id =>
@current_user.id}, :order => " task_date desc"
  @projects = @current_user.projects
  @task_types = params[:project_id] ?
TaskType.find_all_by_project_id(params[:project_id]) : []
end
end
```

2. show e index foram removidos, assim como as views
3. before_filter chamado setup_variables (edit e new precisam das variáveis)
4. Todo redirect volta pra new
5. Usuário e data atual são configurados previamente



UNIVERSIDADE FEDERAL DO PIAUÍ
CENTRO DE CIÊNCIAS DA NATUREZA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

6. params[:project_id] configurado na view
- ii. Em task_type/new.html.erb
 1.

```
<h1>Novo registro de tempo trabalhado</h1>
<%= render :partial => "form", :locals => {:button_label =>
"Registrar"} %>
<%= render :partial => "past_tasks" %>
```
- iii. Em task_type/edit.html.erb
 1.

```
<h1>Editando registro de tempo trabalhado</h1>
<%= render :partial => "form", :locals => {:button_label =>
"Atualizar"} %>
<%= render :partial => "past_tasks" %>
```
 2. Utilização de partials
- iv. Em task_type/_past_tasks.html.erb
 1.

```
<table width="100%">
<tr>
<th>Data</th>
<th>Tempo</th>
<th>Projeto</th>
<th>Tarefa</th>
</tr>
<% @time_logs.each do |tl| %>
<tr>
<td><%= tl.task_date %></td>
<td><%= tl.worked_hours %></td>
<td><%= tl.task_type.project.name %></td>
<td><%= tl.task_type.name %></td>
<td><%= link_to "Editar",
edit_user_time_log_path(@current_user, tl) %></td>
<td><%= link_to "Remover",
user_time_log_path(@current_user, tl), :method => :delete
%></td>
</tr>
<% end %>
</table>
```
 2. Tabela simples mostrando os registros já cadastrados
- v. Em task_type/_form.html.erb
 1.

```
<% form_for([@current_user, @time_log]) do |f| %>
<%= f.hidden_field :user_id %>
<%= f.error_messages %>
<p>Usuário: <%= @time_log.user.name %></p>
<p>
```



UNIVERSIDADE FEDERAL DO PIAUÍ
CENTRO DE CIÊNCIAS DA NATUREZA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

```
<label for="project_id">Projeto</label>
<select name="project_id" id="project_id">
  <option value=""></option>
  <%= options_from_collection_for_select(@projects, "id",
"name",
  params[:project_id] ? params[:project_id].to_i : nil) %>
</select>
<%= observe_field :project_id,
:url => {:controller => "task_types", :action => "index"},
:frequency => 0.25,
:update => :time_log_task_type_id,
:with => "project_id" %>
</p>
<p>
  <%= f.label :task_type_id, "Tipo de tarefa" %>
  <%= select :time_log, :task_type_id,
@task_types.collect{|tt| [tt.name, tt.id]} %>
</p>
<p>
  <%= f.label :task_date, "Data" %>: <%= f.date_select
:task_date %>
</p>
<p>
  <%= f.label :worked_hours, "Horas Trabalhadas" %>: <%=
f.time_select :worked_hours %>
</p>
<p>
  <%= f.label :description, "Descrição" %>:<br />
  <%= f.text_area :description, :cols => 60, :rows => 5 %>
</p>
<p>
  <%= f.submit button_label %> || <%= link_to "Novo
Registro", new_user_time_log_path(@current_user) %>
</p>
<% end %>
```

2. options do select gerados através de options_from_collection_for_select
3. observe_field é um helper AJAX: atualiza um campo quando outro é alterado
4. Tipo de tarefa: helper select
5. Título do botão submit é uma variável local



6. Para a chamada AJAX funcionar, devemos voltar com o método `index` no controlador de tipos de tarefas
 - a. `def index`

```
@task_types = TaskType.find_all_by_project_id
params[:project_id]
respond_to do |format|
  format.js {
    render :layout => false, :inline =>
"<option></option>"
    <%= options_from_collection_for_select
@task_types, :id, :name, params[:project_id] %>"
  }
end
end
```
 - b. Responde apenas chamadas javascript
 - c. Poderíamos criar uma nova view: `index.js.erb`, mas, como teria só uma linha, optamos pelo `render inline`
- y. Um relatório para a aplicação
 - i. Exemplo de relatório: Horas trabalhadas de determinado membro de projeto em uma determinada tarefa
 - ii. `ruby script/generate scaffold SummaryReport project:references task_type:references user:references`
 - iii. `rake db:migrate`
 1. Código mais limpo e possibilidade de salvar consultas
 - iv. Em `summary_report_controller.rb`
 1.

```
class SummaryReportsController < ApplicationController
  # GET /summary_reports/1
  # GET /summary_reports/1.xml
  def show
    @summary_report = SummaryReport.find(params[:id])
    respond_to do |format|
      format.html # show.html.erb
      format.xml { render :xml => @summary_report }
    end
  end
end
# GET /summary_reports/new
# GET /summary_reports/new.xml
def new
  @summary_report = SummaryReport.new
  @projects = Project.find :all
  @users = User.find :all
```



UNIVERSIDADE FEDERAL DO PIAUÍ
CENTRO DE CIÊNCIAS DA NATUREZA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

```
respond_to do |format|
  format.html # new.html.erb
  format.xml { render :xml => @summary_report }
end
end
# POST /summary_reports
# POST /summary_reports.xml
def create
  @summary_report =
  SummaryReport.new(params[:summary_report])
  respond_to do |format|
    if @summary_report.save
      flash[:notice] = 'SummaryReport was successfully
created.'
      format.html { redirect_to(@summary_report) }
      format.xml { render :xml => @summary_report, :status
=> :created, :location => @summary_report }
    else
      format.html { render :action => "new" }
      format.xml { render :xml => @summary_report.errors,
:status => :unprocessable_entity }
    end
  end
end
end
end
```

2. Mesmo gerado, apenas os métodos index, Edit, destroy e update foram removidos, com as views

v. Em `summary_reports/new.html.erb`

1. `<h1>Novo Relatório</h1>`

```
<% form_for(@summary_report) do |f| %>
```

```
<%= f.error_messages %>
```

```
<p>
```

```
<%= f.label :project_id, "Projeto" %><br />
```

```
<%= f.select :project_id, @projects.collect{|proj|
```

```
[proj.name, proj.id] }, :include_blank => true %>
```

```
</p>
```

```
<%= observe_field :summary_report_project_id, :url =>
```

```
{:controller => "task_types", :action => "index" },
```

```
:frequency => 0.25,
```

```
:update => :summary_report_task_type_id,
```

```
:with => 'project_id'
```



UNIVERSIDADE FEDERAL DO PIAUÍ
CENTRO DE CIÊNCIAS DA NATUREZA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

```
    %>
  <p>
    <%= f.label :task_type_id, "Tipo de Tarefa" %><br />
    <%= f.select :task_type_id, [], :include_blank => true %>
  </p>
  <p>
    <%= f.label :user_id, "Usuario" %><br />
    <%= f.select :user_id, @users.collect{|usr| [usr.name,
usr.id] }, :include_blank => true %>
  </p>
  <p>
    <%= f.submit "Buscar" %>
  </p>
<% end %>
```

2. Criação de novo relatório
3. Helper select através do próprio FormBuilder

vi. Em `summary_reports/show.html.erb`

```
1. <table width="100%">
  <tr>
    <th>Data</th>
    <th>Usuário</th>
    <th>Projeto</th>
    <th>Tipo de Tarefa</th>
    <th>Horas Trabalhadas</th>
  </tr>
  <% @summary_report.report_data.each do |rd| %>
  <tr>
    <td><%= rd[:task_date] %></td>
    <td><%= rd[:user] %></td>
    <td><%= rd[:project] %></td>
    <td><%= rd[:task] %></td>
    <td><%= rd[:total_work] %></td>
  </tr>
  <% end %>
</table>
<%= link_to 'Novo Relatório', new_summary_report_path %>
```

2. Iterando sobre o resultado de um método `report_data` do modelo
 - a. Deve ser criado
 - b. Retorno é um Hash

vii. Em `summary_report.rb`



UNIVERSIDADE FEDERAL DO PIAUÍ
CENTRO DE CIÊNCIAS DA NATUREZA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

1.

```
class SummaryReport < ActiveRecord::Base
  belongs_to :project
  belongs_to :task_type
  belongs_to :user
  def report_data
    filters = []
    user_filter = self.user ? "and usr.id = ?" : ""
    filters << self.user.id if self.user
    task_type_filter = self.task_type ? "and tt.id = ?" : ""
    filters << self.task_type.id if self.task_type
    project_filter = self.project ? "and proj.id = ?" : ""
    filters << self.project.id if self.project
    SummaryReport.find_by_sql [%Q{
      select tl.task_date, usr.name as user, proj.name as project,
      tt.name as task, sum(tl.worked_hours) as total_work
      from time_logs tl
      inner join users usr on tl.user_id = usr.id #{user_filter}
      inner join task_types tt on tl.task_type_id = tt.id
      #{task_type_filter}
      inner join projects proj on tt.project_id = proj.id
      #{project_filter}
      group by task_date, user, project, task
      order by tl.task_date desc
    }, *filters]
  end
end
```
2. Método `find_by_sql`
 - a. Executa um SQL diretamente no banco
 - b. Não é usual precisar dele
- viii. Adicionar link para relatórios no menu
 1. `<%= link_to "Relatório", new_summary_report_path %>`
- z. Limpando um pouco o código e se livrando um pouco do “inglês”
 - i. Textos das views é fácil de corrigir, mas e as URLs?
 - ii. Em `config/routes.rb`
 1. `ActionController::Routing::Routes.draw do |map|
 map.resources :summary_reports, :as => "relatorio_simples",
 :path_names => { :new => "novo", :show => "visualizar" }
 map.root :controller => "projects"
 map.resources :users, :as => "usuarios",
 :path_names => { :new => "novo", :edit => "ver" } do |usr|`



UNIVERSIDADE FEDERAL DO PIAUÍ
CENTRO DE CIÊNCIAS DA NATUREZA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

```
usr.resources :time_logs, :as => "registro_de_horas",  
  :path_names => {:new => "novo", :edit => "ver"}  
end  
map.resources :projects, :as => "projetos",  
  :path_names => {:new => "novo", :edit => "ver"} do |proj|  
  proj.resources :task_types, :as => "tipos_de_tarefas",  
    :path_names => {:new => "novo", :edit => "ver"}  
  proj.resources :project_memberships, :as => "associacao",  
    :path_names => {:new => "nova", :edit => "ver"}  
  end  
end  
map.resources :sessions, :as => "sessao", :path_names =>  
{:new => "nova"}  
map.connect ':controller/:action/:id'  
map.connect ':controller/:action/:id.:format'  
end
```

2. Rotas desnecessárias removidas
3. Necessário parar o servidor e iniciar novamente