```
==================================================
1. Setup e Ambiente
==================================================
<- Bootstrapping:
$ sudo gem install ZenTest
$ autotest

$ sudo gem install rcov
$ rake spec:rcov

$ rake stats

<- testa ordenação por início (lectures_controller_spec)
  it "should find all lectures and sort by start time" do
    Lecture.should_receive(:find).with(:all, :order => :start).and_return([@lecture])
    do_get
  end

<- testa renderização de um partial
  it "should render partial" do
    template.expect_render(:partial => "lecture", :collection => @lectures)
    render "/lectures/index.rhtml"
  end

<- should_receive <=> stub!
<- :null_object => true

==================================================
2. Votação (Migration, Controller e Model)
==================================================

<- adiciona votos:
$ script/generate migration add_vote_column
  add_column :lectures, :votes, :integer, :default => 0
  remove_column :lectures, :votes
$ rake db:migrate

<- Adiciona botões de feedback
  it "should display feedback buttons with links" do
    render "/lectures/index.rhtml"
    response.should have_tag("tr>td>a>img", 4)
  end

  <td><%= link_to image_tag("down.gif"), :controller => "lectures", :action => "vote_for", :id => lecture.id, :vote => -1 %>
    <%=h lecture.votes %>
    <%= link_to image_tag("up.gif"), :controller => "lectures", :action => "vote_for", :id => lecture.id, :vote => 1 %>
  </td>

<- Adiciona coluna nova
  it "should format column headers" do
    (...)
    response.should have_tag("tr>th", "Votos")
  end

<- testa controller com nova action (pending examples...):
  describe LecturesController, "handling PUT /lectures/vote_for/1?vote=42" do
    before(:each) do
      @lecture = mock_model(Lecture, :null_object => true)
    end

    it "should find lecture by id" do
      Lecture.should_receive(:find).and_return(@lecture)
      put :vote_for, :id => @lecture.id, :vote => "42"
    end

    it "should update the lecture vote"
    it "redirect to the lectures list"
  end

  it "should update the lecture vote" do
    @lecture.should_receive(:account_vote).with(42)
    put :vote_for, :id => @lecture.id, :vote => "42"
  end
  Lecture.stub!(:find).and_return(@lecture)

  it "redirect to the lectures list" do
    put :vote_for, :id => @lecture.id, :vote => "42"
    response.should redirect_to(lectures_url)
  end

<- testa model com método novo
  it "should account votes" do
    @lecture.account_vote(2)
    @lecture.votes.should == 2

    @lecture.account_vote(1)
    @lecture.votes.should == 3

    @lecture.account_vote(-2)
    @lecture.votes.should == 1
  end

  def account_vote(vote)
    self.votes += vote
  end

<- controller deve atualizar o BD
```

```
  it "should update the lecture vote" do
    (...)
    @lecture.should_receive(:update).and_return(true)
    (...)
  end

==================================================
3. Highlight da Palestra Atual (View, Helper, Model)
==================================================

<- testa highlight num Helper
  :view should call highlight
    it "should highlight lecture" do
      template.should_receive(:highlight).twice.and_return("")
      render "/lectures/index.rhtml"
    end
    '<%= highlight(Time.now, lecture) %>'
    BUG: criar método vazio no helper

  :helper should call is_happening?
    before(:each) do
      @lecture = mock_model(Lecture)
    end

    it "should highlight lecture happening now" do
      @lecture.should_receive(:is_happening?).with(Time.parse("13:00")).and_return(true)
      highlight(Time.parse("13:00"), @lecture).should == "class=\"highlighted\""
    end

    it "should not highlight lecture not happening" do
      @lecture.should_receive(:is_happening?).with(Time.parse("13:00")).and_return(false)
      highlight(Time.parse("13:00"), @lecture).should be_empty
    end

    def highlight(now, lecture)
      lecture.is_happening?(now) ? "class=\"highlighted\"" : ""
    end

  :model should implement is_happening? (TDD step-by-step -- pending)
    before(:each) do
      (...)
      @lecture.start = Time.parse("13:00")
      @lecture.end = Time.parse("14:00")
    end

    it "should happen between start and end" do
      @lecture.is_happening?(Time.parse("13:30")).should be_true
    end

    it "should happen at start" do
      @lecture.is_happening?(Time.parse("13:00")).should be_true
    end

    it "should not happen before start" do
      @lecture.is_happening?(Time.parse("12:59:59")).should_not be_true
    end

    it "should not happen at end" do
      @lecture.is_happening?(Time.parse("14:00")).should_not be_true
    end

    it "should not happen after end" do
      @lecture.is_happening?(Time.parse("14:00:01")).should_not be_true
    end

    def is_happening?(now)
      (self.start...self.end).include? now
    end

==================================================
4. Transforma em AJAX
==================================================

<- Adiciona biblioteca Prototype (requisição AJAX)
  '<%= javascript_include_tag :defaults %>'

  :view Adiciona link_to_remote na view
    it "should link to remote" do
      render "/lectures/index.rhtml"
      response.should have_tag("a[href='#']", 4)
    end

    <td><%= link_to_remote image_tag("down.gif"), :url => {:controller => "lectures", :action => "vote_for", :id => lecture.id, :vote => -1}
%>
      <%=h lecture.votes %>
      <%= link_to_remote image_tag("up.gif"), :url => {:controller => "lectures", :action => "vote_for", :id => lecture.id, :vote => 1} %>
    </td>

  :view Adiciona span c/ id nos votos
    it "should wrap votes with span" do
      render "/lectures/index.rhtml"
      response.should have_tag("span#vote_count_#{@lectures[0].id}")
      response.should have_tag("span#vote_count_#{@lectures[1].id}")
    end

    '<span id="vote_count_<%=h lecture.id %>"><%=h lecture.votes %></span>'
```

```
  :controller Trata requisição AJAX no controller
    it "should accept AJAX request" do
      xhr :put, :vote_for, :id => @lecture.id, :vote => "42"
      response.should have_rjs(:replace_html, "vote_count_#{@lecture.id}")
    end

    respond_to do |format|
      format.html { redirect_to lectures_url() }
      format.js {
        render :update do |page|
          page.replace_html "vote_count_#{@lecture.id}", @lecture.votes
          page.visual_effect :highlight, "lecture_#{@lecture.id}"
        end
      }
    end

  :view Adiciona id no partial
    it "should have id attribute on lecture row (tr) " do
      render "/lectures/index.rhtml"
      response.should have_tag("tr#lecture_#{@lectures[0].id}")
      response.should have_tag("tr#lecture_#{@lectures[1].id}")
    end

    'id="lecture_<%= lecture.id%>"'

  :controller implementa highlight

$ rake spec:rcov
$ rake stats
$ rake spec:reports (rspec_additions.rake)
```