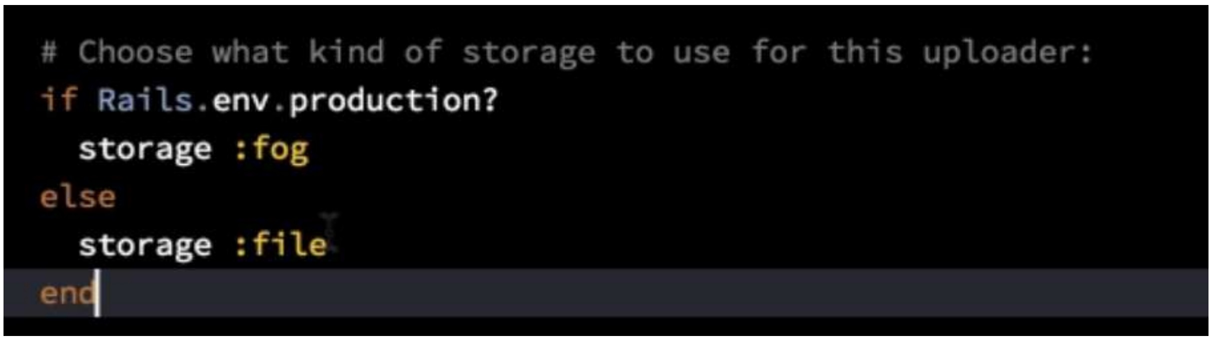
314. Image Upload in Production

Use a storage system depending on environment



```
# Choose what kind of storage to use for this uploader:
if Rails.env.production?
  storage :fog
else
  storage :file
end
```

AWS

Create user

Dashboard->users->create user
Save and download credentials.

Create S3 bucket

Create and give it a name.

Give user access to bucket

policies -> create
Select an already existing policy, **Administrator access**
Make some changes to the policy.
Create the policy.

Go to users, and select user -> attach policy.

Policy created:

{

"Version": "2012-10-17",

"Statement": [

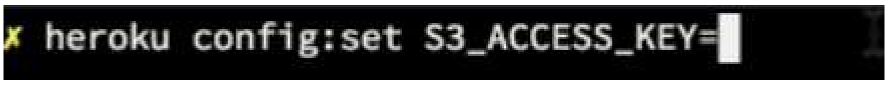
{

"Effect": "Allow",

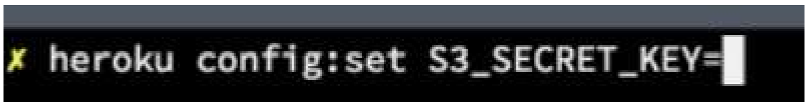
```
"Action": "s3:*",

"Resource": [

"arn:aws:s3:::yours3bucketname",

"arn:aws:s3:::yours3bucketname/*"

]

},

{

"Effect": "Allow",

"Action": "s3:ListAllMyBuckets",

"Resource": "arn:aws:s3:::*"

}

]

}
```

316. Complete Prod Image Upload

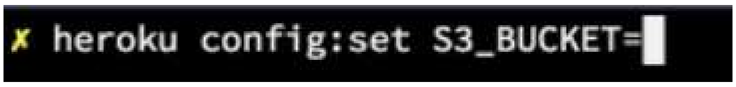
Configure AWS credentials on Heroku



```
✗ heroku config:set S3_ACCESS_KEY=
```



```
✗ heroku config:set S3_SECRET_KEY=
```

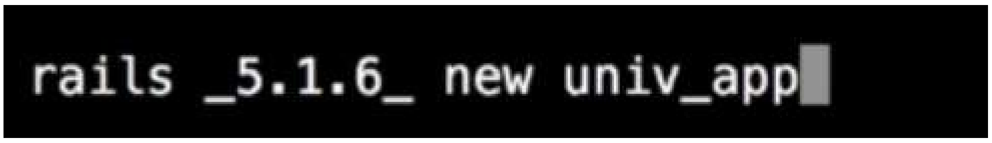


```
✗ heroku config:set S3_BUCKET=
```

University app

379. Start new rails app, run server - both local and cloud-IDE example

Use a specific version of rails to create app



```
rails _5.1.6_ new univ_app
```

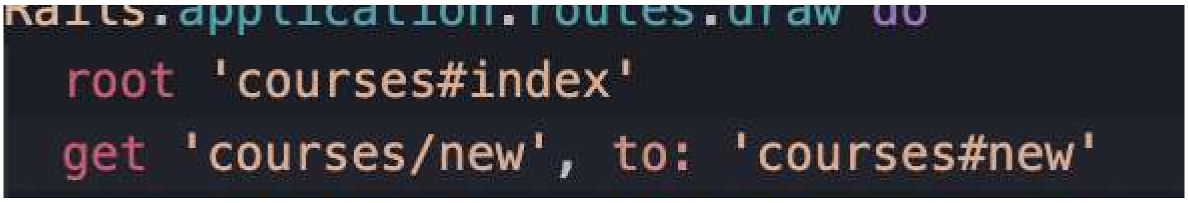
380. App structure, MVC and root route

Naming conventions

- Controller file name -> courses_controller.rb, naming convention -> snake_case (lower case) and plural
- Class name: class CoursesController < ApplicationController
- Actions inside the controller -> we can call them whatever we want. Some names are used by the default routing from rails.
- Views
    - It has to be name with the initial part of the controller name
        - courses_controller -> courses
- View file
    - To connect action with a view, we should add it to the corresponding directory and add the action name into it´s name. -> index.html.erb

382. Add routes

We can create each route by yourself (new, update, create, etc), but there is a better way.



```
Rails.application.routes.draw do
  root 'courses#index'
  get 'courses/new', to: 'courses#new'
```

383. Explore the layout file and erb

Application.html.erb

It is the layout file of the application.
Other pages are being wrapped inside this page.
It is a good place to add navigation, footer, etc.

384. Start styling the app

385. Add front-end framework and navigation

Installation instructions:

https://github.com/mkhairi/materialize-sass

Application.css

It is a manifest file, meaning that any file that is inside the styles directory is going to be served through it and be available to our application.

Application.js

Similar to Application.css

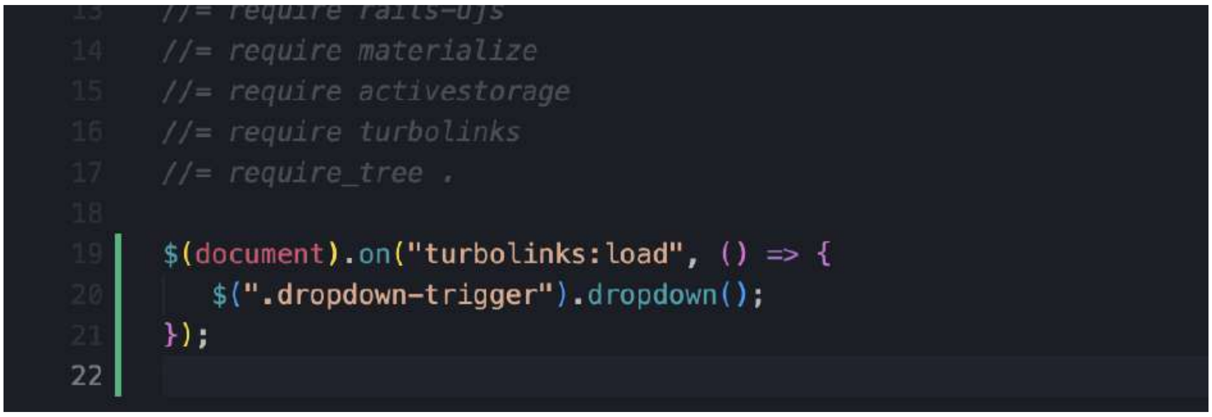
Installation of material icons

https://github.com/Angelmmiguel/material_icons

386. Create a sticky footer

387. Add dropdown feature and hamburger menu

We can add code into our application.js if we want.

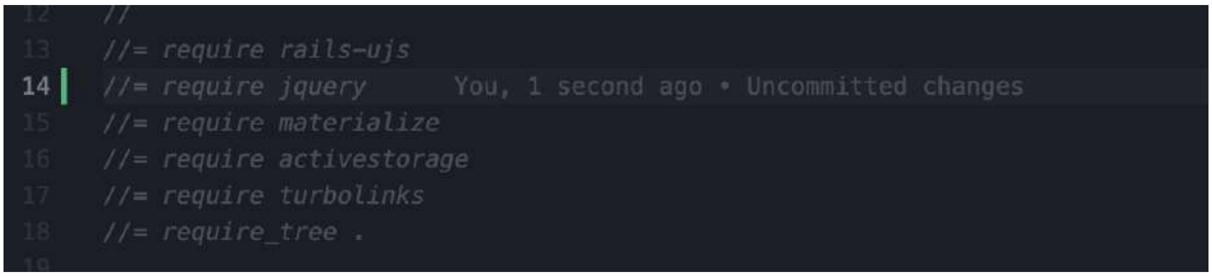


```
//= require rails-ujs
//= require materialize
//= require activestorage
//= require turbolinks
//= require_tree .

$(document).on("turbolinks:load", () => {
    $(".dropdown-trigger").dropdown();
});

```

Add JQUERY

Add gem:
gem 'jquery-rails'

Now we should add a require into our application.js



```
//
//= require rails-ujs
//= require jquery        You, 1 second ago • Uncommitted changes
//= require materialize
//= require activestorage
//= require turbolinks
//= require_tree .
```

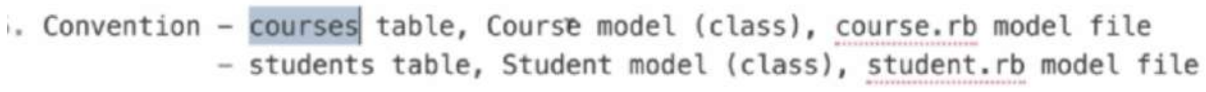
Fix jquery and turbo link error not letting to open the navbar twice

https://github.com/Dogfalo/materialize/issues/5396
https://stackoverflow.com/questions/18769109/rails-4-turbo-link-prevents-jquery-scripts-from-working

389. CRUD from the back-end and courses resource

Ruby creates for you a primary key on every table

Rails database conventions

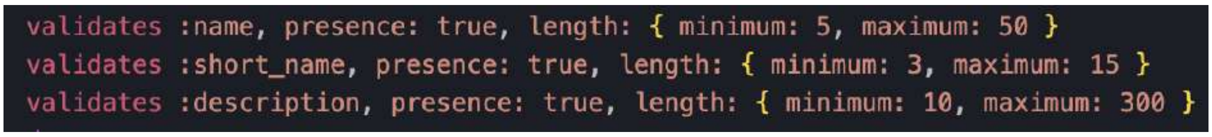


```
. Convention - courses table, Course model (class), course.rb model file
             - students table, Student model (class), student.rb model file
```

Migrations

If we start the name of a migration with “create”, it will add the code to create a table.

Validations



```ruby
validates :name, presence: true, length: { minimum: 5, maximum: 50 }
validates :short_name, presence: true, length: { minimum: 3, maximum: 15 }
validates :description, presence: true, length: { minimum: 10, maximum: 300 }
```

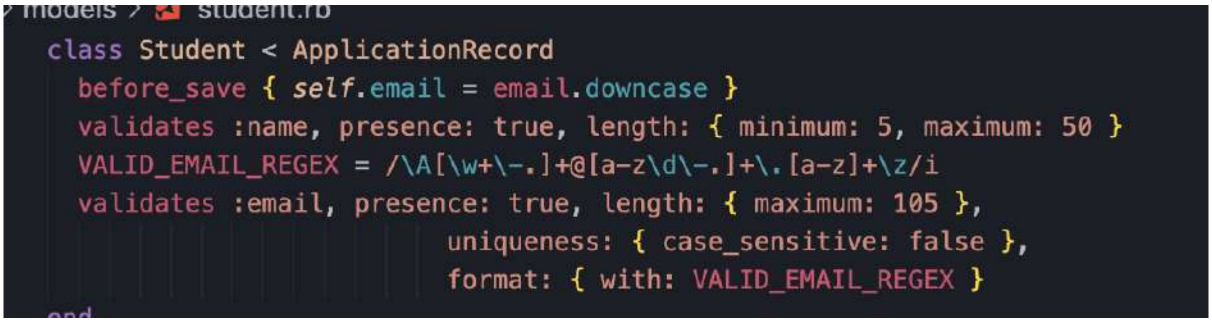
391. Learn how to use partials

They organize better the html documents by dividing the code into sections that can be called.

Their name start with “_”

392. Create students resource

Email validation



```ruby
models > student.rb
class Student < ApplicationRecord
  before_save { self.email = email.downcase }
  validates :name, presence: true, length: { minimum: 5, maximum: 50 }
  VALID_EMAIL_REGEX = /\A[\w+\-.]+@[a-z\d\-.]+\.[a-z]+\z/i
  validates :email, presence: true, length: { maximum: 105 },
                    uniqueness: { case_sensitive: false },
                    format: { with: VALID_EMAIL_REGEX }
end
```

Ruby testing ground for regular expressions:

https://rubular.com/

394. Students index

Forms helpers on rails

Form tag

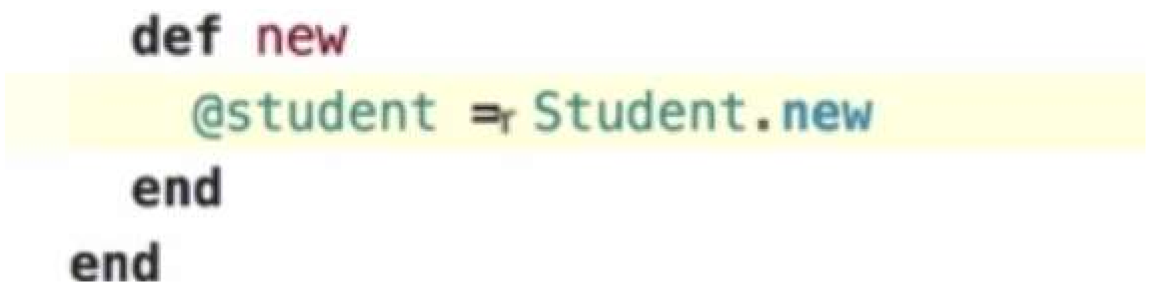
Standard form

Form for

form aiming for a resource

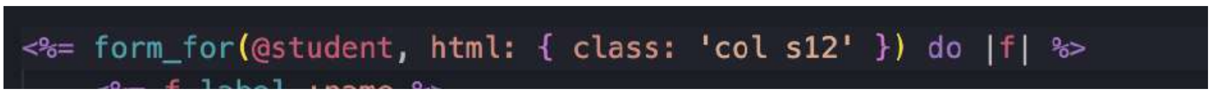
395. Create a new student form

We create an object so we can give it to the form, so it can have the name or any other attribute.



```
  def new
    @student = Student.new
  end
end
```

Add classes to rails helper form



```
<%= form_for(@student, html: { class: 'col s12' }) do |f| %>
```

396. Create students and work with errors

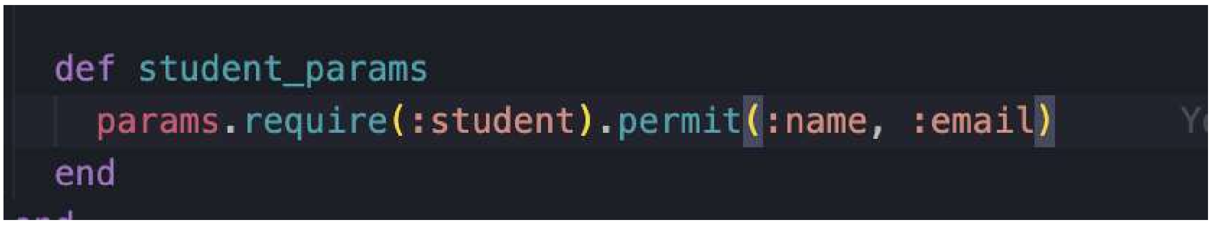
The form helper automatically knows where to submit the data, since we are on new page, it will submit it to the create action.

Get form information

We can do mass assignment, we need to get the data and white-list it.

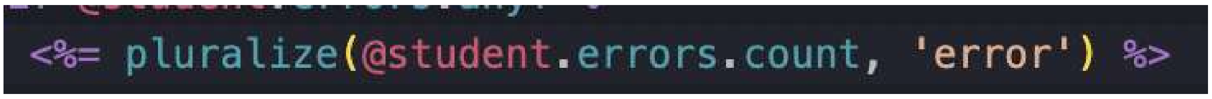
To white list information we create a method, this method will return the data ready to use in our model.

On the requiere we need to pass the top level key, and then on the permit we pass the attributes.



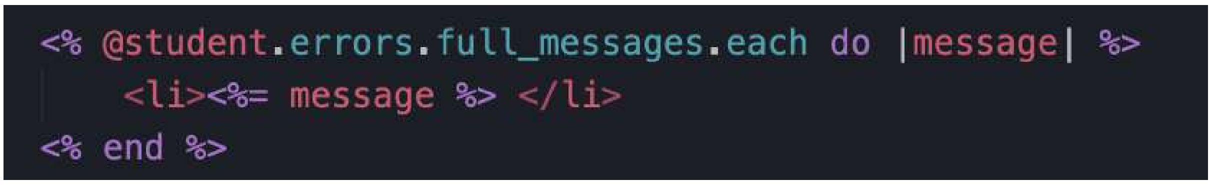
```ruby
def student_params
  params.require(:student).permit(:name, :email)
end
```

Pluralize



```erb
<%= pluralize(@student.errors.count, 'error') %>
```

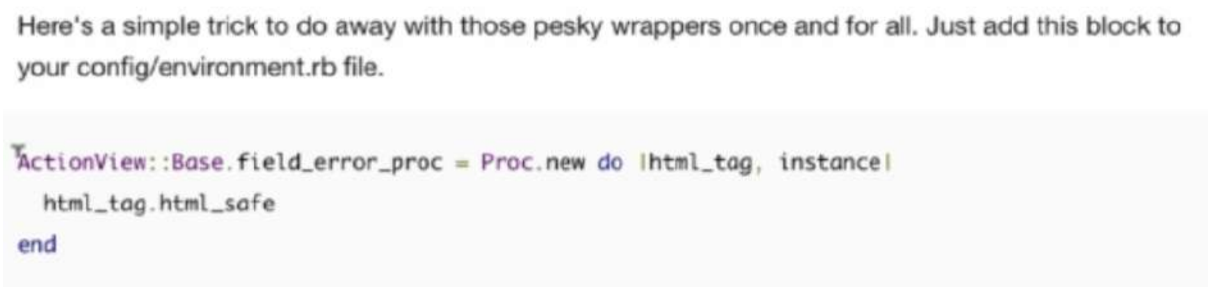
Get full message error



```erb
<% @student.errors.full_messages.each do |message| %>
    <li><%= message %> </li>
<% end %>
```

Disable rails input errors container

You need to add the next code into the config file

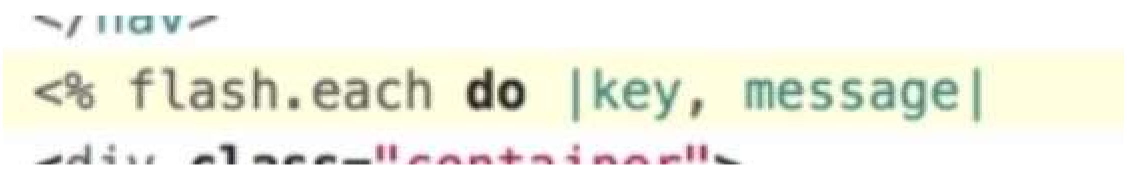


Here's a simple trick to do away with those pesky wrappers once and for all. Just add this block to your config/environment.rb file.

```ruby
ActionView::Base.field_error_proc = Proc.new do |html_tag, instance|
  html_tag.html_safe
end
```

Every time you update enviroment.rb is recommended to restart the server

Iterate over the flash messages

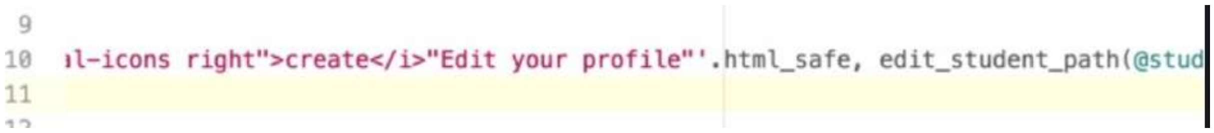


```erb
</nav>
<% flash.each do |key, message|
```

398. Student show page

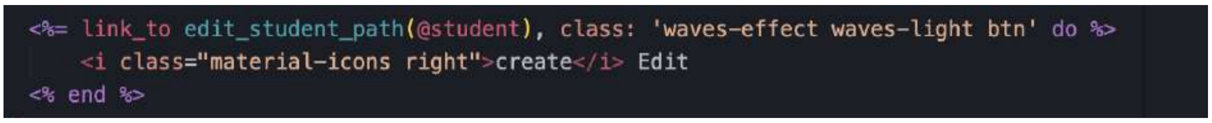
Add html inside link_to

One way is adding inside the text area, the html code plus ".html_safe"



```
ıl-icons right">create</i>"Edit your profile"'.html_safe, edit_student_path(@stud
```

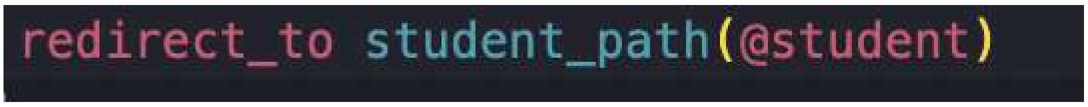
Another way is adding a block at the end of the statement



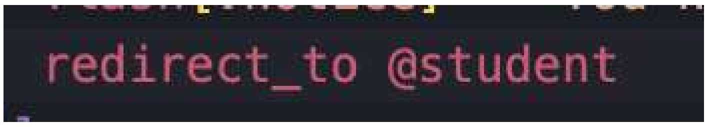
```
<%= link_to edit_student_path(@student), class: 'waves-effect waves-light btn' do %>
    <i class="material-icons right">create</i> Edit
<% end %>
```

Redirect to show page

Both ways work:



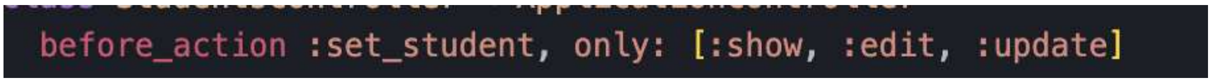
```
redirect_to student_path(@student)
```



```
redirect_to @student
```

Before action

Call a method before the actions, you can omit actions as well



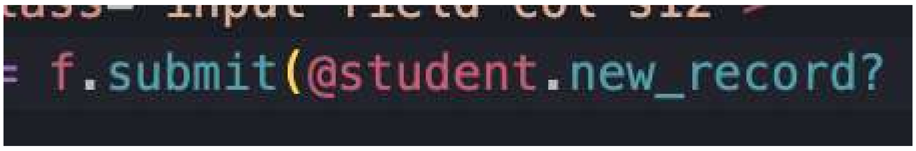
```
before_action :set_student, only: [:show, :edit, :update]
```

Variables and partials

Class variables created on the controller will be available inside renderer elements.

new_record?

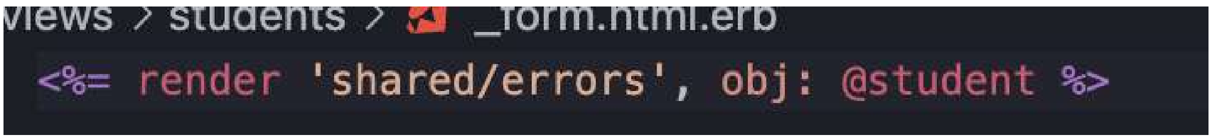
You can check if a model object is new using this method, useful to take different actions on update and new pages.



```
f.submit(@student.new_record?
```

https://stackoverflow.com/questions/7842920/determine-if-activerecord-object-is-new

Pass parameters to partial



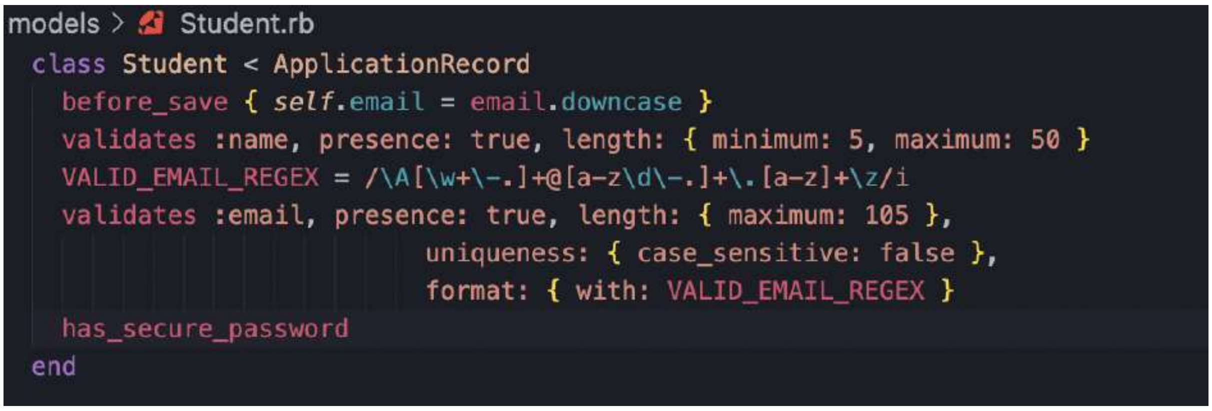
```
views > students > _form.html.erb
<%= render 'shared/errors', obj: @student %>
```

401. Add secure password

To use the default rails password system we need to call the password field "password_digest"

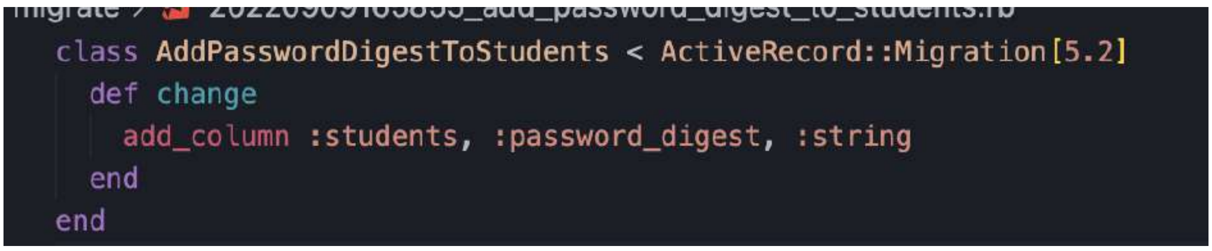
Also we need a bcrypt gem.

Now we symple add "has_secure_password" on user model.



```
models > Student.rb
class Student < ApplicationRecord
  before_save { self.email = email.downcase }
  validates :name, presence: true, length: { minimum: 5, maximum: 50 }
  VALID_EMAIL_REGEX = /\A[\w+\-.]+@[a-z\d\-.]+\.[a-z]+\z/i
  validates :email, presence: true, length: { maximum: 105 },
                    uniqueness: { case_sensitive: false },
                    format: { with: VALID_EMAIL_REGEX }
  has_secure_password
end
```

Add password_digest column to database

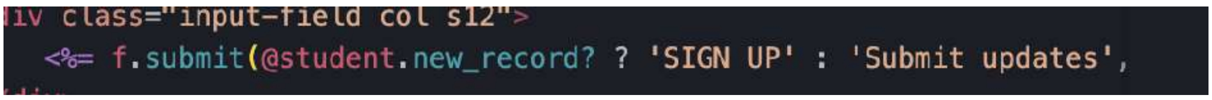


```
migrate > 20220909165835_add_password_digest_to_students.rb
  class AddPasswordDigestToStudents < ActiveRecord::Migration[5.2]
    def change
      add_column :students, :password_digest, :string
    end
  end
```

And you are ready to go.
Do not update your password using "password_digest", use password

402. Update forms to accept passwords and modify styling

Fix submit button click detection



```
iv class="input-field col s12">
  <%= f.submit(@student.new_record? ? 'SIGN UP' : 'Submit updates',
```

Instead of using f.submit, you can also use f.button. This will fix it.

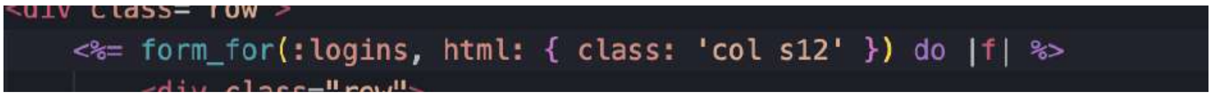
403. Authentication system - build routes and form

Sesiones

If the user logs in, then rails store the user id and other data on a 32 character string, this string is stored into a cookie.
So when the user visits certain parts of our application, rails can use it to authenticate the user.
When the user logs out, that session id becomes nil.

You can call the controller that manages the logins whatever you want. On previous videos it was “sessions”, this time he is going to use logins.

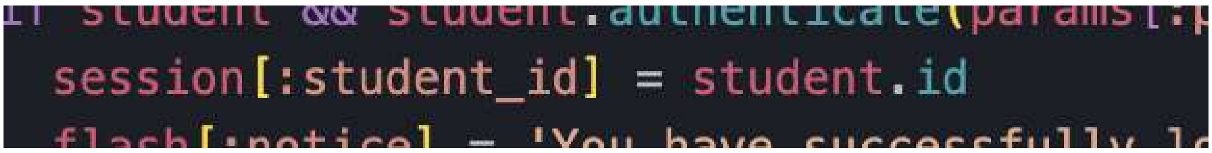
Form for a general models instead a specific object



```
<div class="row">
  <%= form_for(:logins, html: { class: 'col s12' }) do |f| %>
    <div class="row">
```

404. Create and destroy sessions - add auth methods

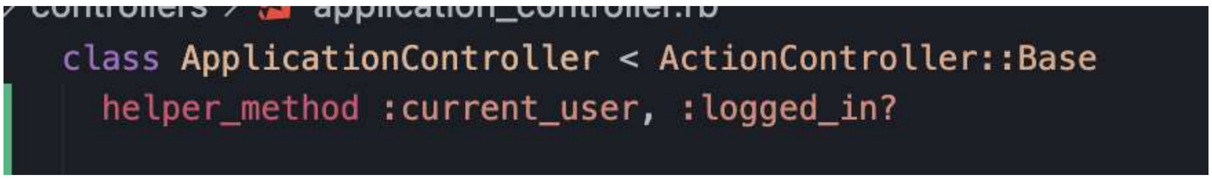
Save id into session



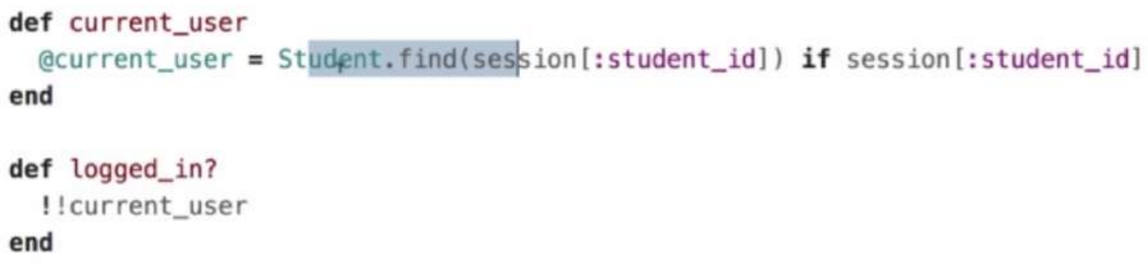
```
if student && student.authenticate(params[:p
  session[:student_id] = student.id
  flash[:notice] = 'You have successfully lo
```

Login method Location

We can add the method to know if a user is logged in on the helper directory.
But if we add it to the helper directory, then only will be available to views.
If we want it to also be available on the controller, we can add it into our ApplicationController.
If you want a method from application_controller to be used on views as well, then declared them into this method:



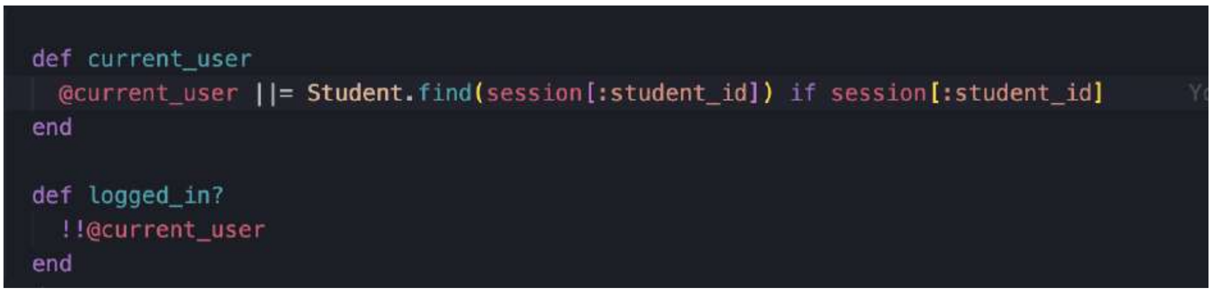
```
controllers > application_controller.rb
class ApplicationController < ActionController::Base
  helper_method :current_user, :logged_in?
```



```ruby
def current_user
  @current_user = Student.find(session[:student_id]) if session[:student_id]
end

def logged_in?
  !!current_user
end
```

Memorization
We are going to improve performance by avoiding loading the user from the database on each method call.

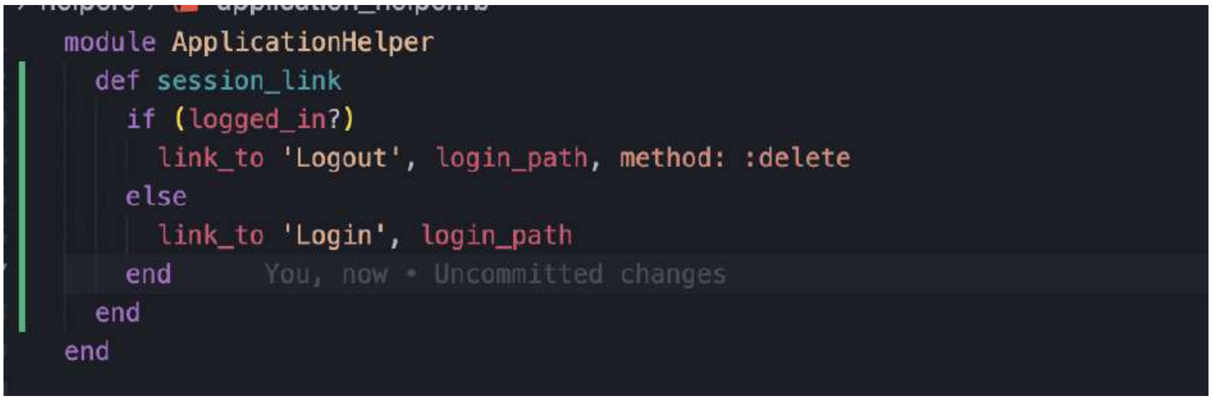
The two pipe characters are for memorization
It will only run the code on the right if the variable does not exist.



```ruby
def current_user
  @current_user ||= Student.find(session[:student_id]) if session[:student_id]
end

def logged_in?
  !!@current_user
end
```

Helper

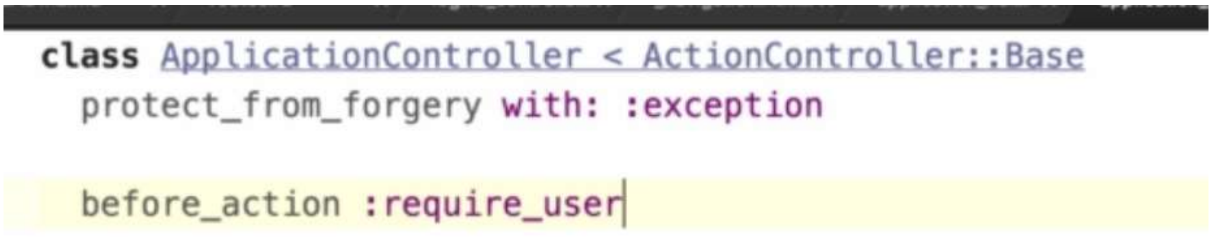
We define a helper (code used in views) to get the sign up link or the log out link.



```ruby
module ApplicationHelper
  def session_link
    if (logged_in?)
      link_to 'Logout', login_path, method: :delete
    else
      link_to 'Login', login_path
    end      You, now • Uncommitted changes
  end
end
```

405. Restrict actions, views and clean up layout

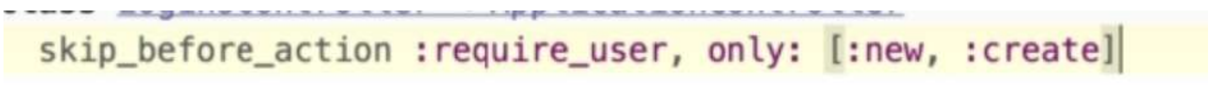
Restriction on all controllers



```ruby
class ApplicationController < ActionController::Base
  protect_from_forgery with: :exception

  before_action :require_user
```

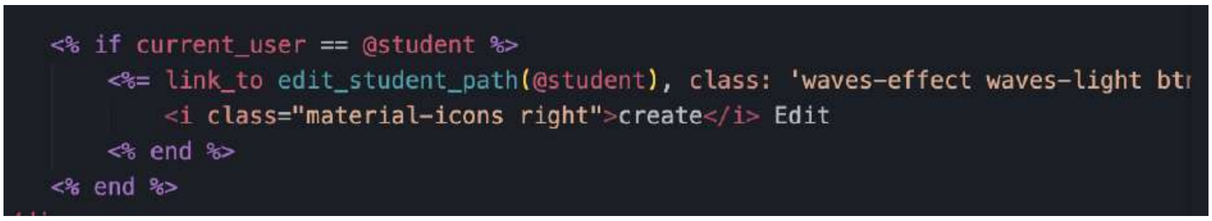
If you add a before action on ApplicationController, since all other controllers inherit from it, all of them will have the same restriction.

Skip before action



```ruby
skip_before_action :require_user, only: [:new, :create]
```

Restrict user to edit only his own profile



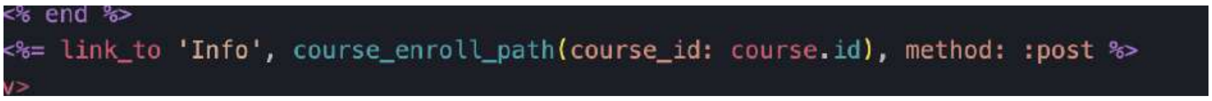
```erb
<% if current_user == @student %>
    <%= link_to edit_student_path(@student), class: 'waves-effect waves-light bt
        <i class="material-icons right">create</i> Edit
    <% end %>
<% end %>
```

406. Introduction to many to many associations

407. Create association from rails console

408. Add associations from front-end

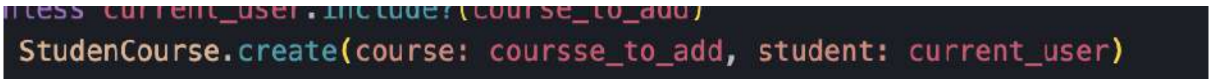
Pass parameters on link_to method



```erb
<% end %>
<%= link_to 'Info', course_enroll_path(course_id: course.id), method: :post %>
v>
```

Create object or id

On the create method of a model that needs an id of another model, you can pass the object and rails is intelligent enough to grab the id from the object.



```
less current_user.include?(course_to_add)
 StudenCourse.create(course: coursse_to_add, student: current_user)
```

redirect_to and object

Similarly you can pass an object to redirect_to and rails will redirect to the show page of that object.