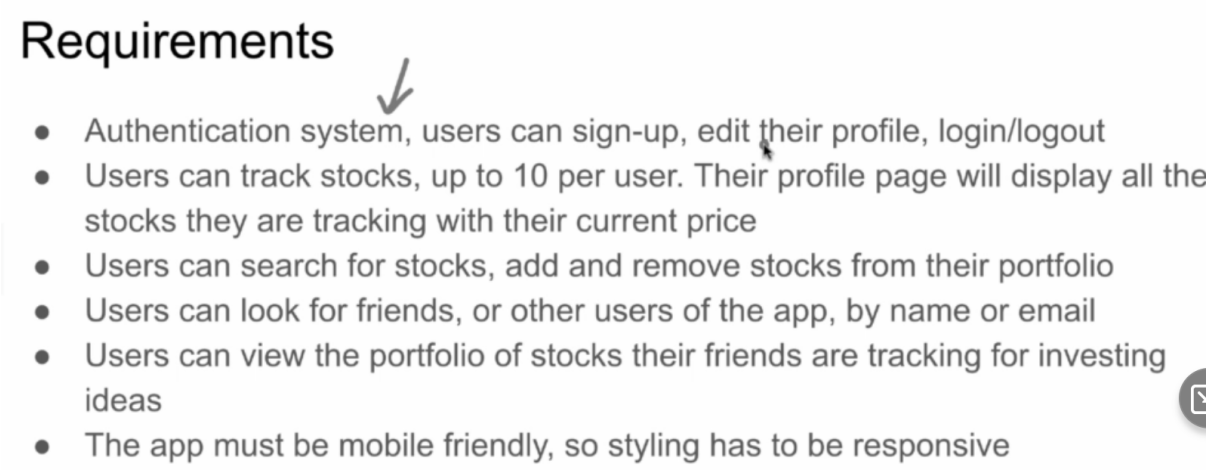
Udemy course complete project link:
https://github.com/udemyrailscourse/finance-tracker-6

241. Finance Tracker Requirements



245. Assignment 2 completion

Controller generator

Index is the action
rails generate controller welcome index

It also creates views and test modules

Install development gems

Bundle install --without production

If you are in production, it is important to have gem.lock updated after a bundle install.

246. Add devise gem for authentication

In the course, we are going to use a gem “device” that handles auth.

- Sign up
    - hashed password
    - confirmation
    - forgot password
    - remember user (so you don't login each time you open de app)

- login
- logout

To learn a complex gem like this one, you should learn and get good with the basics and then go for more complex features.

After adding the gem and using bundle install
$ rails generate devise:install

You can create login and sign up views using

But we are going to create the views using a bootstrap gem to create our views.

247. Create users using devise

Steps below are using the documentation from github

Create model user

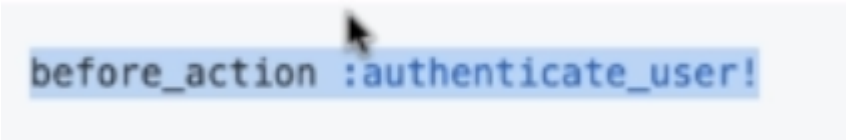
rails generate devise MODEL

```ruby
class User < ApplicationRecord
  # Include default devise modules. Others available are:
  # :confirmable, :lockable, :timeoutable, :trackable and :omniauthable
  devise :database_authenticatable, :registerable,
         :recoverable, :rememberable, :validatable
end
```

- Recoverable
  - Forget my password feature
- rememberable
  - User session stays
- registrable
  - User can registrate

Set controller with user auth

Added on application_controller (if you want auth on every route)



248. Test authentication system, login, logout

See and use devise routes

rails routes | grep user

249. Assignment: Add Bootstrap to the application

Bootstrap install
https://www.holaruby.com/bootstrap_rails

Recordatorio
La siguiente línea agrega todos los estilos de la carpeta

```
*= require_tree .
```

251. Update views

Use gem to automate form styling

Note that
Login and signup views are not on a file under views, they are available through device engineer.
The views are inside the gem

Get devise views into views directory to customize them.

```
$ rails generate devise:views
```

Pero en lugar de usar las que devise usa, usaremos una gem llamada
devise-bootstrap-views
So we have something pretty fast.
Read documentation to install it.

Devise views into directory but with bootstrap

rails generate devise:views:bootstrap_templates

253. Layout Assignment: Add messages and nav partial

Devise gives us two methods that are like flash messages, notice and **alert**.

These two messages do not have styles but model validation messages have.

254. Setup and use API key to get stock data

…

255. Create Stock model with attributes

Generator for model Stock: (generates migrations as well)
rails g model Stock ticker:string name:string last_price:string

It is not recommended to use floats for prices because it is less precise.
Use decimal

256. Stock lookup: build class method to lookup stock info

Added ‘new_lookup’ method to model to get fast information about stocks

Important to not commit any key into the repository.

257. Secure credentials in Rails 6

In the past we used environmental variable to save keys, we can still use them **but rails have a cleaner way to store them**

On **config** directory there is

- credentials.yml.enc
    - Encrypted yaml file with credentials
- master.key
    - It contains the necessary information to decrypt the credentials file

The credential file can be pushed to github, but the master.key file is not.
We should send the master.key file to your teammates in another way.

If you pull a github repository that stores the keys in a similar manner, you should create your own files.

To edit that file you use:
rails credentials:edit
you should set and editor as well

```
 edit.
$ EDITOR=code rails credentials:edit
```

It wont work because vs code opens the file async so the terminal things that you already finished editing.
To fix it, you should do

```
File encrypted and saved.
$ EDITOR="code --wait" rails credentials:edit
```

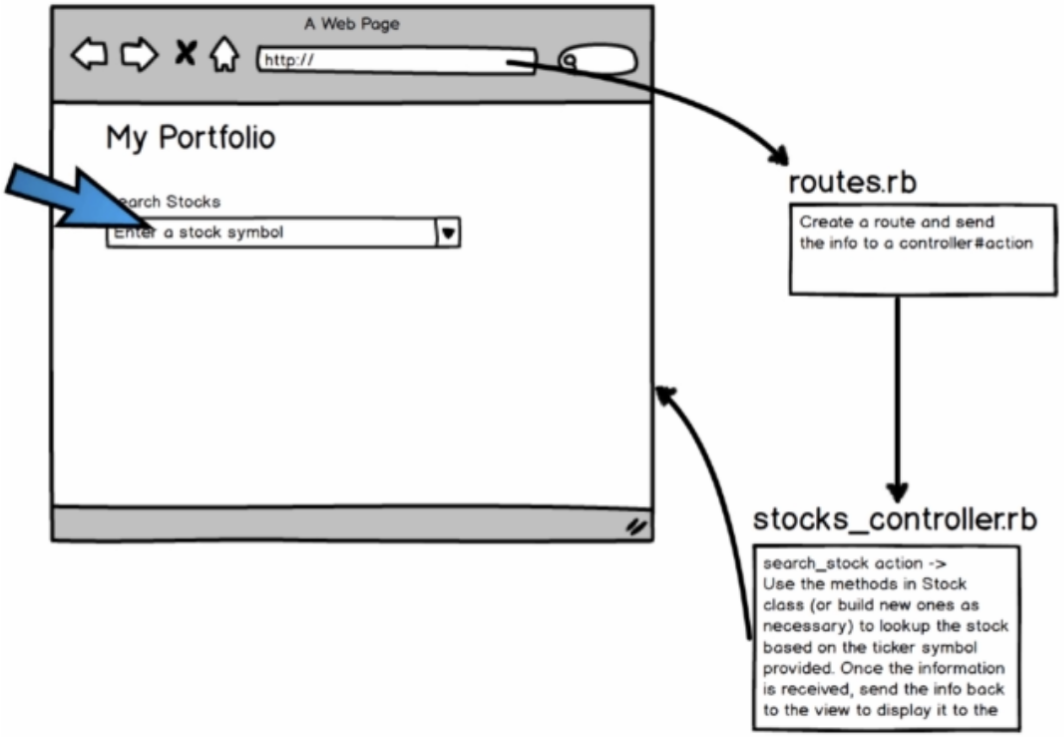
Access credentials

```
Rails.application.credentials.aws[:access_key_id]
```

258. Store secure API key

…

259. Setup front-end structure for stock lookup



Generator for controllers

rails g controller users my_portfolio
It generates another route but we can change it on the Routes file.

260. Build Stock Lookup Form

text_field_tag is an standard input
Parameters:

1. name
2. content -> params -> you recover with params the data you send to the server

Icons

You can use “font awesome gem”.

262. Create and display stock objects in browser

263. Dealing with invalid search results

Error management

```
begin
  response = client.quote(ticker_symbol)
  response.to_a
  new(ticker: ticker_symbol, name: response["company_name"], last_price: response["i
rescue => exception
  return nil
end
```

264. Use Ajax for form submission

Add **remote: true** to the form to make the requests ajax.

```
<%= form_tag search_stock_path, remote: true, method: :get
```

.present? -> check if the hash contains a key.

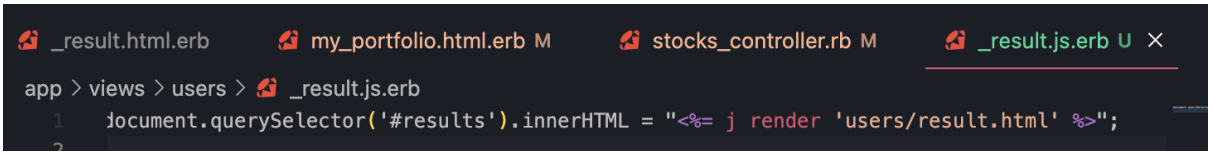
265. Setup JavaScript response

Send js partial

We should create a file with this format: “**[name.js.erb]**”

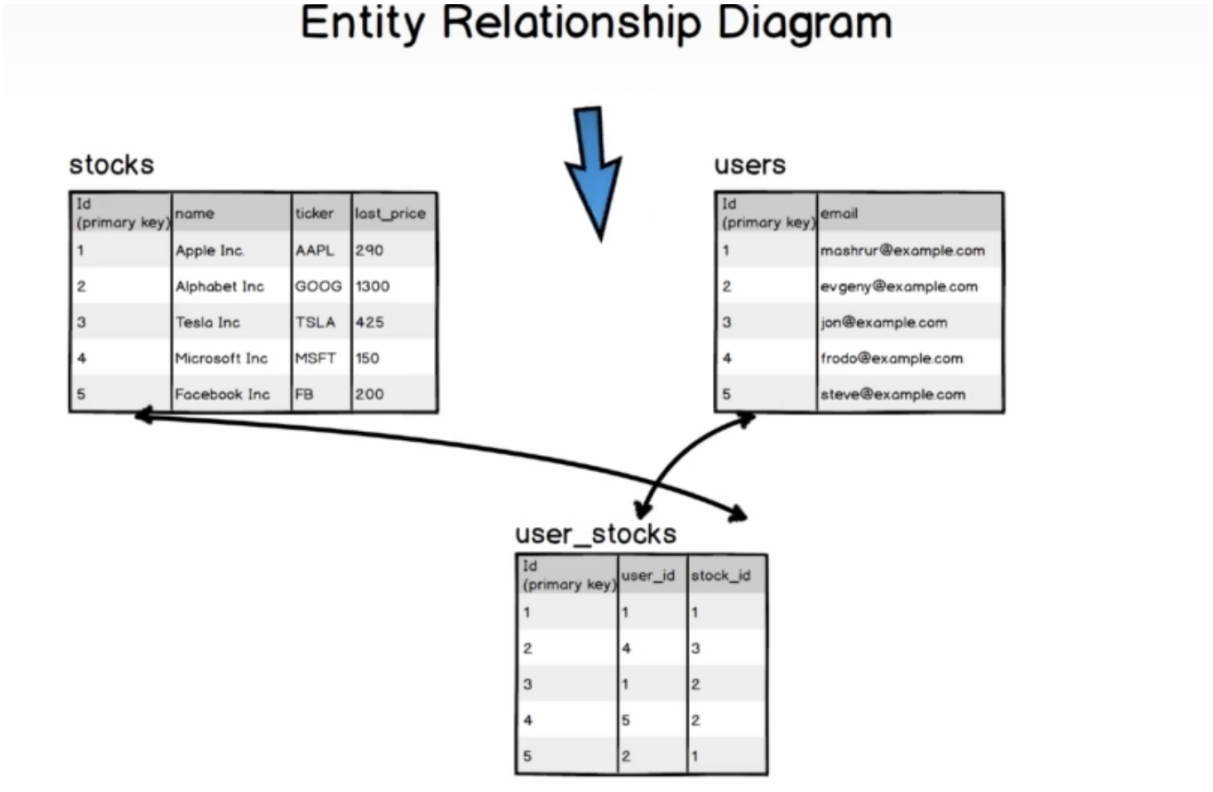
```
respond_to do |format|
  format.js { render partial: 'users/result' }
end
```

If we want to send a javascript from the server, and we want to include a render to a html element inside this js, we should then escape all characters that can cause an error inside our js file using “j”



http://www.carlosramireziii.com/when-to-use-escape-javascript-in-an-sjr-template.html

267. Users and stocks: many-to-many association



268. Setup UserStock resource

Crear recurso completo
rails generate resource UserStock

Create references to other tables
rails generate resource UserStock user:references stock:references

Thanks to these values we already have the association setup on the model.

```ruby
class UserStock < ApplicationRecord
  belongs_to :user
  belongs_to :stock
end
```

Better route view

```
$ rails routes --expanded | grep user_stocks
```

Check current path

```
<li class="nav-item <%= 'active' if request.path == my_portfolio_path %>">
```

271. Track stocks from front-end: browser

Select only a few routes from resources

routes.rb

```ruby
Rails.application.routes.draw do
  resources :user_stocks, only: [:create]
```

Add parameters to link_to

```
<%= link_to 'Add to my portfolio', user_stock_path(user: current_user), ticker: @stock.t
                                   class: 'btn btn-success', method: :post %>
```

Active record: relation

https://stackoverflow.com/questions/38325538/difference-between-activerecord-and-activerecordrelation-objects

273. Add functionality to remove tracking

Where example

```
..]>
 UserStock.where(user_id: user.id, stock_id: stock.id)
```

You can use first to transform it to an active row object,

274. Modify user model

275. Accept additional fields in app - edit action

Two ways to use link_to

Regular call to link_to

```
<%= link_to 'Edit profile', edit_user_registration_path, class: 'dropdown-item' %>
                            <a class="dropdown-item" href="#">Another action</a>
```

Link_to with html code inside

```
<%= link_to edit_user_registration_path, class: 'dropdown-item' do %>
                                <%= fa_icon 'edit' %> Edit Profile
                            <% end %>
```

Make devise to accept custom user fields

Everything about how to do it is in the documentation.

```
before_action :configure_permitted_parameters, if: :devise_controller?
```

The if conditions will perform the method only inside the devise_controller

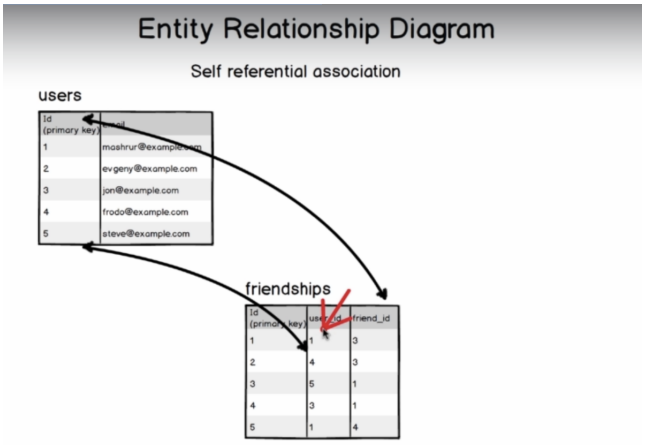
```ruby
class ApplicationController < ActionController::Base
  before_action :authenticate_user!
  before_action :configure_permitted_parameters, if: :devise_controller?

  protected

  def configure_permitted_parameters
    devise_parameter_sanitizer.permit(:account_update, keys: [:first_name, :last_name])
  end
end
```

276. Complete signup assignment

277. Self referential association - users and friends




```
~/complete_ror_course/finance_tracker_section/finance-tracker-6 — -bash    ...mplete_ror_course/finance_
$ rails g model Friendship user:references
```

Other column name reference

```ruby
t.references :user, null: false, foreign_key: true
t.references :friend, references: :users, foreign_key: { to_table: :users}
```

The column name will be “friend”, references table users, and the foreign key as well references table users.

models > friendship.rb

```ruby
class Friendship < ApplicationRecord
  belongs_to :user
  belogns_to :friend, class_name: 'User'
end
```

281. Search users/friends: implement search method

Like query

```ruby
User.where("email like ?", "%hossain%")
```

284. Implement add friend functionality

Friendship_path with post does not expect a parameter, but we can specify it with:

```
                                class: 'btn btn-sm btn-success' %>
<% if current_user.not_friends_with?(friend.id) %>
    <%= link_to 'Add to my portfolio', friendship_path(friend: friend),
                                class: 'btn btn-sm btn-success', method: :post %>
    <% else %>
```

Method build

Alias of new
Build is to create a new instance of an associated model.

```ruby
current_user.friendship.build(friend_id: params[:friend])
```

288. Start Photo App

On rails version 4 we should add the next gem so we can share resources through heroku

```
group :production do
  gem 'pg'
  gem 'rails_12factor'
end
```

Controller generator

Creates controller and action
rails g controller welcome index

290. Setup Authentication System

Install device

rails g devise:install

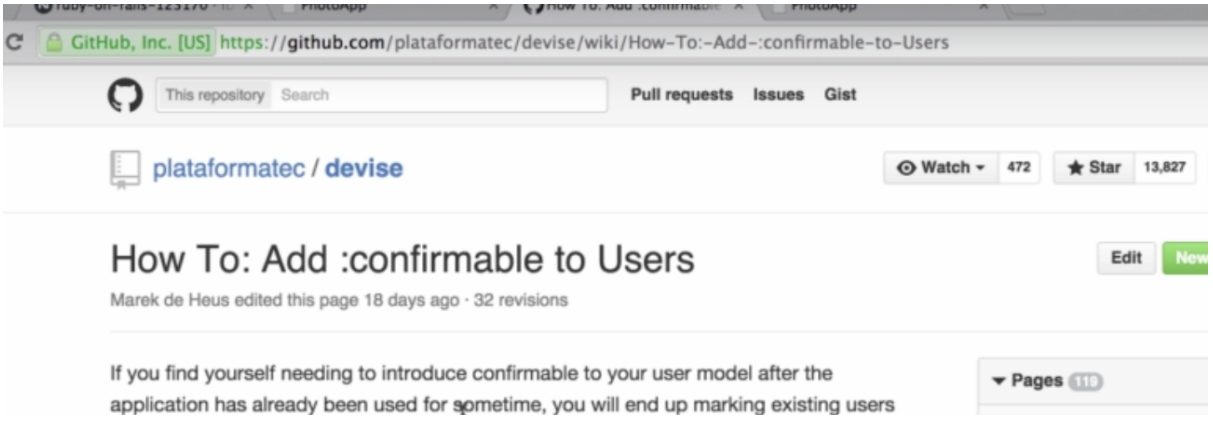
Create devise user

rails g devise user

Add confirmation to our devise user

So you need to confirm your email before starting to use an account.

There is a way to do it with documentation



An easier way is to modify the original migration

```
## Confirmable
t.string   :confirmation_token
t.datetime :confirmed_at
t.datetime :confirmation_sent_at
t.string   :unconfirmed_email # Only if using reconfirmable
```

Remember to migrate afterwards.

On the user model add :confirmable

```
class User < ActiveRecord::Base
  # Include default devise modules. Others available are:
  # :confirmable, :lockable, :timeoutable and :omniauthable
  devise :database_authenticatable, :registerable, :confirmable,
```

Make the app ask for authenticate on every single page

```
class ApplicationController < ActionController::Base
  before_action :authenticate_user!
end
```

Skip auth on specific action

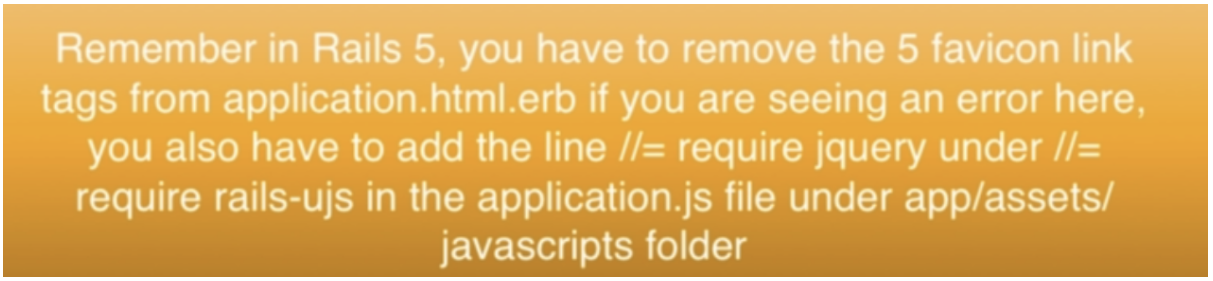
```
class WelcomeController < ApplicationController
  skip_before_action :authenticate_user!, only: [:index]

  def index
  end
end
```

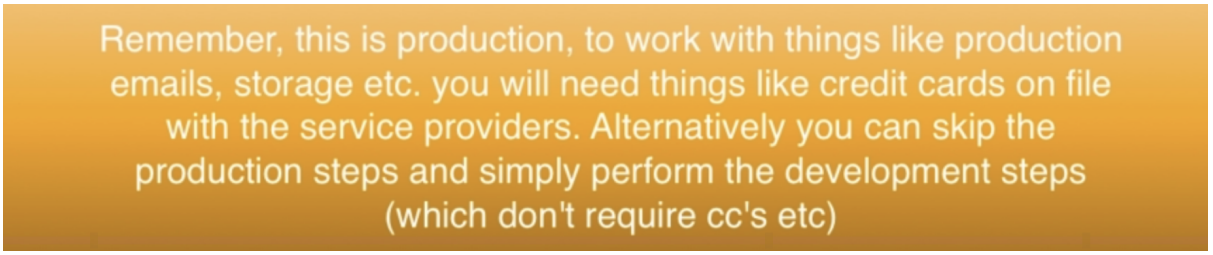
Install bootstrap

rails generate bootstrap:install static
rails g bootstrap:layout application
rails g devise:views:locale en
rails g devise:views:bootstrap_templates

Changes to do on rails 5



If you go to users/sign_up. If you add a user, an email will be sent.
On development you don't usually send emails.
On rails, if someone signs_up then the link or code will appear on the console.



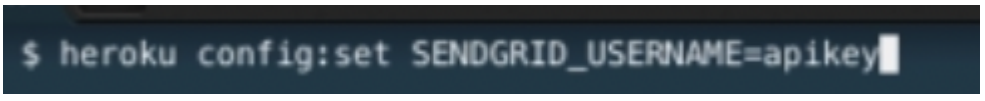
292. Sending Email in Production

*
After having your website hosted by heroku. Add sendgrid for emails into your project
heroku addons:create sendgrid:starter

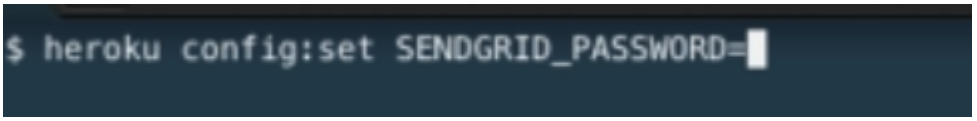
remember you should add your credit card on heroku.

Inside your app on heroku, click on sendgrid tier.
On settings, you can get your api keys.

Config keys on your local project;

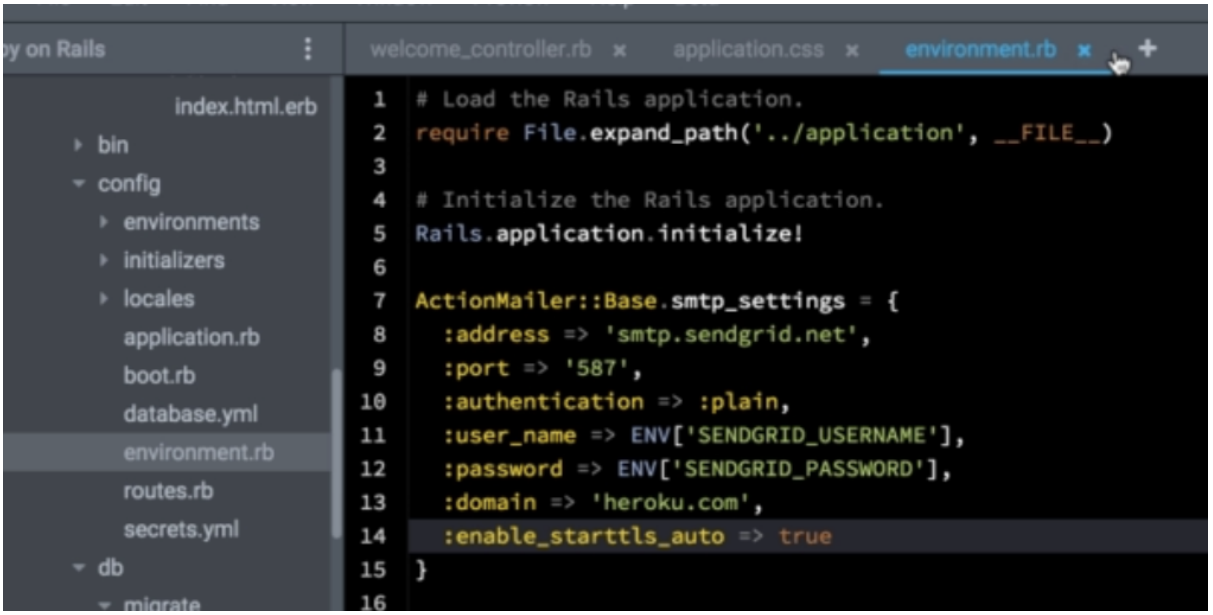


Here paste you api key

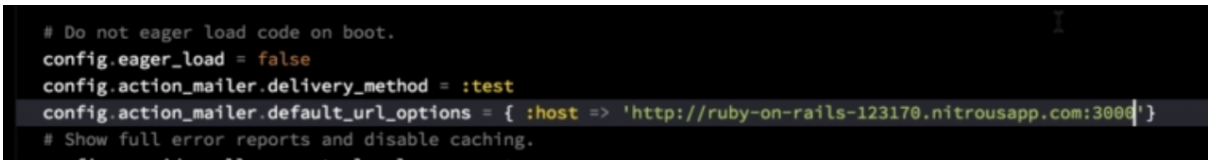


There is a hidden file called .profile
On the bottom, you put the username “api key” and password (the generated api key).
That is how you set the environmental variables.

On envirment.rb write:

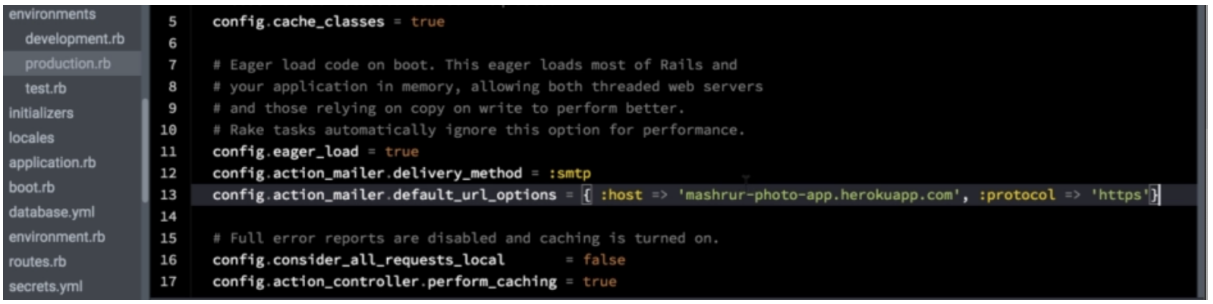


On on development.rb



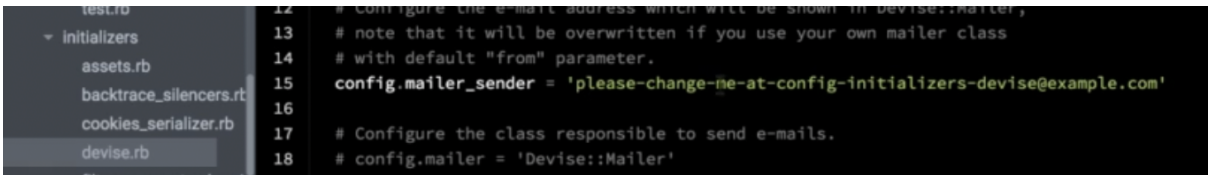
On the last line put the url of our local development project.

on production



294. Update Layout and Test Email in Production

On this file we can change the email in production



296. Build Homepage

298. Stripe and Payment Introduction

install stripe gem
On config/initializers create stripe.rb
Add this code

```
Rails.cofiguration.stripe = (
  :publishable_key => ENV['STRIPE_TEST_PUBLISHABLE_KEY'],
  :secret_key => ENV['STRIPE_TEST_SECRET_KEY']
)

Stripe.api_key = Rails.configuration.stripe[:secret_key]
```

Then convert the keys into environmental variables

300. Payment Model

Generate payment model:

rails g model Payment email:string token:string user_id:integer
We are going to create an association with the model.

Accepts nested attributes for

This allows a model to insert attributes of another model (its parent) through itself.
We did this because we are going to create user through the payment.

```
has_one :payment
accepts_nested_attributes_for :payment
```

> Do you mean using references:user instead of :user_id when generating the migration? Good question, generally references is better, but they are essentially equivalent from a functional, but references provides some additional benefits. You can research the differences online if you are interested.
>
> Overall: "Did he just feel like doing it manually?" Seems about right in this case.

has_one -> because the user is going to pay once for the service.
acceps_… -> will allow you to save payment records throughout the user, because the

Example of using accepts…

```
params = { member: { name: 'Jack', avatar_attributes: { icon: 'smiling' } } }
member = Member.create(params[:member])
member.avatar.id # => 2
member.avatar.icon # => 'smiling'
```

302. Update Form for Credit Card Payments

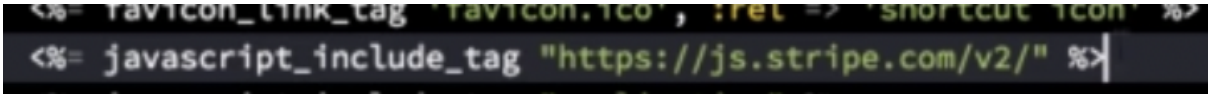
Fields for

Useful to add another model field into a form from another model.

304. Javascript Events

We send the form data to stripe, stripe verifies this info and it sends a token back then with a script we submit the form to the server.

We should add stripe js into the application.html.erb



306. Extend Devise Registrations Controller

When going into route registrations, try to find those controller methods inside my custom controller

310. Image Upload

Gems for images

gem 'carrierwave' -> gem to upload images
gem 'mini_magick' -> image resizing gem
gem ‘fog’ -> to work with images in production.

Reference another model with scaffolding

rails g scaffold Image name:string picture:string user:references

Use bootstrap to generate styled views for Images

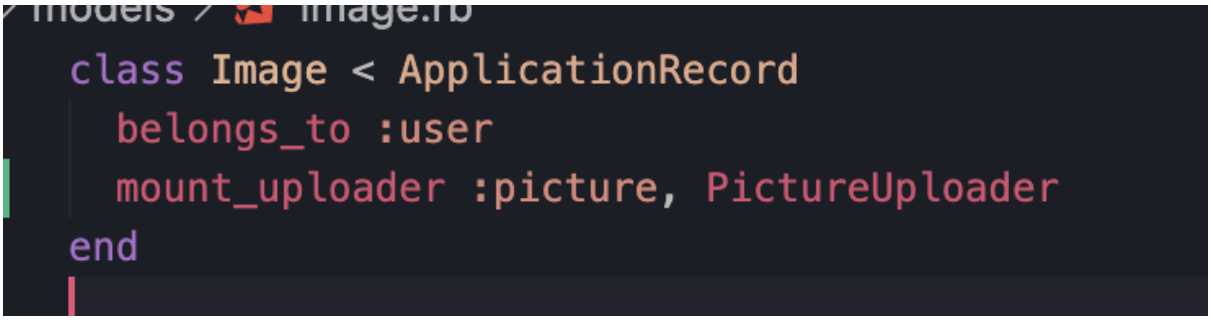
rails g bootstrap:themed Images

Uploader

Using a carrierwaveuploader gem (helps with uploading images), we can use the next command to generate an “uploader”

rails g uploader Picture

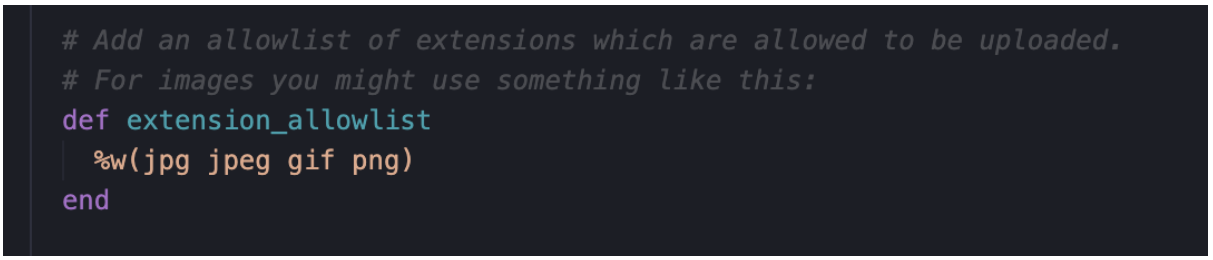
We link Image model with the uploader



:picture -> column name that we are linking
PictureUploader -> the uploader that we are going to use.

312. Image Size Validations

Uncomment this inside “picture_uploader” to add whitelist on image formats



Validate image size on image model

```ruby
validates :picture_size

private

def picture_size
  if picture.size > 5.megabytes
    errors.add(:picture, 'Should be less than 5MB')
  end      You, now • Uncommitted changes
end
end
```

Script to validate size on js

```html
<script type="text/javascript">
  $("#image_picture").bind("change", function() {
    var size_in_megabytes = this.files[0].size/1024/1024;
    if (size_in_megabytes > 5) {
      alert("Maximun file size is 5MB");
    }
  });
</script>
```

Automatic resize image when entering to storage

picture_uploader.rb × image.rb images_cont

app > uploaders > picture_uploader.rb

```ruby
include CarrierWave::MiniMagick
process resize_to_limit: [300, 300]
# Include RMagick or MiniMagick support:
```