

INFORME 4

REFACTORIZACIÓN DE CÓDIGO

GRUPO #6

GORLA, CAMILO

SALICIO, GABRIEL

#REUNION DE EQUIPO

EN LA REUNIÓN DE EQUIPO PARA ANALIZAR EL PROYECTO E IDENTIFICAR POSIBLES “BAD SMELLS” PUDIMOS DETECTAR LOS SIGUIENTES CASOS:

- DUPLICATE CODE
- LONG METHOD
- SPECULATIVE GENERALITY

DUPLICATE CODE

IDENTIFICAMOS CÓDIGO DUPLICADO EN TODAS LAS VISTAS. EN TODOS LOS CASOS CADA VISTA TENÍA DOS BLOQUES DE CÓDIGO REPETIDO.

UNO ERA UTILIZADO PARA INSERTAR TAGS DE REFERENCIA A HOJAS DE ESTILO DENTRO DEL TAG `<HEADER/>`

OTRO PARA RENDERIZAR EL MENÚ PRINCIPAL DE LA APLICACIÓN.

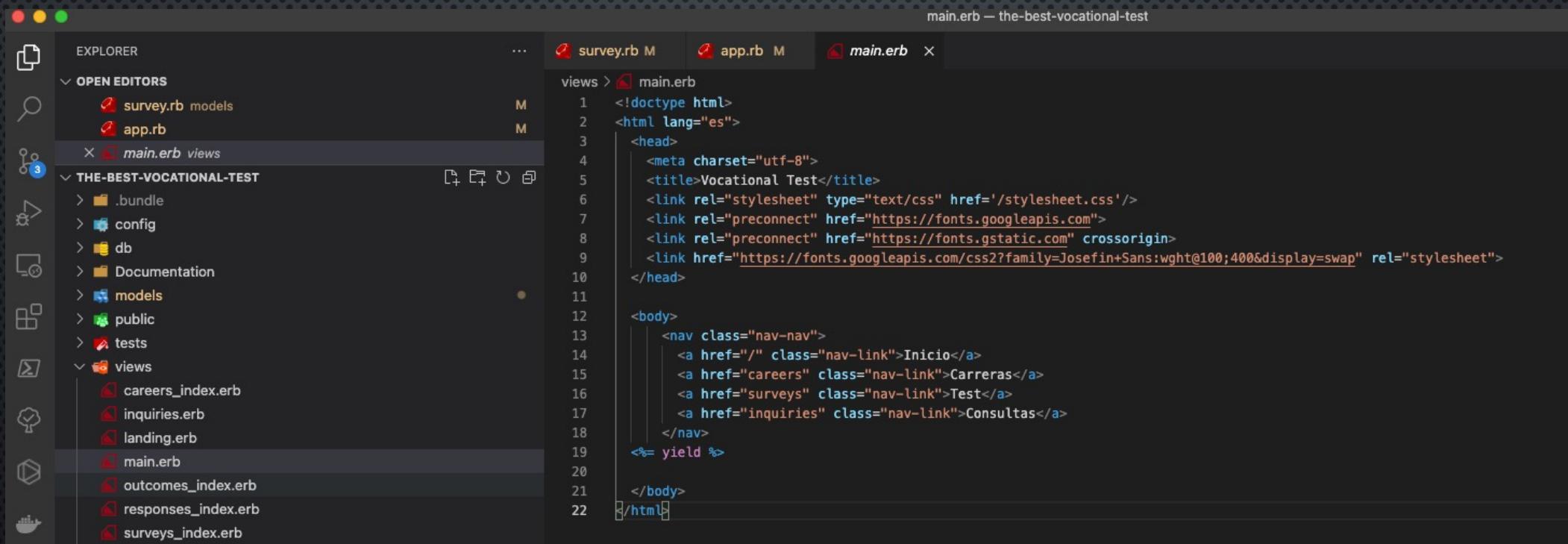
DADO QUE ESTOS BLOQUES SE ENCONTRABAN DISPERSOS EN DIFERENTES CLASES. UTILIZAMOS LA ESTRATEGIA “**EXTRACT CLASS**” PARA DEFINIR UNA NUEVA VISTA (LAYOUT) QUE CONTENGA LOS BLOQUES DE CÓDIGO COMÚN Y GRACIAS A LAS BONDADES DE RUBY + SINATRA AGREGAMOS UNA DIRECTIVA PARA QUE CADA VISTA SEA EMBEBIDA EN UN LAYOUT.

[VER CAMBIOS](#) (<https://github.com/gabrielsalicio/the-best-vocational-test/commit/27943b9277e341090796fd75c360cb2fdAA033ef>)

DUPLICATE CODE

USO DE LAYOUT EN SINATRA:

1 – CREAR UNA CLASE QUE CONTENGA EL CÓDIGO COMÚN A TODAS LAS VISTAS (LAYOUT)

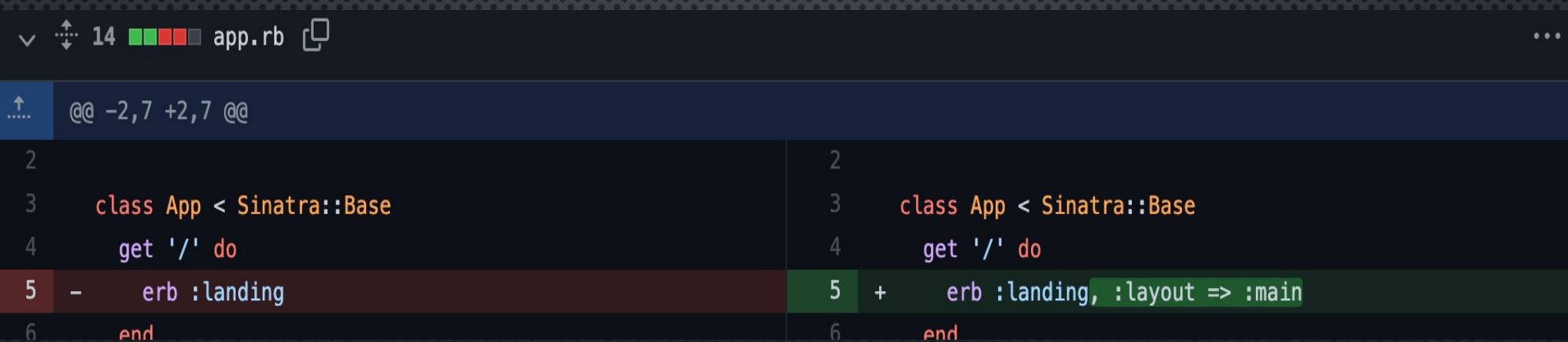


The screenshot shows a dark-themed interface of Visual Studio Code. On the left, the Explorer sidebar displays the project structure for 'THE-BEST-VOCATIONAL-TEST'. It includes files like survey.rb, app.rb, and several index files such as careers_index.erb, inquiries_index.erb, landing_index.erb, main_index.erb, outcomes_index.erb, responses_index.erb, and surveys_index.erb. The main editor area is titled 'main.erb — the-best-vocational-test' and contains the following code:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="utf-8">
    <title>Vocational Test</title>
    <link rel="stylesheet" type="text/css" href='/stylesheet.css'>
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Josefin+Sans:wght@100;400&display=swap" rel="stylesheet">
  </head>
  <body>
    <nav class="nav-nav">
      <a href="/" class="nav-link">Inicio</a>
      <a href="careers" class="nav-link">Carreras</a>
      <a href="surveys" class="nav-link">Test</a>
      <a href="inquiries" class="nav-link">Consultas</a>
    </nav>
    <%= yield %>
  </body>
</html>
```

DUPLICATE CODE

2 – AGREGAR UNA DIRECTIVA EN EL CONTROLADOR PARA UTILIZAR EL LAYOUT



The screenshot shows a code diff interface comparing two versions of an `app.rb` file. The left column shows the original code, and the right column shows the modified code with a green highlight.

```
@@ -2,7 +2,7 @@
2
3   class App < Sinatra::Base
4     get '/' do
5 -     erb :landing
6   end
2
3   class App < Sinatra::Base
4     get '/' do
5 +     erb :landing, :layout => :main
6   end
```

#LONG METHOD

SI BIEN EL PROYECTO CARECE DE LÓGICA COMPLEJA Y LA MAYORÍA DE SUS MÉTODOS NO TIENEN GRAN CANTIDAD DE CÓDIGO. UTILIZAMOS LA TÉCNICA DE “EXTRACT METHOD” POR DOS MOTIVOS.

1 – PARA HACER EL CÓDIGO MAS LEGIBLE

2 – RESALTAR LA IMPORTANCIA DE LA UTILIZACIÓN DE NOMBRES DE MÉTODOS QUE DESCRIBAN LA FUNCIONALIDAD QUE REALIZAN.

```
1 class Survey < Sequel::Model
2   one_to_many :responses
3   many_to_one :career
4
5   def validate
6     super
7     errors.add(:username, 'cannot be empty') if !username || username.empty?
8   end
9
10  dataset_module do
11    def number_of_carrers_between_dates(begin_date, end_date, select_career)
12      where(
13        :created_at => begin_date .. end_date,
14        :career_id => select_career
15      ).all
16    end
17  end
18
19 end
```

#1

CREAMOS UN MÉTODO QUE REALICE LA FUNCIONALIDAD DESEADA Y LE ASIGNAMOS UN NOMBRE DESCRIPTIVO.

#LONG METHOD

#2

CREAMOS UN MÉTODO QUE REALICE LA FUNCIONALIDAD DESEADA Y LE ASIGNAMOS UN NOMBRE DESCRIPTIVO.

```
108
109 post '/inquiries' do
110   @careers = Career.all
111   @begin_date = params[:begin_date]
112   @end_date = params[:end_date]
113   @select_career = params[:select_career].to_i
114   @result = Survey.where(
115     :created_at => @begin_date .. @end_date,
116     :career_id => @select_career
117   ).all.count
118   erb :inquiries, :layout => :main
119 end
120
```

```
post '/inquiries' do
  @careers = Career.all
  @begin_date = params[:begin_date]
  @end_date = params[:end_date]
  @select_career = params[:select_career].to_i
  @result = Survey.number_of_carrers_between_dates(@begin_date, @end_date, @select_career).count
  erb :inquiries, :layout => :main
end
```

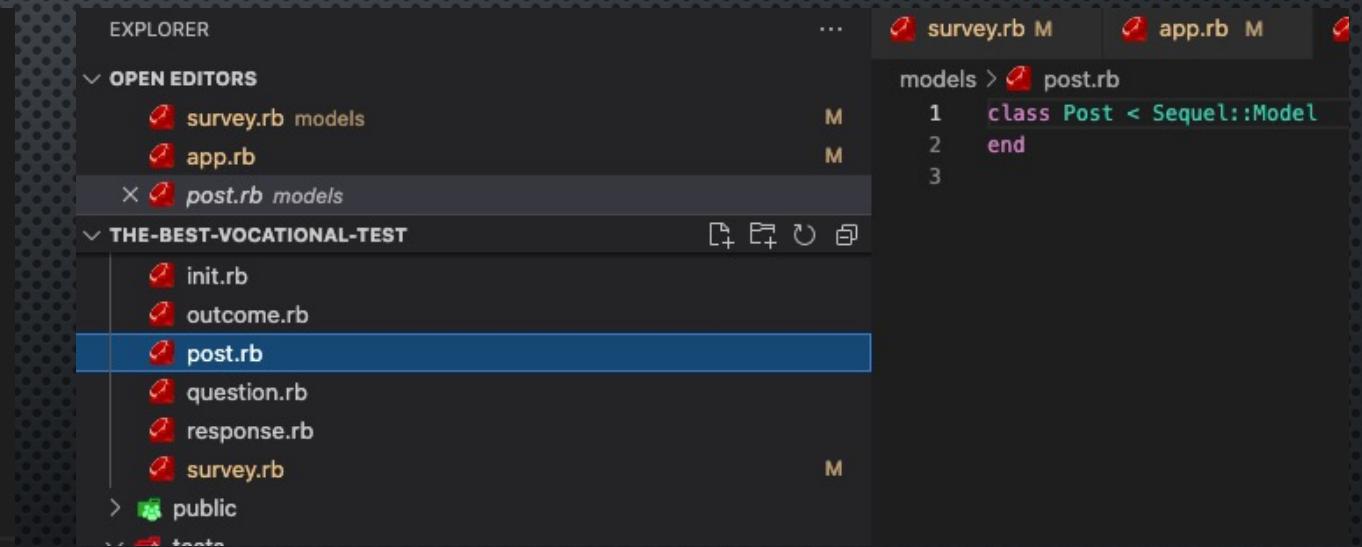
[Ver Cambios](https://github.com/gabrielsalicio/the-best-vocational-test/commit/cd4fe624a242e1bb0c475b8216f6a4f645a6b6e1) (<https://github.com/gabrielsalicio/the-best-vocational-test/commit/cd4fe624a242e1bb0c475b8216f6a4f645a6b6e1>)

#SPECULATIVE GENERALITY

- DETECTAMOS ENDPOINTS Y CLASES DE MODELO NO UTILIZADA. LA ESTRATEGIA A SEGUIR FUE SIMPLEMENTE ELIMINARLOS DEL PROYECTO YA QUE NO EXISTÍA NINGÚN TIPO DE REFERENCIA A ELLOS.

```
#POST & GET of posts
post "/posts" do
  request.body.rewind # in case someone already read it
  data = JSON.parse request.body.read
  post = Post.new(description: data['desc'])
  if post.save
    [201, { 'Location' => "posts/#{post.id}" }, 'CREATED']
  else
    [500, {}, 'Internal Server Error']
  end
end

get '/posts' do
  p = Post.where(id: 1).last
  p.description
end
```



#RUBOCOP

INTALAMOS Y EJECUTAMOS LA GEMA DE RUBOBOP PARA DETECTAR OFENSAS CON EL SIGUIENTE COMANDO:

```
$ RUBOCOP VIEWS MODELS APP.RB >RUBOCOP.TXT
```

8 FILES INSPECTED, 129 OFFENSES DETECTED, 117 OFFENSES AUTO-CORRECTABLE

EJECUTAMOS EL MISMO COMANDO CON LA OPCIÓN DE AUTO CORRECCIÓN

```
RUBOCOP VIEWS MODELS APP.RB -A
```

8 FILES INSPECTED, 22 OFFENSES DETECTED, 11 OFFENSES AUTO-CORRECTABLE

```
models > response.rb
1 class Response < Sequel::Model
2   one_to_one :choice
3   many_to_one :question
4   many_to_one :survey
5
6   def validate
7     super
8     errors.add(:choice_id, :name => 'choice_id cant be nil')
9     if not (choice_id) or (choice_id == nil)
10    errors.add(:question_id, :name => 'question_id cant be nil')
11    if not (question_id) or (question_id == nil)
12      errors.add(:survey_id, :name => 'survey_id cant be nil')
13      if not (survey_id) or (survey_id == nil)
14
15    end
16  end
```

```
1 class Response < Sequel::Model
2   one_to_one :choice
3   many_to_one :question
4   many_to_one :survey
5
6   def validate
7     super
8+   errors.add(:choice_id, name: 'choice_id cant be nil') if !choice_id or choice_id.nil?
9+   errors.add(:question_id, name: 'question_id cant be nil') if !question_id or question_id.nil?
10+  errors.add(:survey_id, name: 'survey_id cant be nil') if !survey_id or survey_id.nil?
11
12  end
```