Informe de Refactorización

Ingeniería de Software 2021
-Taller-

Grupo 10:

Ferreyra Juan
Marchio Luciano
Nolasco Agustin

UNRC
UNIVERSIDAD NACIONAL DE RÍO CUARTO

A continuación presentamos los endpoints/métodos que consideramos que tienen los cambios mas interesantes. Se muestra una version original del código y una posterior a los cambios, entre medio se da una explicación de lo realizado.

```
post '/surveys' do
  data = request.body.read
  if Survey.find(username: params[:username]).nil?
      survey = Survey.new(username: params[:username])
  else
      redirect '/?rejected=true'
  end
  if Question.first #if we have at least one question
    if survey.save
      [201, { 'Location' => "surveys/#{survey.id}" }, 'CREATED']
    else
      [500, {}, 'Internal Server Error']
    end
    first_question_id = Question.first.id
    #Redirect us to the first question
    redirect to("/questions/#{first_question_id}?survey_id=#{survey.id}")
  else
    #if we have no questions then finish
    redirect '/finish'
  end
end
```

El bad smell que se detecto fue el uso innecesario del temporal first_question_id, aplicamos la regla de "Eliminación de temporal" (Inline Temp) y en el redirect usamos directamente Question.first.id. Por otro lado, cambiamos .nil? por .exist? ya que se hace mas legible, también eliminamos el endpoint /finish y mostramos la view no_question_template directamente. La creación del survey se movió, ahora se crea solo si hay preguntas, lo cual tiene mas sentido. Se borro ademas la variable data que no se usaba. Por ultimo cambiamos Question.first que quizás no es tan descriptivo, por Question.empty?

Finalmente quedó como sigue:

```
post '/surveys' do
  if Survey.find(username: params[:username]).exist?
    redirect '/?rejected=true'
  end
  if Question.empty?
    erb :no_question_template
  else
    survey = Survey.new(username: params[:username])
    if survey.save
      [201, { 'Location' => "surveys/#{survey.id}" }, 'CREATED']
```

```
    else
      [500, {}, 'Internal Server Error']
    end
    redirect to("/questions/#{Question.first.id}?survey_id=#{survey.id}")
  end
end
```

```
patch '/responses/:survey_id' do
  response = Response.find(survey_id: params[:survey_id], question_id: params[:question_id])
  response.update(survey_id: params[:survey_id], question_id: params[:question_id], choice_id: params[:choice_id])
  if response.save
    [201, { 'Location' => "responses/#{response.id}" }, 'UPDATED']
    #Redirect us to the next question
    if (params[:next_question] == 'true')
      question = response.question.next
    else
      if (params[:next_question] == 'end')
        redirect "/finish/#{params[:survey_id]}"
      else
        question = response.question.prev
      end
    end
    if question.nil?
      redirect "/finish/#{params[:survey_id]}"
    end
    redirect to("/questions/#{(question.id)}?survey_id=#{params[:survey_id]}")
  else
    [500, {}, 'Internal Server Error']
  end
end

post '/responses/:survey_id' do
  question_id = params[:question_id]
  if (params[:choice_id].nil?)
    question = Question.find(id: question_id)
    if (params[:next_question] != 'true')
      question = question.prev
    end
    redirect to("/questions/#{(question.id)}?survey_id=#{params[:survey_id]}")
  end

```

```
35    response = Response.create(question_id: question_id,
       choice_id: params[:choice_id], survey_id:
       params[:survey_id])
36    if response.save
37      [201, { 'Location' => "responses/#{response.id}" },
       'CREATED']
38      #Redirect us to the next question
39      if (params[:next_question] == 'true')
40        question = response.question.next
41      else
42        question = response.question.prev
43      end
44      if question.nil?
45        redirect "/finish/#{params[:survey_id]}"
46      end
47      redirect
       to("/questions/#{(question.id)}?survey_id=#{params[:survey_id]}")
48    else
49      [500, {}, 'Internal␣Server␣Error']
50    end
51  end
```

El mecanismo de re-dirección a la siguiente pregunta fue mejorado en termino de nombres para que sea mas descriptivo. Detectamos bad smell luego de mejorar esto ya que en patch y post se usaba exactamente el mismo código para la re-dirección a la siguiente pregunta, por lo tanto aplicamos "Extraer método" para eliminar esa repetición innecesaria, con ello se agrego un nuevo método redirect_question que se guardo en el archivo util.rb. En dicho método se puede apreciar que cambiamos el if anidado que había por un case y se colocaron nombres mas representativos a los parámetros.

En particular en el patch se cambió el update, que recibía parámetros extra que no debía tomar ya que no se modificaban.

Por otro lado en el post se detectó otro bad smell al utilizar la variable question_id, allí aplicamos "Eliminación de temporal" (Inline Temp) y usamos directamente params[:question_id] en los dos lugares donde era necesario.

A continuación el resultado:

```
1   patch '/responses/:survey_id' do
2     response = Response.find(survey_id: params[:survey_id],
       question_id: params[:question_id])
3     response.update(choice_id: params[:choice_id])
4     if response.save
5       [201, { 'Location' => "responses/#{response.id}" },
       'UPDATED']
6       redirect_question(response.question,
       params[:incoming_question], params[:survey_id])
7     else
8       [500, {}, 'Internal␣Server␣Error']
9     end
10  end
```

```
11
12 post '/responses/:survey_id' do
13   if (params[:choice_id].nil?)
14     question = Question.find(id: params[:question_id])
15     if (params[:incoming_question] == 'prev')
16       question = question.prev
17     end
18     redirect
    to("/questions/#{(question.id)}?survey_id=#{params[:survey_id]}")
19   end
20   response = Response.create(question_id:
    params[:question_id], choice_id: params[:choice_id],
    survey_id: params[:survey_id])
21   if response.save
22     [201, { 'Location' => "responses/#{response.id}" },
    'CREATED']
23     redirect_question(response.question,
    params[:incoming_question], params[:survey_id])
24   else
25     [500, {}, 'Internal␣Server␣Error']
26   end
27 end
```

Este método se encuentra en el archivo models/util.rb

```
1 def redirect_question(curr_question, requested_question,
    survey_id)
2   case requested_question
3   when 'next'
4     redirect
    "/questions/#{curr_question.next.id}?survey_id=#{survey_id}"
5   when 'prev'
6     redirect
    "/questions/#{curr_question.prev.id}?survey_id=#{survey_id}"
7   when 'end'
8     redirect "/finish/#{survey_id}"
9   end
10 end
```

```
1   def answered?(survey_id)
2     response = Response.find(survey_id: survey_id,
    question_id: self.id)
3     return (not response.nil?)
4   end
5
6   def choice_selected(survey_id)
7     response = Response.find(survey_id: survey_id,
    question_id: self.id)
```

```
    if response.nil?
      return nil
    else
      return Choice.find(id: response.choice_id)
    end
  end
```

El bad smell detectado aquí fue el temporal response, en ambos métodos hicimos "Eliminación de temporal" (Inline Temp) reemplazando directamente por la consulta. También aplicamos "Extraer método" creando uno que nos permite obtener la response de un usuario a partir de su id.

Por último, en el método choice_selected quitamos el if ya que no retornar nada es lo mismo que retornar nil (al menos en esta situación).

Finalmente nos queda:

```
def get_response(survey_id)
  return Response.find(survey_id: survey_id, question_id:
   self.id)
end

def answered?(survey_id)
  return self.get_response(survey_id).exist?
end

def choice_selected(survey_id)
  if self.answered?(survey_id)
    return Choice.find(id:
   self.get_response(survey_id).choice_id)
  end
end
```

```
#Compute the result of a Survey
def compute_result
  #Use a HashMap to index the careers
  careers_count = Hash.new
  for c in Career.all
    careers_count[c.id] = 0
  end

  for r in self.responses
    choice = Choice.find(id: r.choice_id)
    for o in choice.outcomes
      careers_count[o.career_id] += 1
    end
  end

  max_ocurrences_career_id =
   careers_count.key(careers_count.values.max)
```

```
17    self.update(career_id: max_ocurrences_career_id)
18    self.update(completed_at: Sequel.lit('NOW()'))
19  end
20  def completed?
21    return (!self.career.nil?)
22  end
23  def self.count_completed
24    sum = 0
25    for s in Survey.all
26      if (s.completed?)
27        sum += 1
28      end
29    end
30    return sum
31  end
```

El bad smell detectado fue el temporal max_ocurrences_career_id en el método compute_result, se realizó "Eliminación de temporal" (Inline Temp). Por otra parte había un update por cada valor que se actualizaba, ahora fue reemplazado por uno solo que contiene ambos cambios. Los ciclos for se remplazan por .each que hacen el proceso mas transparente y similar al lenguaje natural. El temporal choice también fue eliminado. También se removieron comentarios que eran o se volvieron inútiles.

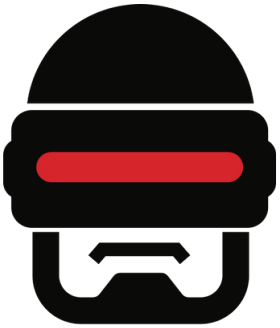
En el método .completed? se cambio el !.nil? por un .exist? que se había definido antes. En el método estático count_completed también se cambió el for por un .each.

El resultado es el siguiente:

```
1   def compute_result
2     careers_count = Hash.new
3     Career.all.each { |career| careers_count[career.id] = 0 }
4
5     self.responses.each do |response|
6       Choice.find(id: response.choice_id)
7         .outcomes.each { |outcome|
      careers_count[outcome.career_id] += 1 }
8     end
9
10    self.update(
11      career_id: careers_count.key(careers_count.values.max),
12      completed_at: Sequel.lit('NOW()')
13    )
14  end
15
16  def completed?
17    return self.career.exist?
18  end
19
20  def self.count_completed
21    count = 0
22    Survey.all.each { |survey| count += 1 if survey.completed?
```

```
    }
23    return count
24  end
```

Para cerrar con esta sección, algo que también se hizo fue borrar variables globales que se utilizaban en las views, en lugar de eso ahora se usa directamente la consulta dentro de la view, modificado solo para consultas simples, como Question.all y Career.all.



Rubocop

Para la parte de rubocop se crearon tres archivos donde almacenamos la información que nos daba la herramienta, así como también generamos un archivo de configuración para la misma.

Después de correrla (sin que corrija las ofensas) nos encontramos con que, por ejemplo, si teníamos un if con una sola linea en su cuerpo, podemos usar un 'unless' (si la condición esta negada). Otra ofensa que se repitió mucho fue utilizar obj.find(:attr => 'something') cuando la convención es obj.find(attr: 'something'). También detectaba ofensas como los espacios, tabs o paréntesis innecesarios. Además nos sugirió usar el método .positive? para chequear si un valor era mayor que 0 . Los mencionados fueron los principales cambios, al menos en el archivo app.rb.

Por otro lado en el modelo survey.rb la herramienta sugirió crear el HashMap con {} en lugar de Hash.new y eliminar el uso del self, ya que era redundante en ese contexto.

En el modelo question.rb recomendó el uso del if de una sola linea, muy parecido al unless mencionado en app.rb. Si bien la herramienta se ejecutó sobre la mayoría de código Ruby de nuestra aplicación, para este análisis solo en tuvimos en cuenta lo comentado anteriormente, ya que fue donde centramos la actividad de refactorización.

Todo lo anterior mencionado fue corregido automáticamente por rubocop, sin embargo quedaron algunas sugerencias que rubocop no podía resolver de forma segura, por ejemplo en un archivo de test se encontraban variables a las que se les asignaba un valor, pero luego no se utilizaban, y rubocop sugería eliminarlas, lo cual hicimos de forma manual.

Finalmente nuestro código quedado sin ofensas.