

Taller IS 2021 - Actividad Nro 04

Ejercicio 2:

Posibles “bad smells” identificados junto a las reglas de refactorización que podríamos aplicar para mejorar la calidad del código.

- 1) Métodos Mayores a 10 LOD.

En el app.rb

Método post `"/finish_survey"` sin refactorizar

```
post "/finish_survey" do
  data = JSON.parse request.body.read
  # obtengo valor del usuario
  @username = data["username"]
  # si existe alguna encuesta asociada a ese usuario la borramos
  oldSurvey=Survey.where(username: @username)
  oldSurvey.each do |survey|
    oldResponse = Response.where(survey_id: survey.id)
    oldResponse.destroy
  end
  oldSurvey.each do |survey|
    survey.destroy
  end
  # creo la encuesta con el usuario
  @survey = Survey.new(username: @username)
  @survey.save
  #guardo las respuestas del usuario
  createResponses(data["choices"],@survey.id)
  response = Response.all
  # creo el arreglo de carreras que contendrá los pesos
  result = {}
  ## inicializando arreglo de carreras y pesos
  for career in Career.all
    result[career.id] = 0
  end
  # Por cada respuesta reviso en outcome con que carrera machea
  # y al arreglo de carreras incremento el peso de esa carrera en 1
  response.each do |response|
    o = Outcome.where(choice_id: response.choice_id).last
    if(o && o.career_id)
      result[o.career_id] = result[o.career_id] + 1
    end
  end
  # Obtengo el máximo del arreglo de carreras
  resultCareer = result.key(result.values.max)
  #busco ese id de carreras en la tabla de carrera y se lo asigno a una variable
  @career = Career.find(id: resultCareer)
  # actualizo el valor del registro de la encuesta con la carrera que ganó
  @survey.update(career_id: @career.id)
  # llamo a la página html que me muestra la carrera ganadora
  redirect to "/finish/#{@survey.id}"
end
```

Método post "/search" sin refactorizar

```
post "/search" do
  result = 0
  data = JSON.parse request.body.read
  @fechaD = data['fechaD']
  @fechaH = data['fechaH']
  carrera = data['carrera']
  @career = Career.find(name: carrera)
  survey = Survey.where(career_id: @career.id)
  survey.each do |p|
    val = Date.parse(String(p.updated_at))
    if(val > Date.parse(@fechaD) && val <= Date.parse(@fechaH))
      result = result + 1
    end
  end
  @salida = " fue elegida "+result.to_s+" veces."
  redirect to "/finish_search/?fechaD=#{@fechaD}&fechaH=#{@fechaH}&salida=#{@salida}&career=#{@career.name}"
  #redirige los resultados a una nueva página
end
```

La regla de refactorización que se podría aplicar sería: **Long Method**

Cuando un método es demasiado grande puede estar pasando que tenga demasiadas variables declaradas o que contenga demasiado código. En el caso que el método sea grande estamos en presencia de posible repetición de código.

La solución más simple es eliminar la redundancia en la propia clase.

Al igual que con una clase con muchas variables, la solución habitual para una clase con demasiado código es extraer la clase, extraer el módulo o extraer la subclase (dividir el método en métodos más pequeños)

Proponemos los métodos en varios métodos más pequeños, y de esta forma reducimos la complejidad de cada uno de los métodos.

para ello dividimos los métodos en métodos más pequeños que cumplan condiciones similares. A continuación mostraremos los cambios realizados sobre cada método.

Metodo post "/finish_survey" Refactorizado

```
post "/finish_survey" do
  data = JSON.parse request.body.read
  @username = data['username'] # si existe alguna encuesta asociada a ese usuario la borramos
  deleteOldSurvey(@username)
  @survey = Survey.new(username: @username) # creo la encuesta con el usuario
  @survey.save #guardo las respuestas del usuario
  createResponses(data['choices'], @survey.id)
  result = {}
  # creo el arreglo de carreras que contendrá los pesos inicializando arreglo de carreras y pesos
  initializedResult(result)
  calcResult(result, @career, @survey)
  redirect to "/finish/#{@survey.id}"
  #llamo a la pagina html que me muestra la carrera ganadora
end

def deleteOldSurvey(username)
  oldSurvey = Survey.where(username: username)
  oldSurvey.each do |survey|
    oldResponse = Response.where(survey_id: survey.id)
    oldResponse.destroy
  end
  oldSurvey.each do |survey|
    survey.destroy
  end
end

def initializedResult(result)
  for career in Career.all
    result[career.id] = 0
  end
end

# Por cada respuesta reviso en outcome con que carrera machea
# y al arreglo de carreras incremento el peso de esa carrera en 1
def calcResult(result, career, survey)
  response = Response.all
  response.each do |response|
    o = Outcome.where(choice_id: response.choice_id).last
    if (o && o.career_id)
      result[o.career_id] = result[o.career_id] + 1
    end
  end
  resultCareer = result.key(result.values.max) # Obtengo el maximo del arreglo de carreras
  @career = Career.find(id: resultCareer) #busco ese id de carreras en la tabla de carrera y se lo asigno a una variable
  @survey.update(career_id: @career.id) # actualizo el valor del registro de la encuesta con la carrera que gano
end
```

Método post "/search" Refactorizado

```
post "/search" do
  data = JSON.parse request.body.read
  @fechaD = data['fechaD']
  @fechaH = data['fechaH']
  carrera = data['carrera']
  res = 0
  @career = Career.find(name: carrera)
  compareDate(@career, @fechaD, @fechaH, :res, binding)
  @salida = " fue elegida "+res.to_s+" veces."
  redirect to "/finish_search/?fechaD=#{@fechaD}&fechaH=#{@fechaH}&salida=#{@salida}&career=#{@career.name}" #redirige los resultados a una
nueva página
end

def compareDate(career, fechaD, fechaH, res, bdg)
  survey = Survey.where(career_id: career.id)
  survey.each do |p|
    val = Date.parse(String(p.updated_at))
    if((val > Date.parse(fechaD) && val <= Date.parse(fechaH)) || (val >= Date.parse(fechaD) && val < Date.parse(fechaH)))
      #aqui debo comparar la fecha de actualizacion del registro con la fecha pasada como parametro
      eval "#{res} = #{res} + 1", bdg
    end
  end
end

end
```

Ejercicio 4:

Ejecutamos la aplicación Rubocop sobre nuestro proyecto y detectó la cantidad de 1069 infracciones en 33 expedientes inspeccionados. La mayoría de las infracciones son de estilo.

A continuación mostramos el Informe de inspección emitido por Rubocop.




RuboCop Inspection Report

33 files inspected, 1069 offenses detected:

- [Gemfile - 7 offenses](#)
- [app.rb - 146 offenses](#)
- [config.ru - 10 offenses](#)
- [db/migrations/000010_alter_table_responses.rb - 14 offenses](#)
- [db/migrations/000011_create_outcomes_table.rb - 2 offenses](#)
- [db/migrations/000012_alter_table_outcomes.rb - 10 offenses](#)
- [db/migrations/000013_alter_table_careers.rb - 5 offenses](#)
- [db/migrations/000014_alter_table_choices.rb - 3 offenses](#)
- [db/migrations/00001_create_posts_table.rb - 2 offenses](#)
- [db/migrations/00002_add_date_fields_to_posts.rb - 1 offense](#)
- [db/migrations/00003_create_careers_table.rb - 1 offense](#)
- [db/migrations/00004_create_questions_table.rb - 2 offenses](#)
- [db/migrations/00005_create_surveys_table.rb - 2 offenses](#)
- [db/migrations/00006_alter_table_surveys.rb - 4 offenses](#)
- [db/migrations/00007_create_choices_table.rb - 2 offenses](#)
- [db/migrations/00008_alter_table_choices.rb - 6 offenses](#)
- [db/migrations/00009_create_responses_table.rb - 2 offenses](#)
- [db/seeds.rb - 337 offenses](#)
- [models/career.rb - 15 offenses](#)
- [models/choice.rb - 16 offenses](#)
- [models/init.rb - 2 offenses](#)
- [models/outcome.rb - 16 offenses](#)
- [models/post.rb - 1 offense](#)
- [models/question.rb - 18 offenses](#)
- [models/response.rb - 18 offenses](#)
- [models/survey.rb - 22 offenses](#)
- [tests/models/careers_test.rb - 75 offenses](#)
- [tests/models/choices_test.rb - 82 offenses](#)
- [tests/models/outcomes_test.rb - 29 offenses](#)
- [tests/models/questions_test.rb - 127 offenses](#)
- [tests/models/responses_test.rb - 46 offenses](#)
- [tests/models/surveys_test.rb - 32 offenses](#)
- [tests/test_helper.rb - 14 offenses](#)

Luego utilizamos el comando para la generación de los archivos rubocop.yml y rubocop_todo.yml, de esta manera evitamos usar el comando rubocop -a al comienzo del ejercicio porque nos había generado cambios en el código que hacían que nuestro programa no funcionara adecuadamente, incluso los cambios sugeridos por rubocop hacía que fallara la compilación del proyecto por lo que la metodología a seguir fue ir comentando uno a uno los estilos encontrados en el archivo rubocop_todo.yml e ir corrigiendo de a una por vez. Además nos dimos cuenta que algunas ofensas al ser corregidas hacen que cierta parte del programa no se ejecutara correctamente.

A continuación mostramos una imagen que nos indica el estado del proyecto sin la aplicación la política de métricas



RuboCop Inspection Report

33 files inspected, 6 offenses detected:

- [tests/models/careers_test.rb](#) - 1 offense
- [tests/models/choices_test.rb](#) - 1 offense
- [tests/models/outcomes_test.rb](#) - 1 offense
- [tests/models/questions_test.rb](#) - 2 offenses
- [tests/models/responses_test.rb](#) - 1 offense

Ejecución de políticas de Métricas:

- Metrics/AbcSize:
 - Max: 28

Esta política nos indica que en los casos de las clases de test existen métodos con mas de 10 lineas, pero como son casos de test consideramos que están correctos.

Ejemplos:

tests/models/careers_test.rb - 3 offenses

Line #27 – **Convention:** Metrics/MethodLength: Method has too many lines. [14/10]

```
def test_career_has_many_outcomes ...
```

Decidimos que esta política no sea tomada en cuenta, pues todos los archivos detectados son los de test.

- Metrics/BlockLength:
 - Max: 28

No se encontraron ofensas asociadas a esta métrica.

- Metrics/ClassLength:
 - Max: 118

En este caso la cantidad de líneas que excede a lo recomendado son 15 líneas, por lo cual consideramos dejar habilitada esta ofensa y tenerla en cuenta para otro momento.

app.rb - 1 offense

Line #4 – **Convention:** Metrics/ClassLength: Class has too many lines. [115/100]

```
class App < Sinatra::Base ...
```

Salida de los TESTs al ejecutarlos luego de la refactorización:

Salida del test survey:

```
darioVostro-3700 ~/ProyectoAYD/sinatra-sequel-starter-app_g11 (develop)$ docker exec -it 1eb2ea2d2fe3 ruby tests/models/surveys_test.rb
MiniTest::Unit::TestCase is now Minitest::Test. From tests/models/surveys_test.rb:4:in `<main>'
Run options: --seed 26084

# Running:

..

Finished in 0.044166s, 45.2838 runs/s, 67.9258 assertions/s.

2 runs, 3 assertions, 0 failures, 0 errors, 0 skips
```

Salida del test career:

```
darioVostro-3700 ~/ProyectoAYD/sinatra-sequel-starter-app_g11 (develop)$ docker exec -it 1eb2ea2d2fe3 ruby tests/models/careers_test.rb
MiniTest::Unit::TestCase is now Minitest::Test. From tests/models/careers_test.rb:4:in `<main>'
Run options: --seed 12047

# Running:

...

Finished in 0.091823s, 32.6714 runs/s, 32.6714 assertions/s.

3 runs, 3 assertions, 0 failures, 0 errors, 0 skips
```

Salida del test question:

```
darioVostro-3700 ~/ProyectoAYD/sinatra-sequel-starter-app_g11 (develop)$ docker exec -it 1eb2ea2d2fe3 ruby tests/models/questions_test.rb
MiniTest::Unit::TestCase is now Minitest::Test. From tests/models/questions_test.rb:4:in `<main>'
Run options: --seed 7630

# Running:

.....

Finished in 0.092563s, 54.0170 runs/s, 64.8204 assertions/s.

5 runs, 6 assertions, 0 failures, 0 errors, 0 skips
```

Salida del test outcome:

```
darioVostro-3700 ~/ProyectoAYD/sinatra-sequel-starter-app_g11 (develop)$ docker exec -it 1eb2ea2d2fe3 ruby tests/models/outcomes_test.rb
MiniTest::Unit::TestCase is now Minitest::Test. From tests/models/outcomes_test.rb:4:in `<main>'
Run options: --seed 19907

# Running:

.

Finished in 0.032691s, 30.5892 runs/s, 61.1783 assertions/s.

1 runs, 2 assertions, 0 failures, 0 errors, 0 skips
```

Salida del test response:

```
darioVostro-3700 ~/ProyectoAYD/sinatra-sequel-starter-app_g11 (develop)$ docker exec -it 1eb2ea2d2fe3 ruby tests/models/responses_test.rb
MiniTest::Unit::TestCase is now Minitest::Test. From tests/models/responses_test.rb:4:in `<main>'
Run options: --seed 21751

# Running:

.

Finished in 0.028820s, 34.6983 runs/s, 69.3966 assertions/s.

1 runs, 2 assertions, 0 failures, 0 errors, 0 skips
```


Salida del test choice:

```
darloVostro-3700 ~/ProyectoAYD/sinatra-sequel-starter-app_g11 (develop)$ docker exec -it 1eb2ea2d2fe3 ruby tests/models/choices_test.rb
MiniTest::Unit::TestCase is now Minitest::Test. From tests/models/choices_test.rb:4:in `<main>'
Run options: --seed 14891

# Running:

...

Finished in 0.088340s, 33.9597 runs/s, 33.9597 assertions/s.

3 runs, 3 assertions, 0 failures, 0 errors, 0 skips
```