

# Actividad 4 - Métricas - Refactorización de Código - Grupo 8

Integrantes:

- Ignacio Gonzalez
- Agustin Calderon
- Edu Pera

## Bad Smells identificados:

1. Duplicated code (Codigo duplicado)

Este chequeo cuando guardamos cualquier objeto en la BDD esta presente múltiples veces en app.rb y otras clases

```
if career.save
  [201, {'Location' => "careers/#{career.id}"}, 'Career succesfully created']
else
  [500, {}, 'Internal Server Error']
end
```

Refactorizacion aplicable:

Extract Method

Es decir crear un nuevo método en algun lugar y llamarlo cada vez que se necesite esta funcionalidad

2. Feature Envy (Envidia de funcionalidad?)

En Survey tenemos definido este método que esta dirigido a la clase response mas que a survey, lo único que toma de un survey es un id

```

55 #for a given collection of choices, creates responses and saves them in database
56 def create_responses(selected_choices)
57   selected_choices.each do |question_and_choice|
58     response = Response.create(question_id: question_and_choice[0], choice_id: question_and_choice[1], survey_id: self.id)
59
60     if response.save
61       [201, { 'Location' => "responses/#{response.id}" }, 'CREATED']
62     else
63       [500, {}, 'Internal Server Error']
64     end
65   end
66 end

```

Refactorización aplicable:

Move Method

Mover el método a la clase de donde se hace su uso.

Es la misma solución que del problema 2, de esta forma traemos un método a la clase response.

```

11 #takes a collection of careers and an id and creates a Score object with
12 #the given survey_id and career, for each career in the collection
13 #self makes the method static so we don't need an instance to invoke it
14 def self.create_scores(careers, survey_id)

```

La solución es simplemente borrar o simplificar los comentarios

## Cosas que detectamos pero no refactorizamos

- Tuvimos que hacer un Shotgun Surgery cuando introducimos la funcionalidad de la consulta de scores, en todos los lugares donde teníamos `survey.careers` tuvimos que cambiarlo por `survey.scores` y esto nos trajo un problema, ya que nos olvidamos de cambiarlo en uno de los archivos y gracias a eso la página se rompía en cierto punto.

## Analisis rubocop

Al correr rubocop por primera vez sobre la aplicación nos dio esto:

```
36 files inspected, 598 offenses detected, 526 offenses auto-correctable
```

---

Podemos ver que analizo 36 archivos en donde nos detecto unas 598 ofensas totales de las cuales 526 se pueden autocorregir.

Volvemos a pasar el rubocop y corregimos las automáticas y nos tira esto:

```
36 files inspected, 62 offenses detected, 5 offenses auto-correctable
```

Vemos que arreglo la gran mayoría de ofensas y que solo quedan unas 62 de las cuales 5 se pueden auto corregir

Finalmente viendo el archivo que nos iba generando para ver cuales eran las ofensas, detectamos ofensas sobre 2 tipos de métricas, LongMethod y ABCSize, en total 2 de la primera métrica y 6 de la segunda.

- LongMethod:
  - En el metodo result en survey.rb, donde se pasaba solo por dos líneas de código 12/10 nosotros no consideramos que era necesario hacer ese cambio de cambiar el método solo por pasarnos 2 líneas.
  - En el método test\_response\_must\_have\_choice\_question\_and\_survey en responses\_test.rb, en este caso se pasaba solo por una línea de código 11/10 y como además era un método para los tests decidimos que no era necesario refactorizarlo.

```

22   def result
23     topCareers = Array.new #array for best careers
24     hashCareer = self.career_scores #obtain scores
25     maxScore = hashCareer.values.max #score of best career
26
27     if maxScore.nil? #case of empty hashmap (no career scored)
28       topCareers[0] = Career.first #null career
29       return topCareers
30     end
31
32     #filter the hash leaving only the best and all careers tied in score with the best one
33     hashCareer.select! {|key,value| value == maxScore}
34
35     hashCareer.keys.each do |career_id| #for every career id in hash
36       topCareers.append(Career.find(id: career_id)) #load Career object with this id in the array
37     end
38     return topCareers
39   end

```

```

models/survey.rb:22:3: C: Metrics/MethodLength: Method has too many lines. [12/10]
  def result ...
  ^^^^^^^^^^^

```

- ABCSize:
  - Esta métrica nos indica que hay muchas asignaciones, llamadas a métodos de otras clases y condiciones en un método, estas ofensas nos salían en métodos de test únicamente, ya que teníamos cosas hardcodeadas para simplemente hacer el test.

Con lo cual no nos pareció necesario realizar una refactorización de esto:

```

tests/models/responses_test.rb:6:3: C: Metrics/AbcSize: Assignment Branch Condition size
for test_response_must_have_choice_question_and_survey is too high. [<8, 18, 0> 19.7/17]
  def test_response_must_have_choice_question_and_survey ...
  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

```

Luego con respecto a las demás ofensas, muchas eran sobre los tests, los cuales no nos importan mucho, con lo cual no corregimos dichas ofensas.