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

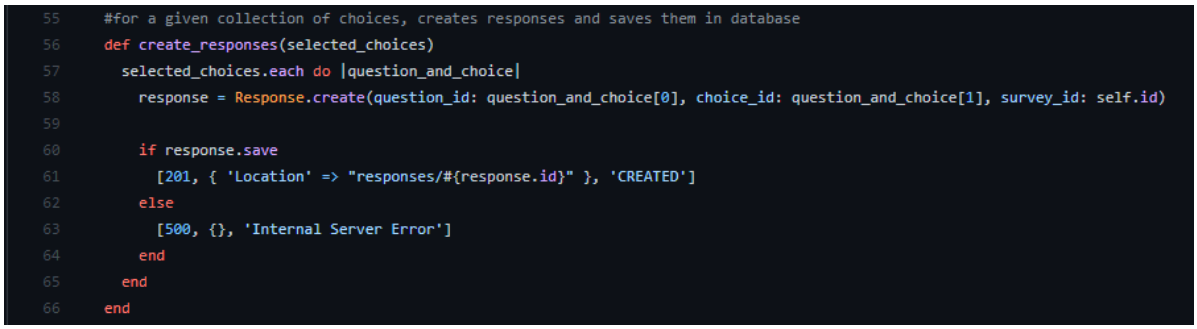
     Es decir crear un nuevo método en algun lugar y llamarlo cada vez que se necesite esta funcionalidad

   Finalmente así quedaría definido el nuevo método:

```ruby
def try_save(item_to_save, message)
  if item_to_save.save
    [201, { 'Location' => "item_to_save/#{item_to_save.id}" }, message]
  else
    [500, {}, 'Internal Server Error']
  end
end
```

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



```ruby
#for a given collection of choices, creates responses and saves them in database
def create_responses(selected_choices)
  selected_choices.each do |question_and_choice|
    response = Response.create(question_id: question_and_choice[0], choice_id: question_and_choice[1], survey_id: self.id)

    if response.save
      [201, { 'Location' => "responses/#{response.id}" }, 'CREATED']
    else
      [500, {}, 'Internal Server Error']
    end
  end
end
```

   Refactorizacion aplicable:

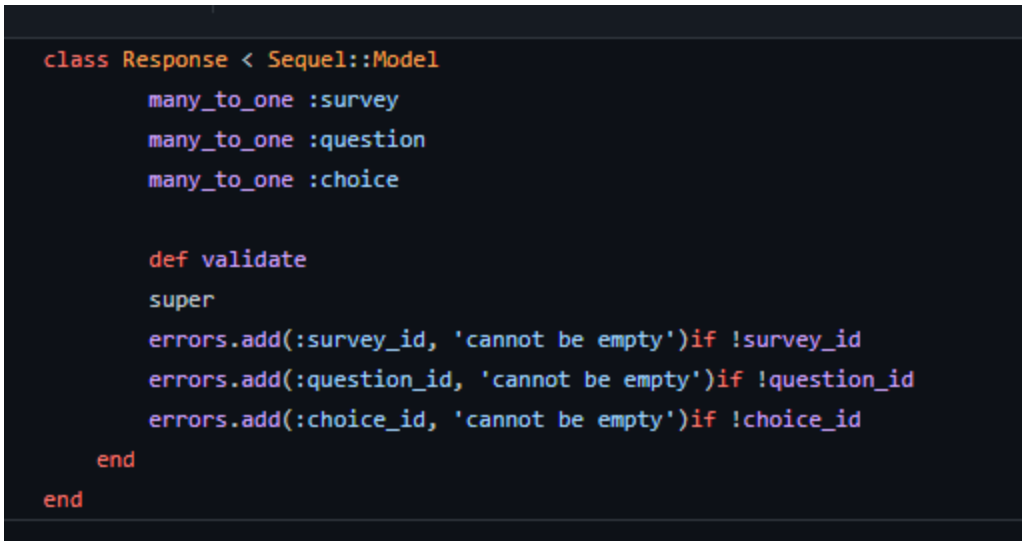
   - Move Method
   - Mover el método a la clase de donde se hace su uso.
     De todas formas en create_responses no necesitamos ningún atributo de response, con lo que quizás no sea un feature envy, pero creemos que esta mejor tener el método en response.

   Finalmente se movió ese método a la clase response donde tiene mas sentido que este (la imagen esta en el 3).

3. Data Class (clase de datos)

   En Response tenemos únicamente atributos y el método validate, lo cual hace que sea prácticamente una clase que solo tiene atributos.

Esto esta relacionado con el problema de 2, ya que podemos traer ese método a esta clase



```ruby
class Response < Sequel::Model
        many_to_one :survey
        many_to_one :question
        many_to_one :choice

        def validate
        super
        errors.add(:survey_id, 'cannot be empty')if !survey_id
        errors.add(:question_id, 'cannot be empty')if !question_id
        errors.add(:choice_id, 'cannot be empty')if !choice_id
    end
end
```

Refactorizacion aplicable:

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

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

Finalmente al aplicar la refactorización anterior donde se movió el método create_response() a la clase response ya esta clase no seria tanto un data class sino que ya tiene algunos métodos:

```ruby
class Response < Sequel::Model
  many_to_one :survey
  many_to_one :question
  many_to_one :choice

  def validate
    super
    errors.add(:survey_id, 'cannot be empty') unless survey_id
    errors.add(:question_id, 'cannot be empty') unless question_id
    errors.add(:choice_id, 'cannot be empty') unless choice_id
  end

  # for a given collection of choices, creates responses and saves them in database
  def self.create_responses(selected_choices, survey_id)
    selected_choices.each do |question_and_choice|
      response = Response.create(question_id: question_and_choice[0], choice_id: question_and_choice[1],
                                 survey_id: survey_id)

```

4. Unnecessary comments

   No esta listado como bad smell, pero lo documentamos acá para hacer el cambio

   Escribimos estos comentarios sin ninguna razón, ya que el nombre del método es muy descriptivo

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

- La solución es simplemente borrar o simplificar los comentarios, finalmente quedo:

```ruby
# create and save scores in db
def self.create_scores(careers, survey_id)
```

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 pagina se rompía en cierto punto.

- El metodo result en survey.rb nos parecia un poco largo y ademas teniamos comentarios en varias lineas. Por lo que el libro decia, en donde hay comentarios es probable que haya que hacer un metodo para esa funcionalidad.
  Finalmente decidimos no refactorizarlo porque el metodo no era demasiado largo y ademas algunos de esos comentarios en el metodo no eran del todo necesarios

- Muchas de las clases eran Data Classes, pero no habia otras clases que tuvieran metodos que encajen con dichas clases, con lo cual la posible solucion era fusionar estas clases en una sola.

  Finalmente decidimos no hacerlo porque a pesar de ser data class, nos interesaba la estructura que nos daban sobre la base de datos y para entender e implementar la funcionalidad del sistema.

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 ejecutar rubocop corrigiendo automaticamente los errores que se pueden 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_has_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.

```ruby
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_has_choice_question_and_survey is too high. [<8, 18, 0> 19.7/17]
  def test_response_must_has_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 y otras eran sobre convenciones sobre nombres de variables y cosas por el estilo que si corregimos.