INFORME
ACTIVIDAD 4

- Grupo 5 -
Lassalle Evelyn
Bortol Juan Pablo
Oga Luciano

Se comenzó realizando una reunión con el equipo tratando de localizar posibles Bad Smells, encontrando método con demasiadas líneas de código, métodos sin ninguna función en particular,errores de indentación,redundancia de ciertas funcionalidades (Duplicate code)

Inline Method
Como primera medida en app.rb se eliminó un método innecesario que no cumplía ninguna función importante dentro del proyecto.

```
get "/hello/:name" do
    @name = params[:name]
    erb :hello_template
end
```

Inlineclass
También, se eliminó el método get "/careers", ya que pudimos observar que para simplificar código el recorrido Career.all se podría realizar directamente en la vista careers_index.erb.

```
get "/careers" do
  @careers = Career.all
  erb :careers_index
end
```

Continuando con la eliminación de código innecesario, se eliminó el método '/surveys_careers' ya que el recorrido de las carreras actuales lo realizamos dentro de la vista surveys_careers.erb y a su vez ya tenemos implementado un post ‘/careers_info_survey’ que accedía a la vista.

```
get '/surveys_careers' do
  @careers = Career.all
  erb :surveys_career
end
```

Por los cambios realizados anteriormente, se decidió proceder con las siguientes modificaciones.

Pre código

```
post '/info_careers' do
  redirect :careers
end
```

Post código

```
post '/info_careers' do
  erb :careers_index
end
```

Con la eliminación de código innecesario logramos reducir la cantidad de líneas de código en general.

Una de las modificaciones que considerábamos importantes era tratar de simplificar el método que realiza el cálculo para filtrar la cantidad de carreras resultantes en un rango de tiempo, dadas las fechas y carrera en sí, utilizando la función filter y count, reemplazando el if y el contador que se utilizaba.

Pre código

```
post '/surveys_careers' do
  @fecha_ini = params[:FechaIni]
  @fecha_fin = params[:FechaFin]
  @career = Career.find(id: params[:career_id])
  @count = 0
  Survey.all.each do |survey|
    if (survey.career_id == @career.id) &&
(survey.created_at.strftime('%Y-%m-%d') >= @fecha_ini)
&& (survey.created_at.strftime('%Y-%m-%d') <=
@fecha_fin)
      @count += 1
    end
  end
  erb :quantity_careers
end
```

Post código

```
post '/surveys_careers'do
    @fecha_ini = params[:FechaIni]
    @fecha_fin = params[:FechaFin]
    @career = Career.find(id: params[:career_id])
    @count = Survey.all.filter{|x| x.career_id == @career.id
&& x.created_at.strftime('%Y-%m-%d') >=  @fecha_ini &&
x.created_at.strftime('%Y-%m-%d') <= @fecha_fin}.count()
    erb :quantity_careers
  end
```

Extract Method

Definimos la función calculate_career_points(survey) con el objetivo de modularizar el post /response que observamos que contenía ciertos módulos de código que podrían ir fuera de este ya que se encargaban de hacer cosas particulares. En este caso la función realizada nos retorna la puntuación de cada carrera, basándose en la cantidad de surveys que tiene asociada.

```
def calculate_career_points(survey)
  pointsCareers = {}
  Career.all.each do |career|
    pointsCareers[career.id] = 0
  end
  survey.responses.each do |response|
    Outcome.all.each do |outcome|
      if (response.choice_id == outcome.choice_id)
        pointsCareers[outcome.career_id] += 1
      end
    end
  end
  return pointsCareers
end
```

Se eliminaron los comentarios que consideramos innecesarios, ya que a simple vista se puede observar claramente de que se encarga cada método.

Consideramos que el resto de los archivos Ruby, como los modelos, tests, migraciones.. no tienen demasiadas cosas a modificar manualmente ya que son archivos con pocas líneas de código y los módulos definidos son cortos y hacen básicamente lo que se necesita.

EJECUCIÓN DE RUBOCOP

Se detallan a continuación los archivos examinados con cada offense retornado y su solución:

TESTS:

- careers_test = 4 offenses, 1 corregido automáticamente, 2 de forma manual.

- choices_test: 3 offensees, 2 corregidos automáticamente.

- outcomes_test: 14 offensees, 12 corregidos de forma automática (indentacion), 1 no supimos solucionarlo.

- offenses sin solucionar: Metrics/AbcSize: Assignment Branch Condition size for test_has_many_responses is too high. [<5, 17, 0> 17.72/17]

- questions_test: 14 offensees, 12 corregidos automáticamente, 1 de forma manual (variable que estaba sin uso).

- responses_test: 15 offensees, 11 corregido automáticamente (indentación), 2 de forma manual(variables que no hacían nada se eliminaron).
  1 offense que no se pudo solucionar de ninguna manera.
  offense sin solucionar: Metrics/AbcSize: Assignment Branch Condition size for test_has_many_responses is too high. [<5, 17, 0> 17.72/17]

- surveys_test: 13 offensees, 11 corregidos automáticamente(indentación), 1 de forma manual (2 variables que no hacía falta declararlas, con crearlas en DB era suficiente).

IMPORTANTE! offense en cuanto a poner un comentario saltaba en todas las ejecuciones de cada test, fue solucionado en todos los casos fácilmente.

MODELOS:
- career: 2 offensees
- choice: 2 offensees
- question: 1 offense corregido de forma automática.
- response: 3 offensees corregidos de forma automática.
- outcome: 2 offensees
- survey: 2 offensees
- post: 1 offense corregido de forma automática.

En todos los casos donde se encontraron 2 offensees, 1 se corrigió siempre de forma automática y el otro manualmente.
offense corregido manualmente(agregar comentario): Style/Documentation: Missing top-level documentation comment

MIGRACIONES:
- Creación tabla career: 1 offense
- Creación tabla post: 1 offense
- Agregar dato en post: 1 offense
- Creación tabla choice: 1 offense
- Creación tabla question: 1 offense
- Creación tabla outcome: 1 offense
- Creación tabla survey: 1 offense
- Creación tabla response: 1 offense
- Asociación choice question: 1 offense
- Asociación response choice: 1 offense

Se pudo observar que en todo los casos retorno el mismo offense, el cual fue corregido de forma automática en todas las oportunidades.
offense: Style/FrozenStringLiteralComment: Missing frozen string literal comment.

SEEDS:
78 offensees, 7 corregidos automáticamente. El resto de los offensees eran de tildes, demasiado código en una misma línea, indentación..los cuales todos fueron corregidos manualmente.

APP:
En la primera ejecución se detectaron 72 offensees, 61 se corrigieron automáticamente y el resto de forma manual.

Se encontraron problemas de indentación, offensees con algunos comentarios, espacios necesarios e innecesarios, expresiones demasiadas largas y demasiadas líneas de código en algunos módulos.

Los offensees corregidos manualmente siempre fueron en todos los casos medianamente sencillos, como por ejemplo reducir una condición que era demasiado larga, en este caso particular se decidió bajar la condición una línea más abajo y el problema fue solucionado.

IMPORTANTE! Rubocop convirtió todos los CamelCase en snake_case