Informe

Integrantes:
- *Arce, Alan*
- *Molina, Agustin*
- *Palacio, Luciano*

Materia:
Ingeniería de software

Actividad 2) Detección de “bed smells” y regla de refactorización aplicada

- Duplicated code

En dos de las vistas, precisamente en careers y outcomes usábamos una secuencia de asignaciones seguida de varios ifs. Y las expresiones que asignabamos o controlabamos eran las mismas.

Applied refactoring rules: Extract method

Lo que hicimos fue reemplazar ambas partes de código por un método que contiene estas asignaciones y un case que reemplaza los ifs que había anteriormente. Reduciendo considerablemente la cantidad de líneas de código en las vistas, ya que solo resto invocar al método antes nombrado.

```
def associateLink
        link = ""
        agronomia = "https://www.unrc.edu.ar/unrc/carreras/ayv_ingenieria_agronomica.htm"
        computacion = "https://www.unrc.edu.ar/unrc/carreras/exa_lic_ciencias_computacion.php"
        economia = "https://www.unrc.edu.ar/unrc/carreras/eco_lic_economia.htm"
        edFisica = "https://www.unrc.edu.ar/unrc/carreras/hum_prof_educacion_fisica.php"
        filosofia = "https://www.unrc.edu.ar/unrc/carreras/hum_lic_en_filosofia.php"
        fisica = "https://www.unrc.edu.ar/unrc/carreras/exa_prof_fisica.php"
        geologia = "https://www.unrc.edu.ar/unrc/carreras/exa_lic_geologia.php"
        historia = "https://www.unrc.edu.ar/unrc/carreras/hum_lic_historia.php"
        literatura = "https://www.unrc.edu.ar/unrc/carreras/ProgAsig5.php?n=%2FBpLfBDYxXmQaxDePIM%2Bvqvf5cLyAysalAWtwPjvud4%3D"
        matematica = "https://www.unrc.edu.ar/unrc/carreras/exa_lic_matematica.php"
        medicina = "https://fcm.unc.edu.ar/"
        periodismo = "https://www.unrc.edu.ar/unrc/carreras/hum_comunicacion_social.php"
        quimica = "https://www.unrc.edu.ar/unrc/carreras/exa_lic_quimica.php"
        sociologia = "https://sociales.unc.edu.ar/licenciaturasociologia"
        veterinaria = "https://www.unrc.edu.ar/unrc/carreras/ayv_medicina_veterinaria.htm"
        arquitectura = "https://faud.unc.edu.ar/carrera-de-arquitectura/"
        arte = "https://artes.unc.edu.ar/"
        if self.name == "Agronomia" then
                link = agronomia
        elsif self.name == "Arquitectura" then
                link = arquitectura
                link = economia
elsif self.name == "Educacion Fisica" then
                link = edFisica
elsif self.name == "Filosofía" then
                link = filosofia
elsif self.name == "Fisica" then
                link = fisica
elsif self.name == "Geologia" then
                link = geologia
elsif self.name == "Historia" then
                link = historia
elsif self.name == "Literatura" then
                link = literatura
elsif self.name == "Matemáticas" then
                link = matematica
elsif self.name == "Medicina" then
                link = medicina
elsif self.name == "Periodismo" then
                link = periodismo
elsif self.name == "Quimica" then
                link = quimica
elsif self.name == "Sociologia" then
                link = sociologia
elsif self.name == "Veterinaria" then
                link = veterinaria
end

return link
```

- Lazy class

Teníamos una clase para llevar la cuenta de la cantidad de veces que una carrera fue resultado de un cuestionario. Esta requería un alto costo de mantenimiento y consumía muchos recursos, además de que era completamente ineficiente e innecesaria.

Applied refactoring rules: Extract class

La clase fue eliminada. Y se reemplazó por un método mucho más efectivo y conciso que realiza una consulta, aportando así la misma funcionalidad para la cual se había hecho la clase en cuestión.

```
post "/quantities" do
  @career = Career.find(name: (params[:carrera]))
  @initialDate = DateTime.iso8601(params[:fecha1])
  @auxDate = Time.parse(@initialDate.to_s)
  @finalDate = DateTime.iso8601(params[:fecha2])
  days = (@finalDate - @initialDate).to_i
  quantities = Quantity.all
  @careers = Career.all
  @cant = 0

  for c in 0..days do
    arr = quantities.select{|q| q.date.year == @auxDate.year && q.date.month == @auxDate.month && q.date.day == @auxDate.day && q.c
    if arr.empty?
      c = 0
    else
      c = arr[0].cant
    end
    @cant += c
    @auxDate = @auxDate + (86400)
  end

  erb :quantities_index
 end

end
```

- Long method
    1) En un método de la clase app utilizabamos dos, uno para crear un nuevo objeto y otro para guardarlo en la base de datos.

Applied refactoring rules: Extract method

Optimizamos este método por medio de la extracción de estas dos invocaciones, a una que utilizaba un método que realiza los dos trabajos requeridos.

    2) En el app.rb teníamos un método bastante largo que realizaba varias tareas para cumplir su propósito.

Applied refactoring rules: Extract method

Lo que se hizo fue tomar una parte del código en donde se encontraban acciones perteneciente a una “etapa” de la tarea y definir un nuevo método con el objetivo de cumplir esa etapa de la tarea. Esto redujo el código en el app.tb y dio mayor nivel de entendimiento al código.

```
  table = {}

  for career in Career.all
    table[career.id] = 0
  end

  for response in @survey.responses
    choice = Choice.find(id: response.choice_id)
    for outcome in choice.outcomes
      table[outcome.career_id] += 1
    end
  end
```

```
  career = table.key(table.values.max)
  career = Survey.searchSuitableCareer(@survey)
  @career = Career.find(id: career)

  @survey.update(career_id: @career.id)
```

```
def self.searchSuitableCareer(survey)
  table = {}

    for career in Career.all
      table[career.id] = 0
    end

    for response in survey.responses
      choice = Choice.find(id: response.choice_id)
      for outcome in choice.outcomes
        table[outcome.career_id] += 1
      end
    end

    return table.key(table.values.max)
  end
end
```

- Misnamed variables

En nuestro código había variables con nombres poco intuitivos, lo que dificultaba el entendimiento.

Applied refactoring rules

Reemplazamos cada uno de estos nombres por uno que representara de una mejor manera el (objetivo | intención | la función) de esa variable.

Por ejemplo: car_id -> career_id

- Large class

La clase app.rb se redujo considerablemente al aplicar cada uno de los métodos de refactorización mencionados.

*** Extract method tiene más de una forma de proceder por eso aparece aplicado más de una vez. Además de ser una técnica muy utilizada.

Actividad 4) Conclusiones del análisis realizado por rubocop (metricas)

Algunas de las ofensas detectadas por la herramienta fueron:

<u>Corrección automática</u>

- Indentación

En nuestro código existían líneas mal indentadas y la herramienta las indento correctamente.

- Espacios de mas entre lineas de código

Estas líneas fueron eliminadas por rubocop.

<u>Corrección manual</u>

- Notación

A algunas de nuestras variables las habíamos escrito con notación camelCase. Y lo resolvimos cambiando estos nombres por la notación que utiliza ruby, la cual es snake_case.

- Variables sin usar

También teniamos variables obsoletas. A estas directamente las eliminamos del código.