

Universidad Nacional de Río Cuarto.

Facultad de Ciencias Exactas, Químicas y Naturales.

Departamento de Computación.

Analista en Computación.

Asignatura: Ingeniería de Software (3304)

Taller IS2020 : Ciclo de Refactorización

Grupo 2

Integrantes:

-Ameri, Pablo Osiris.

-Ré, Nicolas

-Villarreal, Arian

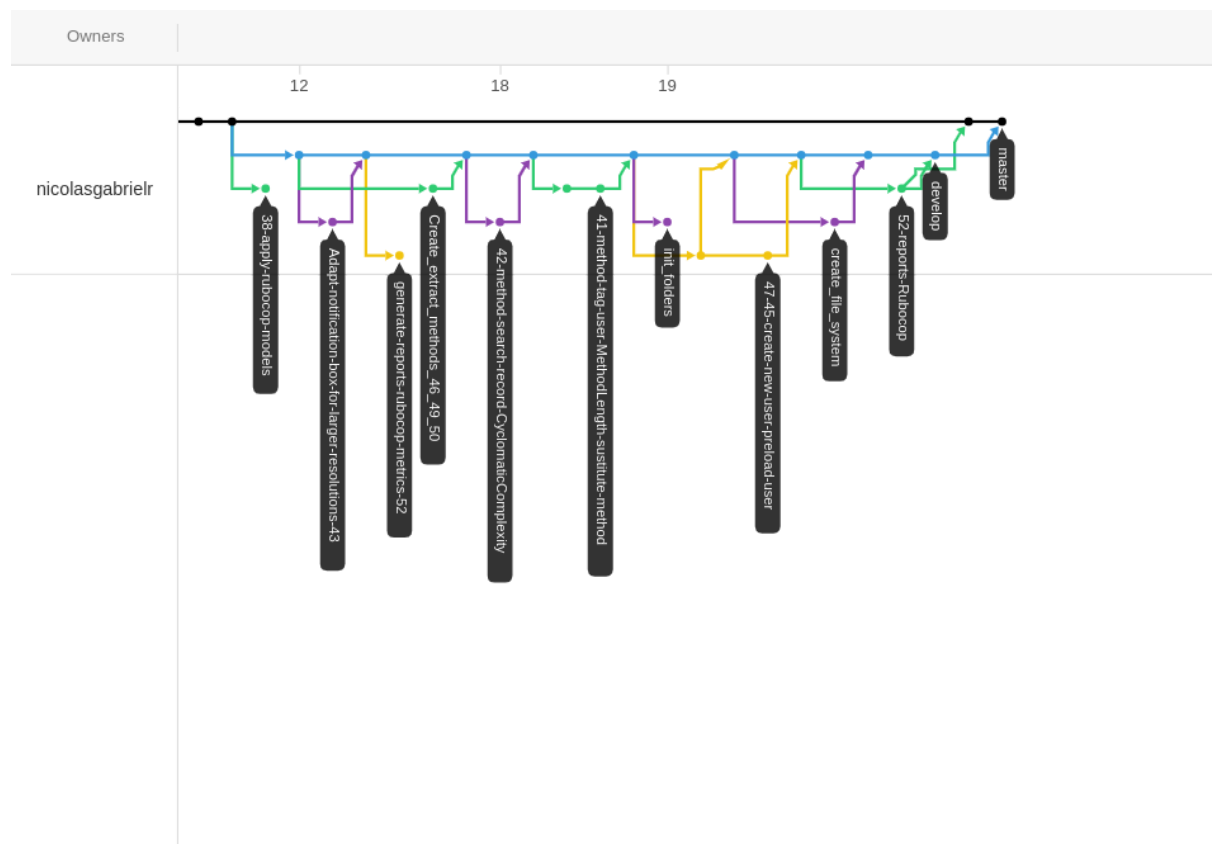
Durante este ciclo de refactorización, se incorporó al proyecto, una herramienta de gestión “Pivotal Tracker” que provee mecanismos ágiles para un mejor desarrollo de las actividades. Esta herramienta resultó bastante intuitiva a la hora de su utilización y facilitó significativamente la organización de las tareas.

Nuestro proyecto en Pivotal Tracker:

<https://www.pivotaltracker.com/n/projects/2465910>

Durante el sprint, cada tarea generada en Pivotal Tracker se corresponde con su respectivo issue en git que a su vez durante su resolución, se generó para cada una, su correspondiente branch (ramas), todas estas se derivaron de la rama develop que creamos inicialmente desde la rama Master.

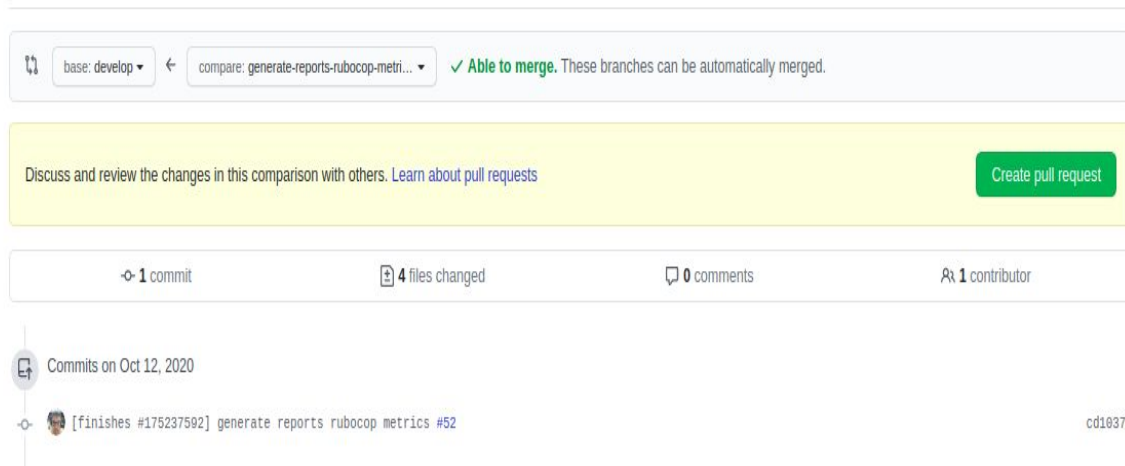
Cuando llegamos a una versión “aceptable” o “sin errores” se mergeó nuevamente a la rama Master.



Al finalizar cada issue, el integrante designado al mismo realizaba un **pullrequest** hacia la rama mencionada anteriormente (develop):

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).



Y en caso de no tener conflictos, otro integrante del grupo chequeaba la posible presencia de anomalías o errores, si todo estaba correcto aprobaba la solicitud y confirmaba los cambios sobre develop mediante un merge. Esto permitió al proyecto una mayor organización y manejo de los cambios. (1)

Si bien, desde nuestra presentación inicial consideramos que nuestra aplicación estaba bastante pulida en cuanto a funcionalidad, teníamos bastantes errores relacionados a estilos (longitud de métodos, longitud de nombres, etc), algoritmos de complejidad ciclomática elevada entre otros.

Por lo cual, a modo recomendación de los profesores, incorporamos Rubocop, un analizador de código estático basado en la guía de estilos de Ruby, que se instala dentro del proyecto y analiza el código en busca de errores de estilos, y en caso de ser posible, los corrige automáticamente.

Al ejecutarlo dentro de nuestra app.rb inicialmente, los resultados que obtuvimos fueron 351 ofensas. La mayoría se corrigieron de forma automática.



Todos los reportes generados en html:

<https://github.com/nicolasgabrielr/ProjectGroup/tree/master/reports/html>

(2)

Algunas de las ofensas correspondientes a métricas no podían ser corregidas por la aplicación.

Para realizar esas tareas, recurrimos a algunas de las técnicas de refactorización mencionadas en el libro “Refactoring Ruby Edition” y guías de estilos de Ruby, en el caso de ser necesario, ante la imposibilidad de ajustarse a las sugerencias de RuboCop, seteamos lo que nosotros consideramos correcto para nuestra aplicación.

Line #6 – Convention: Metrics/ClassLength: Class has too many lines. [489/100]

Para este caso en particular, se modificó el tamaño máximo de ClassLength. Dado que nuestra clase contiene a todo el resto de la app.

Ofensas en el método search_record:

Line #171 – Convention: Metrics/AbcSize: Assignment Branch Condition size for search_record is too high. [<10, 9, 15> 20.15/17]

Line #171 – Convention: Metrics/CyclomaticComplexity: Cyclomatic complexity for search_record is too high. [10/7]

Line #171 – Convention: Metrics/MethodLength: Method has too many lines. [21/10]

Line #171 – Convention: Metrics/PerceivedComplexity: Perceived complexity for search_record is too high. [12/8]

Solución: Utilizamos el método de sustitución como se muestra a continuación

```
1 def search_record(resolution, ini_d, end_d, author)
2   get_initial_and_final_date
3   ini_d == '' ? ini_d = @initial_date : ''
4   end_d == '' ? end_d = @end_date : ''
5   if resolution != ''
6     ds = Document.by_resolution_like(resolution)
7     @arr = documents_array(ds)
8   elsif ini_d || end_d
9     ds = []
10    if author != ''
11      user_by_author = User.find(username: author)
12    end
13    if user_by_author
14      ds = Document.by_date_and_user(ini_d, end_d, user_by_a
15      uthor.id)
16    elsif (author == '')
17      ds = Document.by_date(ini_d, end_d)
18    end
19    @arr = documents_array(ds)
20  end
21  if @arr[0] == nil
22    @not_found_docs = "No se encontraron actas relacionadas
23    con su búsqueda.."
24  end
25 end
```

```
1 def search_record(resolution, ini_d, end_d, author)
2   get_initial_and_final_date
3   ini_d == '' ? ini_d = @initial_date : ''
4   end_d == '' ? end_d = @end_date : ''
5   if resolution != ''
6     @arr = documents_array(Document.by_resolution_like(resol
7     ution))
8   else
9     search_record_by_date_and_author(ini_d, end_d, author)
10  end
11
12  @not_found_docs = "No se encontraron actas relacionadas con su
13  búsqueda.." if @arr[0].nil?
14
15  end
16
17  def search_record_by_date_and_author(ini_d, end_d, author)
18    ds = []
19    author != '' and ds = User.find(:username => author) ? Doc
20    ument.by_date_and_user(ini_d, end_d, user_by_author.id) : Docu
21    ment.by_date(ini_d, end_d)
22    @arr = documents_array(ds)
23  end
24 end
```

Ofensas en el método tagg_user:

Line #204 – Convention: Metrics/AbcSize: Assignment Branch Condition size for tagg_user is too high. [<6, 19, 5> 20.54/17]

Line #204 – Convention: Metrics/MethodLength: Method has too many lines. [25/10]

Solución: También fue suficiente la utilización del método de sustitución

```
1 def tagg_user(dni, document)
2   user_tagg = User.find(dni: dni)
3
4   if user_tagg
5     not_tagged_in_document = Notification.find(user_id: user_tagg.id,
6     document_id: document.id)
7     if !not_tagged_in_document
8       document.add_user(user_tagg)
9     end
10  else
11    request.body.rewind
12    hash = Rack::Utils.parse_nested_query(request.body.read)
13    params = JSON.parse hash.to_json
14    string_dni = dni.to_s
15    not_user_tagg = User.new(
16      surname: string_dni,
17      category: "not_user",
18      name: string_dni,
19      username: string_dni,
20      dni: dni,
21      password: "not_user#{string_dni}",
22      email: "#{string_dni}@email.com")
23    if not_user_tagg.save
24      document.add_user(not_user_tagg)
25    else
26      [500, {}, "Internal server Error"]
27    end
28  end
29 end
```

```
1 def tagg_user(dni, document)
2   if User.find(:dni => dni)
3     if User.find(:dni => dni) && !Notification.find(:user_id => User.f
4     ind(:dni => dni).id, :document_id => document.id)
5       document.add_user(User.find(:dni => dni))
6     end
7   end
8   else
9     not_user_tagg = add_generic_user(dni.to_s, dni)
10
11   if not_user_tagg.save
12     document.add_user(not_user_tagg)
13   else
14     [500, {}, "Internal server Error"]
15   end
16 end
17
18 def add_generic_user(string_dni, dni)
19   User.new(
20     :surname => string_dni,
21     :category => 'not_user',
22     :name => string_dni,
23     :username => string_dni,
24     :dni => dni,
25     :password => "not_user#{string_dni}",
26     :email => "#{string_dni}@email.com"
27   )
28   if not_user_tagg.save
29     document.add_user(not_user_tagg)
30   else
31     [500, {}, "Internal server Error"]
32   end
33 end
```

Ofensas en post “/newUser”

Line #415 – Convention: Metrics/BlockLength: Block has too many lines. [44/25]

```

1 post '/newUser' do
2   request.body.rewind
3   hash = Rack::Utils.parse_nested_query
4   (request.body.read)
5   params = JSON.parse hash.to_json
6   pre_load_user =
7     User.find(dni: params["dni"])
8     exist_username =
9       User.find(username: params["username"])
10
11     exist_email =
12       User.find(email: params["email"])
13     if pre_load_user
14
15       if pre_load_user.category == "not
16         _user"
17
18         pre_load_user.update(
19           surname: params["surname"],
20           category: "user",
21           name: params["name"],
22
23           username: params["username"],
24           dni: params["dni"],
25
26           password: params["key"],
27           email: params["email"]
28         )
29         redirect "/"
30       else
31         [500, {}, "El usuario ya exist
32         e"]
33       end
34     else
35       if (exist_username)
36         @log_err = "El usuario ingresad
37         o ya existe"
38       elsif (exist_email)
39
40         @log_err = "El email ingresado ya existe"
41       else
42         user = User.new(
43           surname: params["surname"],
44           category: "user",
45           name: params["name"],
46           username: params["username"],
47           dni: params["dni"],
48           password: params["key"],
49           email: params["email"]
50         )
51         if user.save
52           redirect "/"
53         else
54           [500, {}, "Internal server
55           Error"]
56         end
57       end
58     end
59   erb:newUser
60 end

```

46 end

```

1 post '/newUser' do
2   request.body.rewind
3   hash = Rack::Utils.parse_nested_query
4   (request.body.read)
5   params = JSON.parse hash.to_json
6   pre_load_user =
7     User.find(:dni => params['dni'])
8     exist_username =
9       User.find(:username => params['usernam
10       e'])
11     exist_email =
12       User.find(:email => params['email'])
13     if
14       pre_load_user && (pre_load_user.category
15       == 'not_user')
16
17       update_pre_load_user(pre_load_user, param
18       s)
19       redirect '/'
20     else
21       if exist_username
22
23         @log_err = 'El usuario ingresado ya exist
24         e'
25       elsif exist_email
26
27         @log_err = 'El email ingresado ya exist
28         e'
29
30       else
31         user = add_new_user(params)
32
33         if user.save
34           redirect '/'
35
36         else
37           [500, {}, 'Internal server
38           Error']
39
40         end
41       end
42     end
43   erb :newUser
44 end
45
46 def add_new_user(data)
47   User.new(
48     :surname => data['surname'],
49     :category => 'user',
50     :name => data['name'],
51     :username => data['username'],
52     :dni => data['dni'],
53     :password => data['key'],
54     :email => data['email']
55   )
56 end
57
58 def update_pre_load_user(user, data)
59   user.update(
60     :surname => data['surname'],
61     :category => 'user',
62     :name => data['name'],
63     :username => data['username'],
64     :dni => data['dni'],
65     :password => data['key'],
66     :email => data['email']
67   )
68 end

```


Otro ejemplo con la utilización del método de extracción y longitud del método:

```
1 def set_menu
2   if user_not_logged_in?
3     redirect '/index'
4   end
5   case @current_user.category
6     when "superAdmin" then
7       @admin = "visible"
8       @superAdmin = "visible"
9     when "admin" then
10      @admin = "visible"
11      @superAdmin = "hidden"
12    else
13      @admin = "hidden"
14      @superAdmin = "hidden"
15    end
16    @usuario = session[:user_name]
17  end
```

```
1 def set_menu
2   redirect '/index' if user_not_logged_in?
3
4   case @current_user.category
5     when 'superAdmin'
6       show_super_admin
7     when 'admin'
8       show_admin
9     else
10      show_user
11    end
12    @usuario = session[:user_name]
13  end
14
15  def show_super_admin
16    @admin = 'visible'
17    @superAdmin = 'visible'
18  end
19
20  def show_admin
21    @admin = 'visible'
22    @superAdmin = 'hidden'
23  end
24
25  def show_user
26    @admin = 'hidden'
27    @superAdmin = 'hidden'
28  end
29 end
```

El resto de las ofensas fueron corregidas en forma automática por la aplicación, puede verse en el archivo subido al repositorio <https://github.com/nicolasgabrielr/ProjectGroup/tree/master/Docs> junto a todos los cambios efectuados.