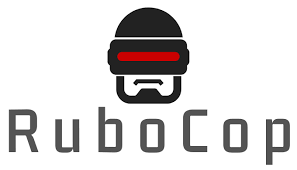
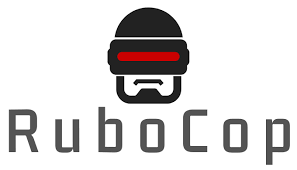
## línea horizontal



Ciclo De Refactorización

20/10/2020

**─**

Lopez Ayrton

Rafti Nicolas

Proyecto-A-Y-D-S.  
https://github.com/ayrton1907-art/Proyecto-A-Y-D-S.git  
5800 Río Cuarto(Córdoba).

# Objetivo:

*El objetivo fundamental de la refactorización es mejorar la estructura interna del software, los casos de refactorización presentados evidencian que efectivamente se logra «limpiar» el código fuente GeneXus, haciéndolo más fácil de entender y modificar+*

Hitos :

1. Realizar una reunión de equipo para identificar “Code Smells” de su proyecto, luego definir y planificar un nuevo sprint.
2. Instalar y ejecutar la aplicación Rubocop, Observar el resultado del análisis y en particular el de las métricas resultantes.
3. Corregir las ofensas de la clase app.rb (resguardar una copia original para comparar), utilizando la corrección automática provista por la herramienta.

# Hitos

## Planificación Sprint e identificación de tareas.

Por medio de la metodología Scrum, se realizó una nueva reunión con los 3 integrantes del grupo, para organizar un nuevo Sprint, en donde se identificaron las actividades críticas para poder avanzar de una forma más eficiente, organizada y ordenada.

Para esto se empezó con la lectura de la documentación necesaria para poder realizar el Sprint.

Luego de eso se identificaron de forma conjunta las las nuevas tareas a realizar en el Sprint

Estas fueron:



* Luego de identificadas las tareas se continuó con la estimación temporal de las mismas por medio de una herramienta llamada planitpoker. Que permite que cada integrante indique una unidad que él crea necesaria para realizar cada tarea, sin ser influenciado por las opiniones de los demás compañeros hasta después de votar el tiempo de la actividad, Luego de que cada participante del equipo estimó un tiempo se inicia una discusión para llegar a un acuerdo escuchando las razones de los demás de su estimación.
* Esto se repite para cada actividad o tarea del Sprint.

En la imagen anterior se puede observar que cada actividad ya fue estimada y confirmada con una unidad de tiempo.

Para estimas se utilizaron tarjetas basadas en la metodología Scrum:



Cada número indica una o más unidades de tiempo.

La tarjeta de interrogación indica un tiempo inestimable, ya que por alguna razón esta no puede ser definida, en nuestro caso se decidió usarla para las actividades que pensaramos que duraran mas que solo un Sprint. como por ejemplo en nuestro caso, en donde el cambio completo de interfaz duro 4 Sprint.

La Tarjeta que tiene la taza de café, es para tomar un descanso de 5 minutos en la planificación.

Al terminar esto, se procede a elegir a un Scrummaster que será el encargado de distribuir estas tareas de la forma más adecuada que crea, asi como tambien tiene la responsabilidad de mantener al equipo motivado, hacer un seguimiento continuo de las actividades que se realizan, y crear un análisis de riesgo, por si surge algún problema que complique alguna actividad crítica en el Sprint.

Para elegir el ScrumMaster se utiliza un software web, que define aleatoriamente a un miembro del equipo como ScrumMaster, este software se llama fluky.

|  |
| --- |
| *Esta elección de ScrumMaster solo se da porque es un proyecto de estudio y aprendizaje, en un proyecto de otra índole, este ScrumMaster será una persona que tenga un rol de Organización y planeación de un Proyecto. (Esto no implica que un programador no pueda ser Scrum Master)* |

En esta Sprint (n° 4) el ScrumMaster es Lopez Ayrton

* Lopez junto con las opiniones de los demás integrantes, Rafti Nicolas y Bortol Juan, se encarga de dividir las tareas a cada uno de los integrantes y de empezar a completar el cronograma en el GANTT del proyecto.
* El análisis de riesgo que se hace, consistirá en poner un tiempo estimado en el que se debe informar al equipo, en el cual se aclara que esta actividad no podría efectuarse en el tiempo estimado, para esto la persona que esté más avanzada o que tenga alguna actividad con holgura , se sumará a trabajar también en esa actividad para poder llegar a tiempo, esto siempre y cuando la actividad a retrasarse este en el camino crítico del sprint.

|  |
| --- |
| *En este Sprint no se tiene mucho margen para organizar un retraso u imprevisto.*  *Por eso cada actividad tiene una estimación, pero cada integrante del equipo, deberá terminarla lo antes posible y seguir con alguna otra actividad para asegurar un mejor funcionamiento del análisis de riesgo definido anteriormente.*  *Al inicio del Sprint, uno de los miembros del equipo Juan Bortol no seguirá formando parte del proyecto, esto implica organizar nuevamente el GANTT y una dedicación de tiempo mucho mayor que la estimada para cada uno de lo dos integrantes restantes, por ese mismo motivo dos actividades deciden dejarse para el próximo Sprint, estas actividades son:*   * *Realizar un manual de usuario para los administradores que se encargan del funcionamiento del software.* * *Realizar otras métricas que permitan identificar de una manera diferente distintas mejoras a implementar en el software.* |

Esta nueva organización eja el GANTT organizado de esta forma:



* Al finalizar la reunión se carga toda la información en el Pivotal Tracker del proyecto, para tener un seguimiento más continuo e interactivo del [proyecto](https://www.pivotaltracker.com/n/projects/2470689).



## Instalar la gema de Rubocop en el Proyecto

*Proyecto-A-Y-D-S/Gemfile*

|  |
| --- |
| ruby '2.6.4' source 'https://rubygems.org'  gem 'pg' gem 'rubocop' gem 'sequel' gem 'sinatra', '2.0.7' gem 'sinatra-contrib' gem 'sinatra-websocket' gem 'thin'  group :development do  gem 'dotenv', '2.7.5'  gem 'rerun' end |

Luego de la instalación se continua con la ejecucion de rubocop:

Conectarse al sitio o contenedor o que contenga el servidor web

servidor de la web en un contenedor de docker.

|  |
| --- |
| root@root:~/Documentos/Proyecto-A-Y-D-S$ sudo docker-compose up --build  WARNING: Found orphan containers (proyecto-a-y-d-s\_cabin\_db\_1) for this project. If you removed or renamed this service in your compose file, you can run this command with the --remove-orphans flag to clean it up.  Starting proyecto-a-y-d-s\_db\_1 ...  Starting proyecto-a-y-d-s\_web\_1 …  .  .  .  web\_1 | Rails backports are deprecated. |

|  |
| --- |
| root@root:~/Documentos/Proyecto-A-Y-D-S$ sudo docker ps  CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  09b5cca3875a proyecto-a-y-d-s\_web "bash -c 'bundle che…" 8 hours ago Up About a minute 0.0.0.0:9292->9292/tcp proyecto-a-y-d-s\_web\_1  089986fddc1b postgres "docker-entrypoint.s…" 2 days ago Up About a minute 5432/tcp proyecto-a-y-d-s\_db\_1  ayrton@ayrton:~/DocumentosProyecto-A-Y-D-S$ sudo docker exec -it 09b5cca3875a /bin/bash  root@09b5cca3875a:/usr/src/app# rubocop  .  .  .  23 files inspected, 747 offenses detected, 649 offenses auto-correctable  root@867fd3b1d20a:/usr/src/app# rubocop -a  Inspecting 23 files  .  .  .  23 files inspected, 124 offenses detected, 36 offenses auto-correctable  root@ccb35d6f1780:/usr/src/app# |

## Solucion de las primeras ofensas de Rubocop

Ofensas fueron del estilo:

Style:

|  |
| --- |
| app.rb:39:17: C: Style/RedundantParentheses: Don't use parentheses around a method call.  @page = (request.path\_info)  ^^^^^^^^^^^^^^^^^^^ |

En donde no envolver con paréntesis la llamada a un método como en este caso.

|  |
| --- |
| @page = request.path\_info |

Otro tipo de error fue el siguiente:

|  |
| --- |
| pp.rb:265:25: C: Style/StringLiterals: Prefer single-quoted strings when you don't need string interpolation or special symbols.  @page\_before\_name = "Documentos" |

Este tipo de errores fue solucionado por el rubocop de manera automática con el -a.

|  |
| --- |
| @page\_before\_name = 'Documentos' |

|  |
| --- |
| app.rb:273:27: C: Style/HashSyntax: Use the new Ruby 1.9 hash syntax.  erb :modify\_document, :layout =>@layoutEnUso |

Errores como el anterior también fueron solucionados de manera automática por el rubocop.

|  |
| --- |
| erb :modify\_document, layout: @current\_layout |

Un error muy común en el app.rb fue la manera de escribir los for, que si bien estaban bien escritos, habia una forma mas prolija y correcta de hacerlo:

|  |
| --- |
| app.rb:331:6: C: Style/SafeNavigation: Use safe navigation (&.) instead of checking if an object exists before calling the method.  @notification\_delete && @notification\_delete.each do |element| ...  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ |

Y esta forma era la siguiente:

|  |
| --- |
| @notification\_delete&.each do |element| |

Layout:

|  |
| --- |
| app.rb:49:28: C: Layout/ExtraSpacing: Unnecessary spacing detected.  if !session[:type] && @urlAdmin.include?(request.path\_info)  ^ |

En donde se observa un espacio en blanco que no debería ir.

|  |
| --- |
| if !session[:type] && @urlAdmin.include? (request.path\_info) |

|  |
| --- |
| app.rb:317:45: C: Layout/SpaceAfterComma: Space missing after comma.  cat = Category.new(name: params[:name],description: params[:description] ) |

|  |
| --- |
| app.rb:341:35: C: Layout/SpaceAroundOperators: Surrounding space missing for operator =.  @notification.documento =(Document.find(id: @not.document\_id)) |

Los 2 errores mostrados anteriormente fueron solucionados de manera automática por el comando rubocop -a:

|  |
| --- |
| cat = Category.new(name: params[:name], description: params[:description]) |

Y el otro:

|  |
| --- |
| @notification.documento = (Document.find(id: @not.document\_id)) |

Lint

|  |
| --- |
| app.rb:84:121: C: Layout/LineLength: Line is too long. [164/120]  @newUserName = User.new(name: params[:name],surname:  params[:surname],dni: params[:dni],email: params[:email],password: params[:password],rol: params[:rol])  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ |

En este caso el largo de la línea es de 164 y rubocop no dice que debería ser de 120 o menos, pero en la documentación de ruby dice que esta debe ser de 80 caracteres.

Por lo que se decidió respetar la cantidad de 80 caracteres por línea como se solicita en la documentación de Ruby.

Para esto la solución fue:

|  |
| --- |
| @new\_user = User.new(  name: params[:name],  surname: params[:surname],  dni: params[:dni],  email: params[:email],  password: params[:password],  rol: params[:rol]  ) |

En donde también se soluciono el problema de:

|  |
| --- |
| app.rb:84:9: C: Naming/VariableName: Use snake\_case for variable names.  @newUserName  ^^^^^^^^^^^^ |

Sin embargo, y por convención los métodos también se deben escribir con snake\_case en vez de camelCase por lo que también se marcó como error las funciones cuyo nombre estaba escrito con camelCase, un ejemplo de ello es el siguiente

|  |
| --- |
| app.rb:423:7: C: Naming/MethodName: Use snake\_case for method names.  def notifyUser(user, message) #Funciona |

Metrics

|  |
| --- |
| app.rb:224:3: C: Metrics/BlockLength: Block has too many lines. [37/25]  post '/create\_document' do #FUNCIONA ...  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^ |

En donde la solución pensada para esta, fue modularizar por medio de definiciones que hicieras las actividades más puntuales y que pudieran ser repetidas en distinto métodos para que se reutilice.

Esto se realizó en la parte de modularización junto a otras funciones que poseían el mismo problema, a continuación se muestran otro ejemplo.

|  |
| --- |
| app.rb:431:3: C: Metrics/MethodLength: Method has too many lines. [28/10]  def newDat(date) ...  ^^^^^^^^^^^^^^^^ |

Algunas de estas ofensas fueron solucionadas de manera automática, como por ejemplo la falta o sobra de espacios en blanco, la reorganización de la estructura “if”, así como también se encargó de dejar toda la estructura de un “each” en una línea si era posible .

## Branch

Para estas mejoras, se crearon ramas particulares en la plataforma [GIT](https://github.com/) donde se encuentra el proyecto.

* [rubocop\_auto-correctable](https://github.com/ayrton1907-art/Proyecto-A-Y-D-S/tree/rubocop_auto-correctable) = En esta rama se corrió el robocop y se hicieron las correcciones automáticas.
* [Cambios\_nombres\_variables](https://github.com/ayrton1907-art/Proyecto-A-Y-D-S/tree/Cambios_nombres_variables) = En esta rama se paso el nombre de variables y métodos a de camelCase a snake\_case cumpliendo con los estilos de Ruby.
* [largo\_lineas\_codigo](https://github.com/ayrton1907-art/Proyecto-A-Y-D-S/tree/largo_lineas_codigo) = En esta rama se solucionaron los largos de líneas que superan los 80 caracteres, esto generalmente se daba en las llamada a las definiciones, que se soluciono organizando de una forma vertical cada parámetro pasado a la misma.
* [refactorizacion\_final](https://github.com/ayrton1907-art/Proyecto-A-Y-D-S/tree/refactorizacion_final) = Rama que tuvo las modalizaciones finales del proyecto, esta rama fue el último merge.

La rama [proyecto\_sin\_refactorizar](https://github.com/ayrton1907-art/Proyecto-A-Y-D-S/tree/proyecto_sin_refactorizar) contiene la versión no refactorizada del código.

La ofensa pendiente que quedó es:

|  |
| --- |
| root@09b5cca3875a:/usr/src/app# rubocop app.rb Inspecting 1 file C Offenses: app.rb:9:1: C: Metrics/ClassLength: Class has too many lines. [493/100] class App < Sinatra::Base ... ^^^^^^^^^^^^^^^^^^^^^^^^^ 1 file inspected, 1 offense detected root@09b5cca3875a:/usr/src/app# |

Por falta de tiempo esta ofensa no pudo solucionarse, Para solucionarla la idea es hacer uso de nuevas subclases en donde estas puedan contener un conjunto de definiciones y métodos que se vinculen, de esta forma se podría reducir de forma considerable la cantidad de líneas de la clase principal par que esta no supere las 100.

Esto implicaría revisar método por método al igual que las definiciones ya que el pasaje de información entre las clases debe ser darse de la forma adecuada para que ésta no cause problemas ni tampoco penaliza la refactorización ya efectuada hasta el momento, aparte de esto en el “app.rb”, en los test y modelos quedaron un total de 17 ofensas para solucionar.

# Ideas finales:

* Gracias a este trabajo el código se volvió más legible tanto para nosotros como para personas ajenas al

proyecto.

* Además que a medida que modularizábamos el código empezábamos a usar las convenciones de Ruby.
* Por otro lado, también se notó que refactorizar es un trabajo arduo, ya que implica revisar cada línea y hacer en algunos casos grandes cambios para respetar las convenciones.
* Esto sirvió para entender la gran importancia de seguir una convención y lo que esto implica, luego de comparar el código con la versión que no está factorizada encontramos grandes diferencias que notamos ahora, no era lo más correcto.
* También podemos notar la gran ayuda que implica tener una herramienta como rubocop, que permite automatizar solución de errores.

Links:

[Documento con las salidas de rubocop](https://drive.google.com/file/d/153sRabSl_qLeIKX05zMpyoz03l4St4PR/view?usp=sharing)

https://github.com/ayrton1907-art/Proyecto-A-Y-D-S

https://www.planitpoker.com/

<https://www.fluky.io/>

<https://www.pivotaltracker.com/n/projects/2470689>

<https://miro.com/app/board/o9J_kidvIGg=/>