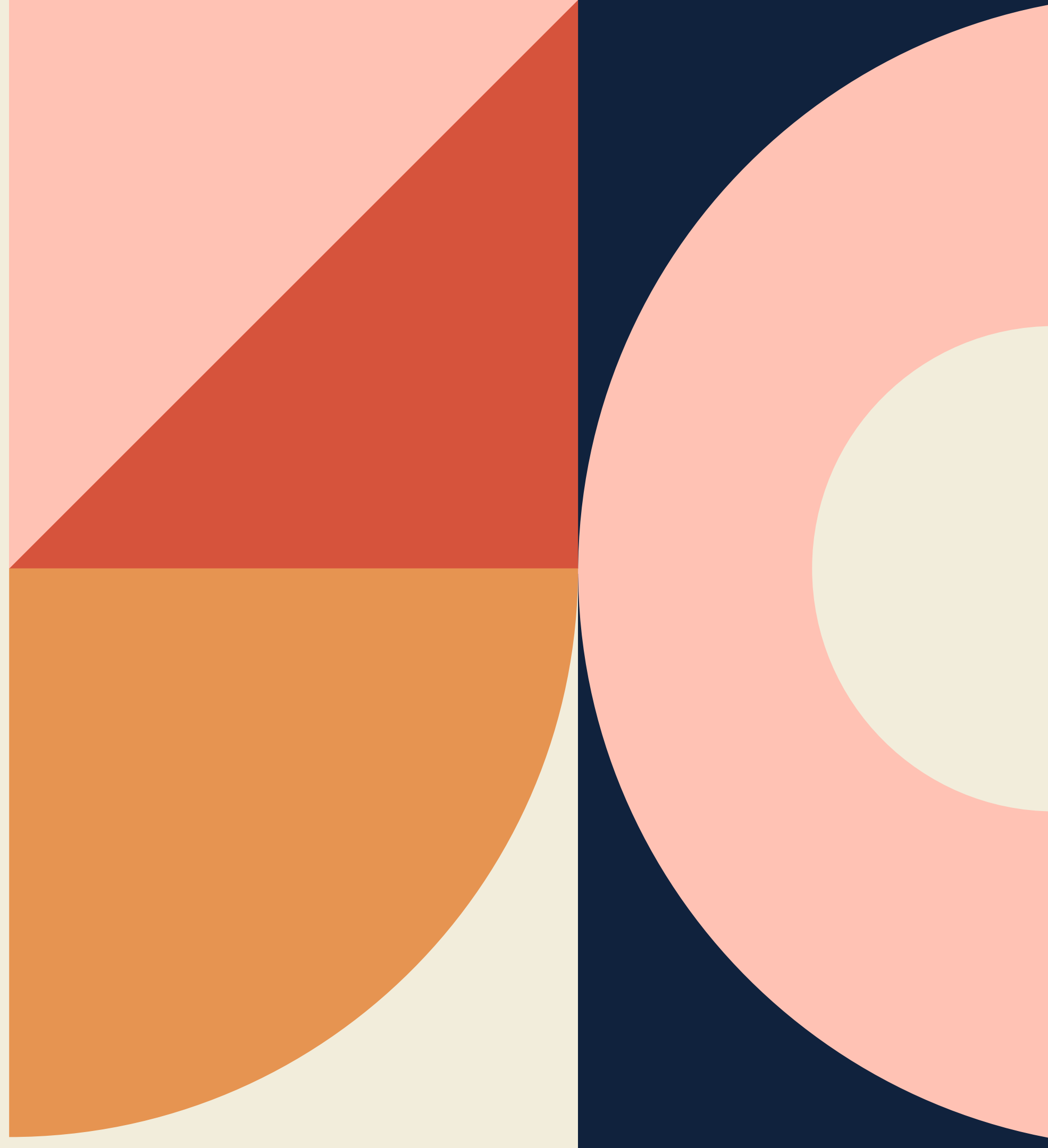


PRESENTACIÓN FINAL "TALLER"

23 de noviembre de 2021

Ingeniería de software



01

GRUPO 7: INTEGRANTES

- Baseggio, Axel Gabriel
- Fernández, Tomás
- Mascanfroni, Lucrecia

Ingeniería de software



02

ACTIVIDADES DESARROLLADAS

Ingeniería de software



Actividad 1) :

Cada equipo de trabajo deberá planificar y ejecutar un nuevo sprint de dos semanas comenzando el 31/08 y finalizando el 14/09 . Para ello deberá realizar las siguientes tareas:

- Identificar aquellas funcionalidades pendientes, incompletas o con fallas del proyecto realizado en Análisis y Diseño de Software (2021).
- Deberán describir las nuevas historias de usuario destinadas a para completar o corregir lo observado en el punto anterior:
 - ID: Identificador único de la funcionalidad o trabajo.
 - Descripción de la funcionalidad/requisito.
 - Priorización.(Muy alta, alta, media, baja)
 - Estimación del esfuerzo necesario. (por ahora definir la cantidad de horas puede insumir).
- Un miembro del equipo deberá asumir el rol de ScrumMaster.
- Definir la pila del Sprint (Sprint backlog) .
- Realizar la reunión de Planificación del Sprint (Sprint Planning).
- Realizar una planificación determinando la lista de tareas con sus respectivas dependencias y recursos asignados. Utilizar un diagrama de Gantt para su representación. Para construir el diagrama de Gantt pueden utilizar GantProject (<https://www.ganttproject.biz/>) o cualquier otra herramienta apropiada.
- Ejecutar el Sprint.
- Subir la documentación al repositorio git del proyecto.
- En la reunión de taller correspondiente al día 14/09 el ScrumMaster asignado expondrá el trabajo realizado.

LO QUE HICIMOS PARA LA RESOLUCIÓN DE ESTA ACTIVIDAD:

Realizamos una reflexión sobre la gestión del proyecto en base al cuatrimestre pasado.


ACTIVIDAD 2) :

05

Cada miembro de equipo deberá registrar una cuenta en Pivotal Tracker (<https://www.pivotaltracker.com/>). Luego uno de los integrantes del equipo deberá crear el proyecto e invitar al resto como members.

Para poder realizar el seguimiento de cada equipo, los docentes coordinadores del taller (marcelouva y aarsaute) solicitamos nos agreguen al proyecto como viewers.

- Cada equipo deberá realizar una simulación de la reunión de planificación de sprint (sprint planning meeting) utilizando la técnica de Planning Poker a fin de asignar a cada historia de usuario la estimación de esfuerzo en Story Points. Pueden utilizar una herramienta para esta actividad, por ejemplo <https://planningpokeronline.com/>
- Una vez estimadas las historias de usuario, deberán ser agregadas con sus respectivos atributos al proyecto creado en Pivotal Tracker en el panel Backlog .
- El equipo debe simular la ejecución de 2 sprints, para ello tendrán que setear la velocidad inicial del panel del sprint corriente (current iteration) con una cantidad suficiente de Story Points que permita la realización de todas las historias estimadas.

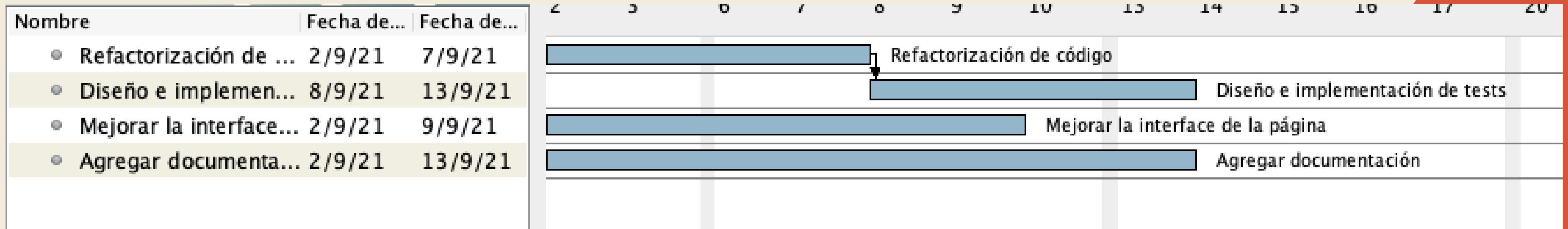


06

ACTIVIDADES QUE REALIZAMOS PARA LA RESOLUCIÓN DE ESTA ACTIVIDAD:

- Refactorización de código.
- Diseño e implementación de tests.
- Mejorar la interfaz de la página.
- Agregar documentación.

DIAGRAMA DE GANTT



ACTIVIDAD 3) :

Implementar el siguiente requerimiento:

Como usuario se desea conocer la cantidad de encuestas resultantes de una carrera en un rango de fechas.

Por ejemplo: "02/05/21" - "14/09/21" - Lic. en Computación

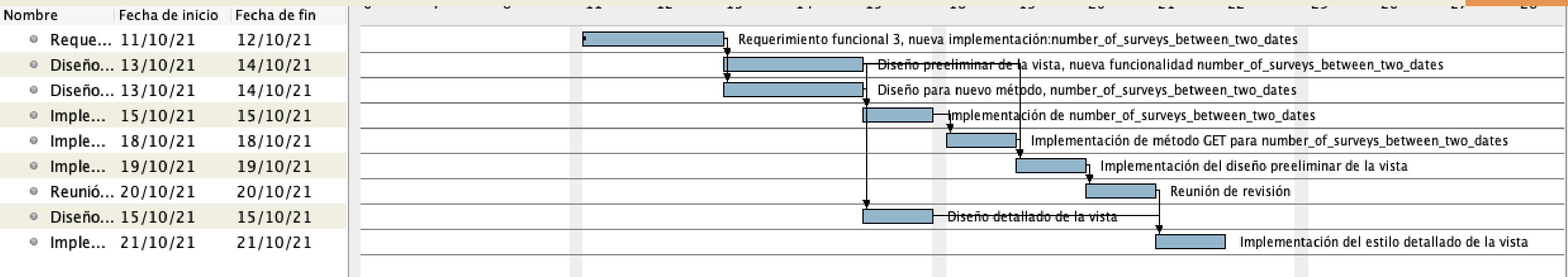
" Lic. en Computación - 58 encuestas "

PARA REALIZAR ESTA ACTIVIDAD TUVIMOS UNA REUNIÓN DONDE QUEDAMOS EN HACER LAS SIGUIENTES ACTIVIDADES:

09

- Diseño del método que da solución al nuevo requerimiento. (FT)
- Implementación del método. (FT)
- Implementación del método GET para la funcionalidad. (FT)
- Diseño preliminar de la vista para la funcionalidad. (BA)
- Implementación de la vista para la funcionalidad. (BA)
- Diseño detallado de la vista. (ML)
- Implementación del diseño detallado. (ML)

DIAGRAMA DE GANTT



ACTIVIDAD 4) :

CADA EQUIPO DEBERÁ REALIZAR LAS SIGUIENTES TAREAS:

1. DEL LIBRO “REFACTORING RUBY EDITION” LEER LOS CAPÍTULOS 3, 6, 7 Y 8.
[HTTP://WWW.R-5.ORG/FILES/BOOKS/COMPUTERS/LANGUAGES/RUBY/STYLE/JAY_FIELDS_SHANE_HARVIE-REFACTORING_RUBY_EDITION-EN.PDF](http://www.r-5.org/files/books/computers/languages/ruby/style/jay_fields_shane_harvie-refactoring_ruby_edition-en.pdf)
2. REALIZAR UNA REUNIÓN CON TODO EL EQUIPO, ANALIZAR EL PROYECTO IDENTIFICANDO POSIBLES “BAD SMELLS” JUNTO A LAS REGLAS DE REFACTORIZACIÓN QUE PODRÍAN APLICARSE PARA MEJORAR LA CALIDAD DEL CÓDIGO. DOCUMENTAR ESTE ANÁLISIS EN UN INFORME EN DONDE SE OBSERVE LOS “BAD SMELLS” DETECTADOS (JUSTIFICANDO) JUNTO A LAS REGLAS DE REFACTORIZACIÓN A APLICAR (JUSTIFICANDO).
3. GENERAR EL BRANCH “REFACTOR” A PARTIR DE LA RAMA “DEVELOP”, LUEGO APLICAR LOS CAMBIOS PROPUESTOS Y FINALMENTE (CON EL ACUERDO DE TODO EL EQUIPO) MERGEAR LOS CAMBIOS AL BRANCH “DEVELOP” NUEVAMENTE.
4. LEER EL DOCUMENTO [HTTPS://GITHUB.COM/RUBOCOP-HQ/RUBY-STYLE-GUIDE](https://github.com/rubocop-hq/ruby-style-guide) CON EL OBJETIVO DE CONOCER LA GUÍA DE ESTILOS RUBY. INSTALAR Y EJECUTAR LA APLICACIÓN RUBOCOP ([HTTPS://RUBOCOP.ORG](https://rubocop.org)) SOBRE SU PROYECTO. CORREGIR LAS OFENSAS DETECTADAS (DE MANERA AUTOMÁTICA O MANUAL). ANALIZAR EL RESULTADO DEL ANÁLISIS REALIZADO POR RUBOCOP (PARTICULARMENTE EL DE LAS MÉTRICAS) E INCORPORARLO AL INFORME SOLICITADO.

En la reunión se identificaron algunos de los siguientes bad smells y se aplicaron los siguientes métodos para tratarlos:

En el archivo REQF3:

- Renombramiento de la vista REQF3 por `number_of_survey_between_two_dates`.
- Renombramiento de las variables `firstDate` y `lastDate` por `initial_date` y `final_date`.

13

En el archivo app:

- En el método POST de responses se reemplazo el ciclo:

```
params[:question_id].each do |q_id|  
    response = Response.new(choice_id: params[q_id], survey_id: survey.id, question_id: q_id)  
    response.save  
end
```

Aplicando el método de cierre de colección:

```
params[:question_id].map { |question_id| Response.create(choice_id: params[question_id],  
survey_id: survey.id, question_id: question_id) }
```

En el modelo survey:

- El nombre del método **result**, que pondera las carreras dadas las respuestas de una survey, se cambia por **survey_result**.
- En el método survey_result:

```
def survey_result(careers)
  career_weights = {}
  careers.each do |career|
    career_weights[career] = 0
  end

  responses.each do |response|
    response.choice.outcomes.each do |outcome|
      career_weights[outcome.career] += 1
    end
  end

  career_weights.sort_by { |career,outcomes| outcomes }
end
```

Aplicando el método de cierre de colección:

```
def survey_result(careers)
  career_weights = {}
  careers.map { |career| career_weights[career] = 0 }

  responses.map { |response|
    response.choice.outcomes.map { |outcome|
      career_weights[outcome.career] += 1 } }

  career_weights.sort_by { |career,outcomes| outcomes }
end
```

Aplicando el método de extracción:

```
def survey_result(careers)
  evaluarte_responses(careers).sort_by { |career,
    outcomes| outcomes }
end

def evaluarte_responses(careers)
  career_weights = {}
  careers.map { |career| career_weights[career] = 0 }
  responses.map { |response|
    response.choice.outcomes.map { |outcome|
      career_weights[outcome.career] += 1 } }
end
```


En el modelo career:

- Se cambio el nombre del método *s_for_dates* por *number_of_surveys_between_two_dates*.

Una vez hecho esto se utilizó la herramienta *RUBOCOP* para analizar y corregir automáticamente los bad smells que quedaron en el proyecto.

ACTIVIDAD 5) :

Cada equipo de trabajo deberá refactorizar el código fuente de su proyecto a fin de implementar una arquitectura Model View Controller.

Deberán crear las clases Controller y Service que requiera su proyecto. El proceso de refactorización deberá realizarse en una rama nueva generada a partir de la rama 'develop'. Luego que el código esté testeado y que el equipo apruebe la modificación deberá ser mergeado a la rama 'develop'.

Lo que se realizó para aplicar el patrón MVC fueron las siguientes modificaciones:

- Archivo **app.rb** antes:

19

```
1  # frozen_string_literal: true
2
3  require './models/init.rb'
4
5  # Endpoint of the app
6  class App < Sinatra::Base
7    # GET method of root
8    get '/' do
9      erb :landing
10    end
11    # End of GET method of root
12
13    # GET method of careers
14    get '/careers' do
15      @careers = Career.all
16      erb :careers_index
17    end
18    # End of GET method of careers
19
20    # GET methods all surveys given id of the career and two dates.
21    get '/surveys' do
22      @result = {}
23      @result = Career.find(id: params[:careerId]).number_of_surveys_between_two_dates(params[:initial_date], params[:final_date]) if params[:careerId] && params[:initial_date] && params[:final_date]
24      @careers = Career.all
25      erb :number_of_surveys_between_two_dates
26    end
27
28    # POST methods of surveys
29    post '/surveys' do
30      @survey = Survey.new(name: params[:name])
31
32      if @survey.save
33        @questions = Question.all
34        erb :surveys_index
35      else
36        redirect '/'
37      end
38    end
39    # End of POST method of surveys
40
41    # POST method of responses
42    post '/responses' do
43      survey = Survey.find(id: params[:survey_id])
44      params[:question_id].map { |question_id| Response.create(choice_id: params[question_id], survey_id: survey.id, question_id: question_id) }
45      @result = survey.survey_result(Career.all)
46      @career_result = @result[@result.size - 1][0]
47      survey.update(career_id: @career_result.id)
48      erb :outcome_index
49    end
50    # End of POST method of responses
51  end
52
```

- Archivo **app.rb** después:

20

```
1 # frozen_string_literal: true
2
3 require './models/init.rb'
4 require './controllers/career_controller.rb'
5 require './controllers/survey_controller.rb'
6
7 # Endpoint of the app
8 class App < Sinatra::Base
9   use CareerController
10  use SurveyController
11
12  # GET method of root
13  get '/' do
14    erb :landing
15  end
16  # End of GET method of root
17 end
```

- Además, se crearon los controllers.

• Survey controller

22

```
1  # frozen_string_literal: true
2
3  require 'sinatra/base'
4
5  # Class SurveyController
6  class SurveyController < Sinatra::Base
7    configure :development, :production do
8      set :views, settings.root + '/../views'
9    end
10
11    # GET methods all surveys given id of the career and two dates.
12    get '/surveys' do
13      @result = {}
14      @result = Career.find(id: params[:careerId]).number_of_surveys_between_two_dates(params[:initial_date], params[:final_date]) if params[:careerId] && params[:initial_date] && params[:final_date]
15      @careers = Career.all
16      erb :number_of_surveys_between_two_dates
17    end
18
19    # POST methods of surveys
20    post '/surveys' do
21      @survey = Survey.new(name: params[:name])
22
23      if @survey.save
24        @questions = Question.all
25        erb :surveys_index
26      else
27        redirect '/'
28      end
29    end
30    # End of POST method of surveys
31
32    # POST method of responses
33    post '/responses' do
34      survey = Survey.find(id: params[:survey_id])
35      params[:question_id].map { |question_id| Response.create(choide_id: params[question_id], survey_id: survey.id, question_id: question_id) }
36      @result = survey.survey_result(Career.all)
37      @career_result = @result[@result.size - 1][0]
38      survey.update(career_id: @career_result.id)
39      erb :outcome_index
40    end
41    # End of POST method of responses
42  end
43
```

- Career controller

```
1  # frozen_string_literal: true
2
3  require 'sinatra/base'
4
5  # Career controller.
6  class CareerController < Sinatra::Base
7    configure :development, :production do
8      set :views, settings.root + '/../views'
9    end
10
11    # GET method of careers
12    get '/careers' do
13      @careers = Career.all
14      erb :careers_index
15    end
16    # End of GET method of careers
17  end
```


Una vez hecho esto se utilizó nuevamente la herramienta RUBOCOP para analizar y corregir automáticamente los bad smells que quedaron en el proyecto.

Fin de la presentación

¡Muchas gracias!

Baseggio, Axel Gabriel

Fernández, Tomás

Mascanfroni, Lucrecia