Decisiones de diseño para exportar en PDF
Entrega-2

alumna: Ana Cossio

- El formato elegido para las notas es markdown
- Para poder exportar a pdf los archivos .rn en el formato markdown
  - se tuvo que instalar:
    - ✓ sudo apt install -y python3-pip
    - ✓ sudo apt install -y build-essential libssl-dev libffi-dev python3-dev
    - ✓ pip3 install md2pdf

> (utilice md2pdf debido que algunas gemas como kitPDF a la hora de guardar el archivo pdf, me tiraba el error: **EACCES: permission denied)** y no pude solucionar dicho problema

Para exportar las notas persistidas en el disco duro agregue dos clases en Commands::Books::**ReportOf** (para que reporte todas las notas de las carpetas que se le indique como parámetro) y Commands::Notes::**Report** (para reportar una nota en particular)

Se modifico el archivo commands.rb, donde se añadió:
prefix.register 'reportOf', Books::ReportOf → para Commands::Books
prefix.register 'report', Notes::Report → para Commands::Notes

❖ Exportar una nota en particular:

```
Commands:
    ruby bin/rn notes report title --global  #indica que {title} es del libro global
    ruby bin/rn notes report title --book name_book  #name_book es el nombre del libro
                                                      #donde se encuentra la nota title
```

Cuando se ejecute unos de los comandos y los argumentos sean correctos, se creará una instancia de Models::Note, dicha instancia cuenta con el método report, que se puede apreciar en la siguiente imagen.

```ruby
def path_all_pdf
      "#{PATH}/#{title}_#{book.name}.pdf"
end

def report(save_pdf=nil)
 self.persists?
 save_pdf = save_pdf.nil? ? (self.path_full_changed_pdf) : (self.path_all_pdf)
      `"#{MD2PDF}" "#{self.path_full}" "#{save_pdf}"`
end
```

El método tiene un parámetro opcional, si no recibe ningún parámetro, la nota sabe que debe exportarse como pdf en la misma carpeta donde se encuentra.
Esto es así para aprovechar y reutilizar el código más adelante.

❖ Exportar múltiples notas de un cuaderno en particular:

Commands:
ruby bin/rn books reportOf **--global** #Exporta todas las notas del libro global
ruby bin/rn books reportOf **--book** name_book #exporta todas las notas del book "name_book"
ruby bin/rn books reportOf --all #Exporta toda las notas de los books que hay en el disco

Se creará una instancia del Modelo Book, este llamara a su método **report_my_notes.**
Donde obtendrá sus notas correspondientes y sólo exportará a pdf aquellas notas que no se encuentren vacías, luego a cada nota se le pedirá que se exporte a pdf.

```ruby
def my_notes_not_empty
      (self.notes).reject{ |note| (Models::Note.new(note,self.name)).empty? }
end

def report_my_notes(all=nil)
 self.my_notes_not_empty.each {|note|
               (Models::Note.new(note,self.name)).report(all)}
end
```

❖ **Exportar todas las notas de todos los cuadernos:**

```
Commands:
      ruby bin/rn books reportOf --all  #Exporta toda las notas de los books que hay en el disco
```

Cuando se opta por esta opción se llama a la clase del modelo Book, ya que este conoce todas sus instancias. Esta clase utiliza su método de clase **report_all_notes:**

```
def self.all_instancias_book
   names_of_books=Models::Book.all_books
   names_of_books.map {|book| (Models::Book.new(book))}
end

 def self.report_all_notes
  Models::Book.all_instancias_book.each{ |book| book.report_my_notes(true)}
 end
```

En report_all_notes obtiene todas los libros que se encuentran en la carpeta **.my_rns**, una vez completado esto, a cada libro se le pedirá que exporte sus notas con el método report_my_notes, pero esta vez se le pasara un parámetro en “True”, de esta manera las notas saben que deben hacer el reporte en PDF en la carpeta principal **.my_rns**