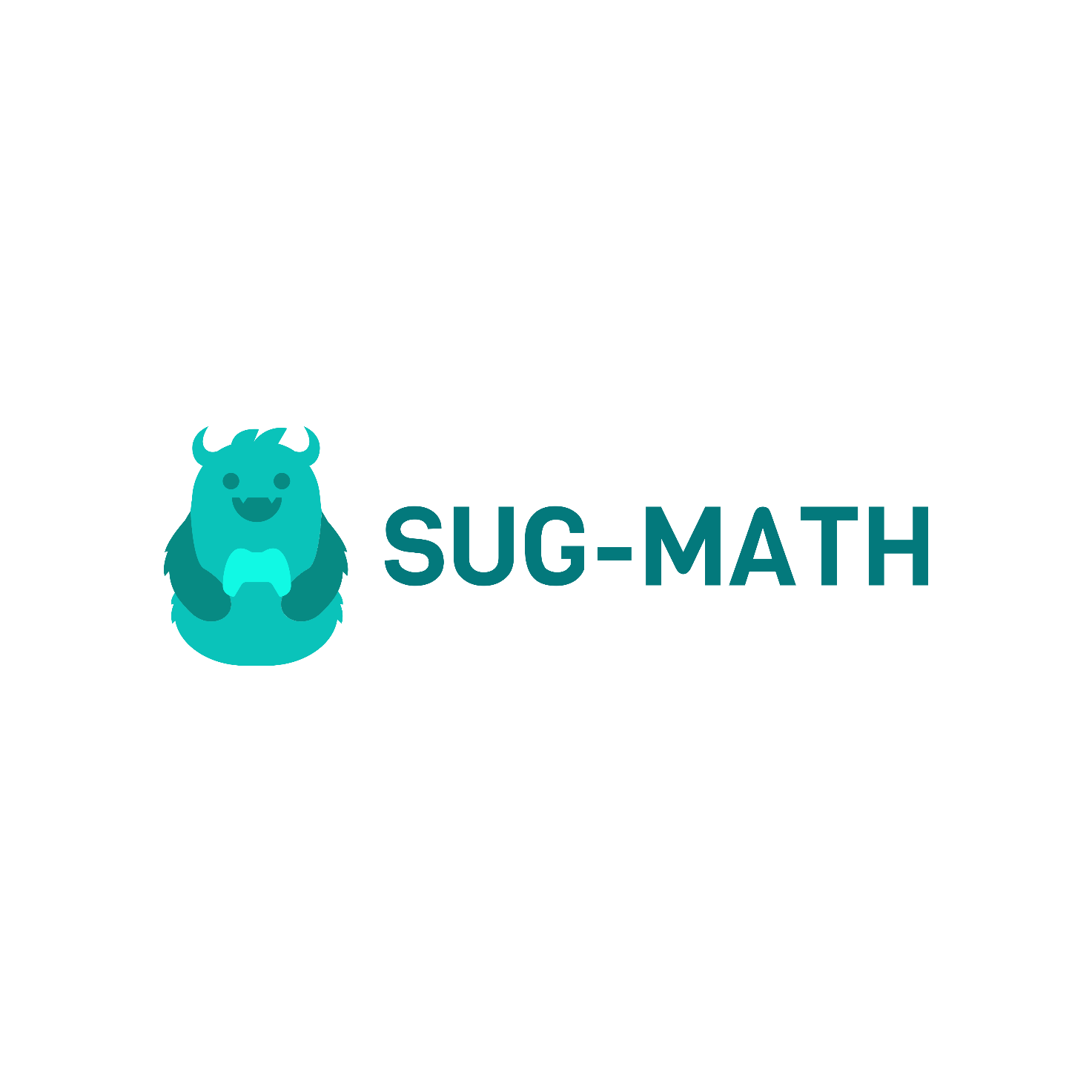
****

Estudiantes: Buchieri Giovanni

Guzman Alexis Uriel

Vázquez Santiago

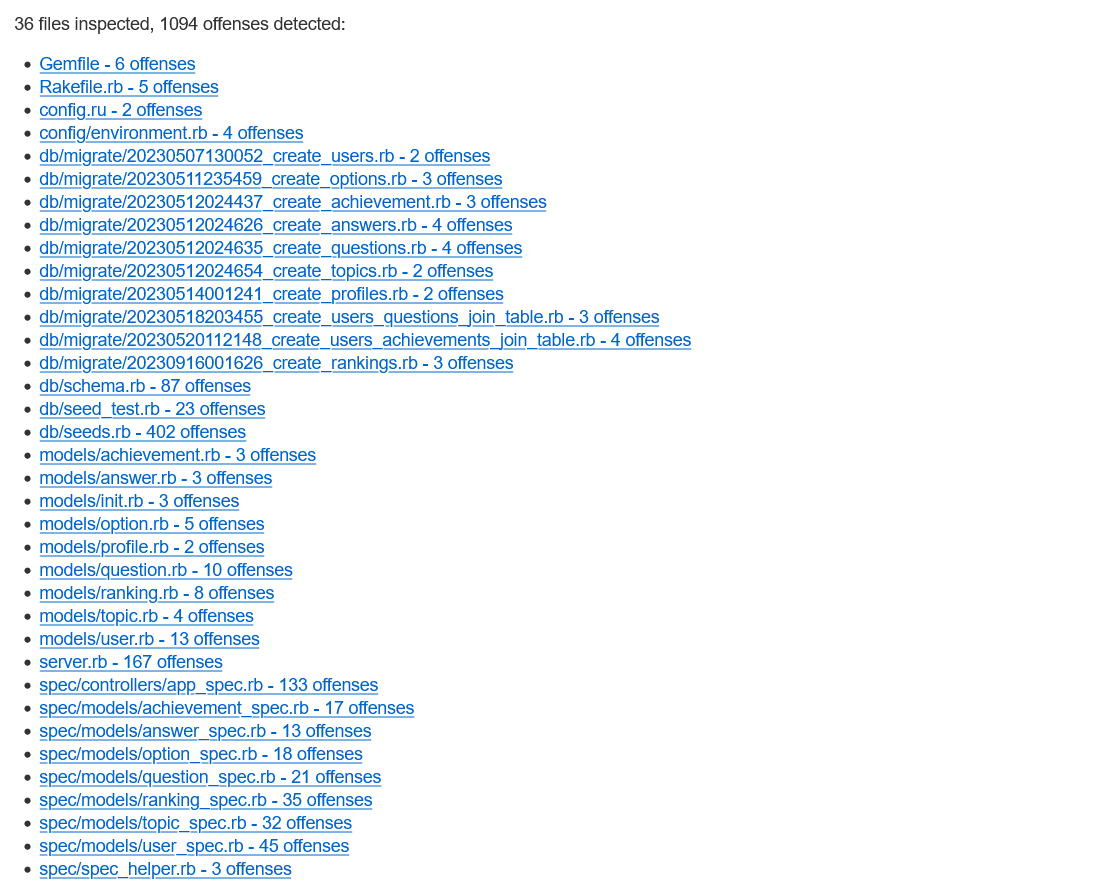
INFORME DE refactirización

Proyecto: SUG-MATH

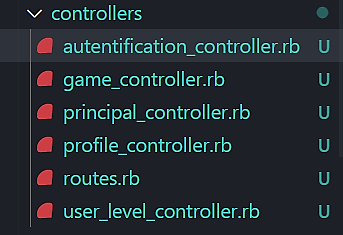
Actividad 4: Refactorización de código

Ante la falta de experiencia presente por parte del equipo en el desarrollo de aplicación en cuestión, el diseño, el modelado y la organización de clases, como asi también de componentes de la aplicación se vieron afectados de manera negativa. Como problema principal que se afrontara en el siguiente informe, se tiene la falta de un apartado de controladores, y una mala distribución de responsabilidades en cuanto al diseño del controlador principal Server.rb, por ende, nos abocaremos en refactorización de este aspecto para una mejor legibilidad, mayor comprensión por parte del lector y la incorporación/corrección de ofensas en los estándares de diseño, propias del lenguaje Ruby.

**Primera vista del proyecto:**

Como se puede apreciar en la siguiente captura de la primera ejecución de la gema rubocop, se inspeccionaron 36 archivos con extensiones de Ruby, entre ellos como se ve en la captura extraída de la terminal, se encontraron 1094 ofensas a los estándares previamente mencionados, de los cuales 1009 son auto corregibles por la herramienta. Sin dejar de lado el gran numero, vale aclarar que la mayoría de los posibles cambios que son detectados, se encuentran en el archivo Seed.rb, es decir, el diseño principal tiene que sufrir mejoras, pero no representa el peor caso en el que se podría encontrar.

**Separación de controladores:**

Como primera acción de refactoreo, se dividió el controlador principal (archivo server.rb) en sub controladores, que contienen un subconjunto de las rutas del mismo. Esto ayuda a la distinción de diferentes funciones de la ampliación, como para el testeo de las mismas. El porqué de este cambio es claro, *Code Smell: Large Class*, todo el código estaba contenido en un solo archivo, y la mantención de este era extremadamente costoso, tomaba tiempo encontrar el método o sección que se quería testear, etc. A continuación, se presentan el cambio realizado con su ya mencionada división.

Además, para facilitar posteriores correcciones, estos controladores se escribieron de la manera mas correcta posible, intentando omitir todo tipo de imperfección en las convenciones y en la sintaxis del lenguaje Ruby. Este trabajo se basó firmemente en eliminar los siguientes *Code Smells* de los controladores:

**Re ejecución de la herramienta rubocop:**

En este punto tras haber hecho una división y corrección tan exhaustiva sobre el principal problema que presentaba el sistema, la clase

**Ejecución de la herramienta de auto reparado de rubocop:**

Tras ejecutar el reparado automático de desperfectos,

***Sección de Olores encontrados:***

1. Large Class
2. Long Method
3. Divergent Change \*
4. Dispensable (cod, unnecessary comments, code dead, Speculative generality
5. Lazy Class