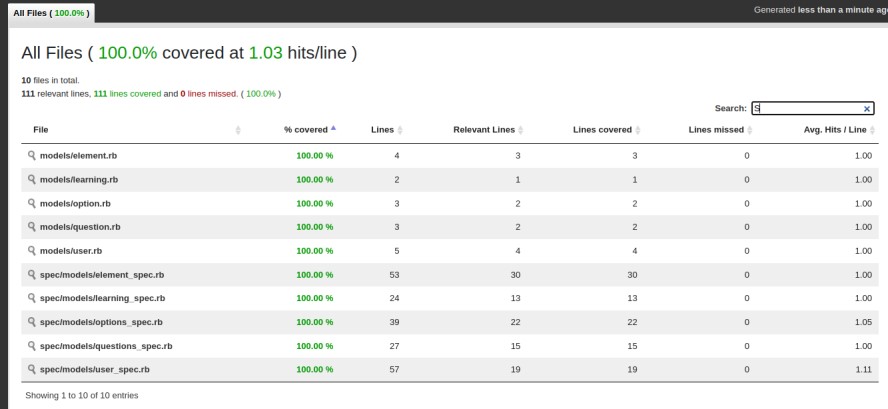
**Informe Ingeniería de Software**

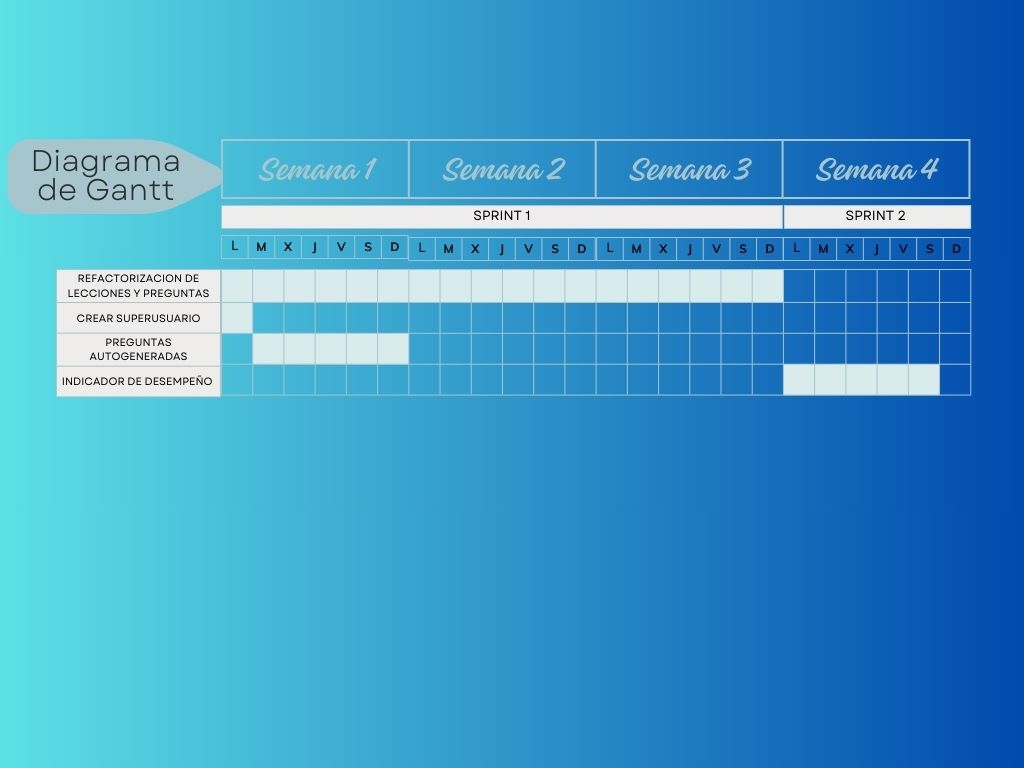
Actividad 1 – Test de Cobertura



En la actividad 1 tuvimos que ejecutar SimpleCov para ver la cobertura de los test que habíamos realizado y estos nos arrojaron buenos resultados, por lo que al principio no tuvimos que modificar el código para que los resultados de estos test fueran correctos.

Además de esto tuvimos que revisar las funcionalidades incompletas y pendientes del taller de Análisis y Diseño de Sistemas y llegamos a la conclusión de que llegamos a cumplir con los plazos de entrega por lo que no tuvimos ninguna funcionalidad incompleta y/o pendiente.

Actividad 2 – Gestión de tareas



Propusimos tareas nuevas y estimamos el tiempo que nos llevaría realizarlas utilizando Planning Poker, configuramos y vinculamos Pivotal Tracker con GitHub para llevar el registro del avance estas actividades.

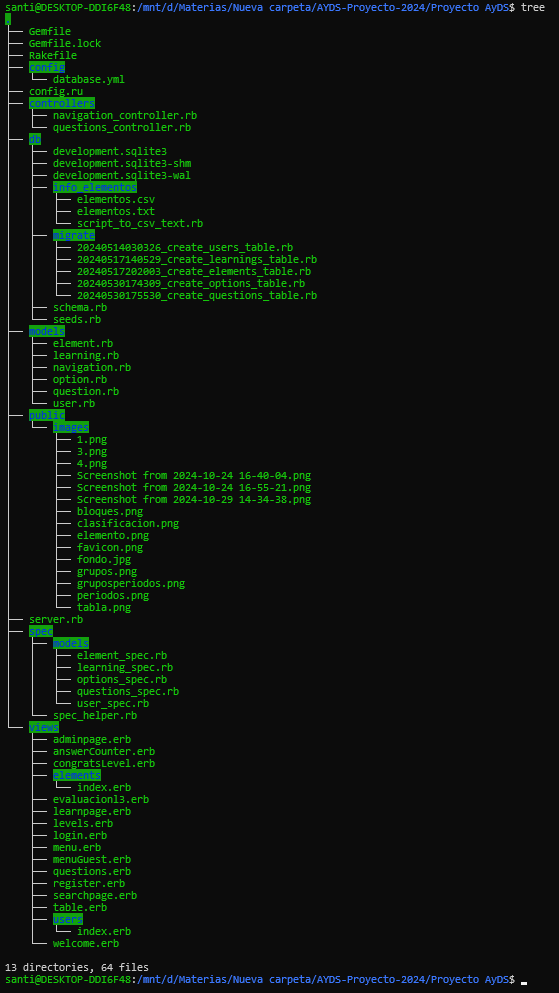
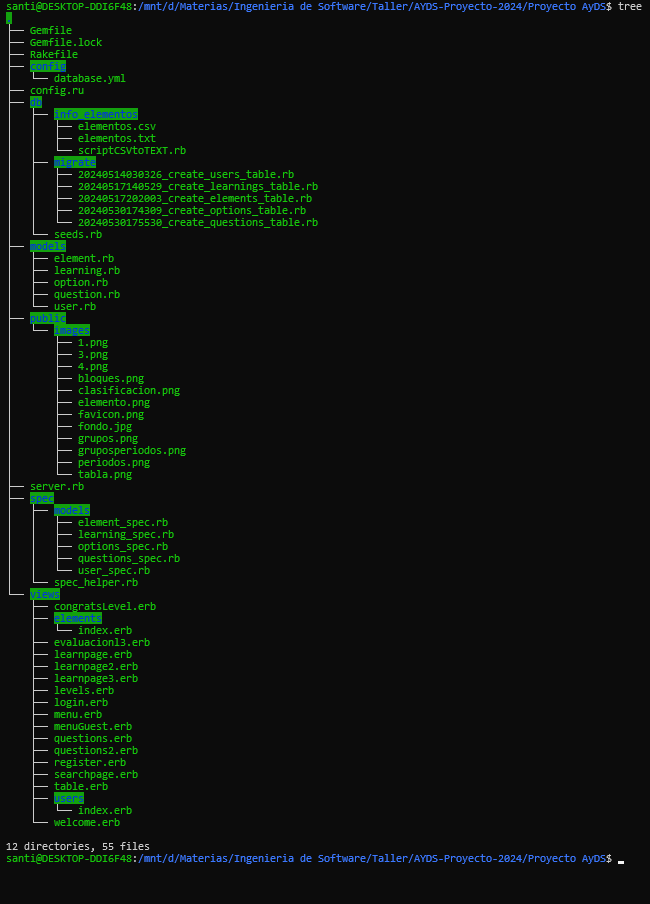
La actividad 2 proponía realizar tareas nuevas que agregaran funcionalidades al programa, pero debido a que la reestructuración del funcionamiento del recorrido de las lecciones era mucho más importante realizarla primero, tuvimos que dejar para más adelante algunas actividades que dependían fuertemente de esta reestructuración, como reparar el indicador de desempeño y unificar los modelos y vistas duplicados.

Actividad 3 – Gestión de tareas

Durante esta actividad agregamos a GitHub ramas para poder aislar las distintas tareas del desarrollo principal de del programa y así no interferir con la estabilidad del mismo

A parte, seguimos trabajando en la reestructuración y también realizamos algunas tareas que no dependían de esta, como crear un usuario administrador que posee todos los niveles de aprendizaje desbloqueados para facilitar el testeo de estos, además de incorporar una funcionalidad de preguntas autogeneradas.

Actividad 4 – Refactorización de la clase app.rb

 Antes Despues

Durante la actividad 4 continuamos con la reestructuración del funcionamiento del recorrido de las lecciones y también refactorizamos la clase server.rb ya que esta era muy extensa, además, tenía múltiples responsabilidades y esto dificultaría la escalabilidad del programa.

Extrajimos las clases navigation\_controller.rb y questions\_controller.rb para enfocar su funcionamiento a un tipo de acción en específico y así separar las responsabilidades de la clase server.rb, además de reducir el tamaño de esta.

