Análisis de deuda técnica

Se llevaron a cabo tres tipos de pruebas para analizar el proyecto.

En primer lugar, se realizó un análisis estático del código. Para el análisis del backend, se utilizó **StyleCop Analyzers**, que se agregó como un paquete Nuget y se encargó de mostrar advertencias sobre mala codificación o código que no sigue buenas prácticas. Para el análisis estático del frontend, se empleó **ESLint**, el cual también detecta errores relacionados con malas prácticas. En ambos casos, los errores y advertencias encontrados se reportaron como **issues** en GitHub, utilizando la etiqueta "**code review**" y asignándoles una baja prioridad y severidad leve.

El único problema de code review que se reportó con una prioridad media fue el problema #22, que señala que se está utilizando net 5.0, el cual ya no está soportado y no recibirá actualizaciones de seguridad, por lo que existe una leve urgencia en repararse.

En segundo lugar, procedimos a ejecutar los **tests unitarios** que se habían elaborado para el código del backend. Se identificaron únicamente 3 tests con errores, los cuales correspondían a la clase ReflectionHelpers. Estos problemas se reportaron en GitHub como issues, utilizando la etiqueta "**bug**", asignándoles una severidad mayor y una prioridad inmediata, con el fin de que se les brindara la atención necesaria de manera oportuna.

Por último, llevamos a cabo **pruebas exploratorias** donde probamos la funcionalidad de la aplicación en su conjunto, sin seguir un plan de pruebas estructurado previamente. Durante estas pruebas, se realizaron diversas interacciones con la aplicación con el objetivo de identificar cualquier problema o error no previsto.

Los resultados de estas pruebas exploratorias se registraron y se reportaron en forma de issues en Github, detallando el problema encontrado, la gravedad del mismo y su posible solución. Esto permitió que el equipo de desarrollo pudiera tomar las medidas necesarias para corregir los problemas y garantizar la calidad y la funcionalidad de la aplicación.

Se asignó una severidad adecuada a todos los issues reportados en GitHub, con el fin de reflejar la gravedad del problema detectado. Además, el Product Owner trabajó en conjunto con el equipo para determinar la prioridad de cada issue, en función de su importancia para la funcionalidad de la aplicación y su impacto en el usuario final.