Universidad ORT Uruguay

Facultad de Ingeniería

Informe de avance 1

Ingeniería de Software Ágil 2

Repositorio en Github

Los distintos documentos que forman parte de la entrega se encuentran dentro del repositorio Documentación/Entrega 1

Melissa Molinari – 229896

María Agustina Lamanna – 223040

Sebastian Daners - 255927

Profesores: Álvaro Luis Ortas García, Carina Paola Fontán San Martín

Grupo: M7A

Septiembre 2023

Definición del proceso de ingeniería en el contexto de KANBAN

Definición de roles

En este caso, comenzamos definiendo los roles que llevaríamos a cabo durante esta primera entrega. Según lo planteado por el marco de trabajo ágil KANBAN no se tienen roles definidos. Pero decidimos adaptarlo a las necesidades requeridas para este proyecto y en específico para esta entrega. Por lo tanto, los tres cumplimos roles de desarrolladores y testers. A su vez definimos el rol de SM (Scrum Master), el mismo lo adoptó Sebastián Daners. Consideramos que este rol sería beneficioso sobre todo para la instancia de retrospección ya que cumplía un rol de moderador durante la reunión. Por otra parte, brindó algunas recomendaciones de formas de mejorar nuestro trabajo para futuras instancias de trabajo y además colaboró con sus opiniones fundamentadas al momento de la toma de decisiones respecto a los artefactos planteados por KANBAN.

Por otra parte, definimos el rol de PO (Product Owner), el mismo lo adoptó Melissa Molinari. En este caso, al tratarse de un análisis de deuda técnica sobre un proyecto ya finalizado, el PO no realizó tareas de relevamiento de requerimientos. Sino, que en conjunto con el resto del equipo se clasificaron las prioridades de los bugs encontrados en el proyecto. Consideramos que este rol tendrá un papel más significativo en las siguientes etapas de este proyecto.

Explicación del tablero y su vínculo con el proceso de ingeniería

En esta oportunidad, elegimos utilizar un tablero ágil con cuatro columnas. En la primera columna *To do* se fueron posicionando las tarjetas correspondientes a las tareas a realizar. En la columna Doing se encontraban aquellas tarjetas de las tareas que se estaban realizando. Finalmente, en la columna *Done* se dejaban las tarjetas correspondientes a las tareas finalizadas.

Como se puede denotar, mencionamos *tareas* y no *user stories*. Esto se debe a que en esta primera etapa, en donde realizamos un análisis de deuda técnica y un testing exploratorio no se llevaron a cabo tareas dirigidas a la codificación y diseño. Por lo tanto, decidimos no hablar de user stories. Por esta razón es que decidimos utilizar un tablero ágil, solo con tres columnas y no expandimos las tareas que conforman el *doing*. Consideramos que una simple columna bastaba para entender las tareas que se estaban llevando a cabo.

A su vez, consideramos que para las futuras etapas del proyecto en donde se llevarán a cabo tareas de desarrollo tendremos que utilizar un tablero más descriptivo y adaptado a dichas actividades. Por lo tanto, consideramos

que en esos casos utilizaremos un tablero sustentable con columnas descriptivas para cada actividad a realizar.

Guía del repositorio

Elementos que contiene

El repositorio cuenta con una carpeta *Código* la cuál incluye el código fuente de la aplicación. Por otra parte, se encuentra la carpeta *Documentación* donde se irá creando una subcarpeta correspondiente a cada entrega. Dentro de la misma se podrán encontrar los artefactos recomendados para dicha entrega.

Otros elementos incluídos dentro del repositorio son el tablero Kanban. Se tomó la decisión de crear un nuevo tablero para cada entrega. En esta primera entrega, su correspondiente tablero se denomina *Entrega 1* y en él se detallaron las tareas realizadas para esta entrega, junto a los responsables de la misma, el tiempo empleado para realizar dicha actividad, el esfuerzo (medido en horas-persona) y una pequeña descripción de lo realizado.

Por otra parte, se especificaron los issues encontrados durante el análisis de deuda técnica, con su correspondiente clasificación. Estos se pueden encontrar en la sección denominada *Issues*.

Finalmente, para cada entrega se realizará una nueva versión del repositorio. Estas se podrán identificar mediante la etiqueta *tag* con el nombre ***v.número_entrega***.

Estrategia de flujo

Utilizaremos principalmente la estrategia de branching GitFlow. Se crearán branches de tipo feature o bugfix/issue donde luego se hará merge para tener siempre la rama main en estado de posible deploy. También se planea minimizar la duración de estas brunches para disminuir la cantidad de posibles merge conflicts.

Análisis de la deuda técnica

Análisis de código

Se comenzó por realizar un análisis estático del código. Para lo cual se investigó entre distintas herramientas como Roslyn Analyzers , Resharper o StyleCop Analyzers. Se optó por *StyleCop* ya que puede ser fácilmente instalada en “Administrar paquetes de Nuggets para la solución” y luego basta con compilar para que las advertencias aparecen en el tab “lista de errores”. Los errores encontrados advierten sobre malas prácticas de clean code en el código del proyecto, las cuales fueron reportadas en Github como issues. Dado que son errores que no afectan la funcionalidad si no que la lectura del código se considera que su severidad y prioridad es baja.

Para complementar el análisis utilizamos también la herramienta *NDepend* la cual reveló otros problemas en el código los cuales están reportados en el issue #16 . NDepend resulta útil para clasificar la severidad de los issues encontrados ya que provee un estimado de las horas necesarias para solucionar los problemas y los clasifica según esa métrica. Por ejemplo los issues reportados en github son aquellos que fueron clasificados como “críticos” por Ndepend ya que se estima que llevaría más de 2 horas para solucionarlos.

Análogamente para realizar el análisis del código del frontend se utilizó la herramienta *EsLint*. Se detectaron dos issues que se repetían varias veces en diferentes archivos del código del frontend y que fueron registrados como los issues #10 y #13.

A continuación se realizó el testing exploratorio en búsqueda de defectos en las funcionalidades especificadas por la letra. La metodología que se decidió seguir fue crear una lista de las funciones que debe poder llevar a cabo la app y probar una por una para asegurar su funcionamiento. Una vez finalizado reportamos los bugs como issues en github.

Issues detectados

Luego de poner en conjunto entre todos los integrantes del grupo tomamos la decisión de seguir un cierto orden para el análisis del sistema. Comenzamos leyendo la letra entregada a los que desarrollaron el sistema, de esta forma determinamos cuáles eran los requerimientos solicitados. Una vez listados dichos requerimientos, comenzamos a probar el sistema para corroborar si los mismos estaban implementados o no. En caso de estar implementados también verificamos que el funcionamiento fuera agradable para el usuario.

Una vez finalizado dicho análisis, generamos 7 bugs los cuales fueron detallados dentro del proyecto de GitHub en la sección denominada *Issues*.

Sobre la instalación de la aplicación, encontramos que por defecto el puerto que expone el backend y el puerto que busca el frontend son distintos. Sin embargo, al ejecutar el backend mediante IIS (o mediante Visual Studio en modo IIS) este si expone el puerto que está buscando el frontend. Decidimos no ponerlo como issue ya que el objetivo del proyecto es tener el backend funcionando en IIS

Clasificación de issues

Para esta primera entrega consideramos 2 posibles categorías de issues, bugs o problemas de estándares/code smell.

Para la clasificación de bugs utilizamos como referencia principal el Anexo 1 de la letra del proyecto. Esto significa que dividimos a los bugs en dos rangos, según su prioridad y según su severidad.

Según la prioridad se dividen en los 4 siguientes puntos:

- ***Inmediata***: plazo máximo 24 hs. (generalmente los defectos de severidad crítica se asocian a prioridad inmediata).
- ***Alta***: plazo máximo 48 hs.
- ***Media***: plazo máximo un par de semanas.
- ***Baja***: sin plazo.

Y según la severidad se dividen en estos 4 puntos:

- ***Crítico***: un defecto que obstaculice o bloquee completamente la prueba o el uso de un producto o función.
- ***Mayor***: una función principal que no cumpla con los requisitos y se comporte de manera diferente a lo esperado. Cuando funciona muy lejos de las expectativas o no está haciendo lo que debería estar haciendo.
- ***Menor***: función que no cumpla con sus requisitos y se comporte de manera diferente a lo esperado, pero su impacto es insignificante hasta cierto punto o no tiene un impacto importante en la aplicación.
- ***Leve***: defecto cosmético.

Para los problemas de estándares utilizamos solamente el rango de prioridad, ya que nunca tendrían una severidad mayor a baja. Por ende cada issue está clasificado solamente por prioridad.

Resumen de bugs

Prioridad	Severidad	Cantidad
Inmediata	Crítico	2
Inmediata	Menor	1
Alta	Mayor	1
Media	Menor	1
Alta	Leve	1

Resumen de issues

Prioridad	Cantidad
Baja	12
Elevada	0
Media	4
Inmediata	0

Sobre la entrega

Alcance de la entrega

Está primera entrega se centró en un análisis profundo de la aplicación mediante testing exploratorio así como análisis estático del código. Se generó un tablero donde se registraron las tareas a llevar a cabo y además los issues y bugs encontrados en la aplicación.

Todos los avances hechos durante está primera etapa se pueden encontrar en el repositorio de git provisto.

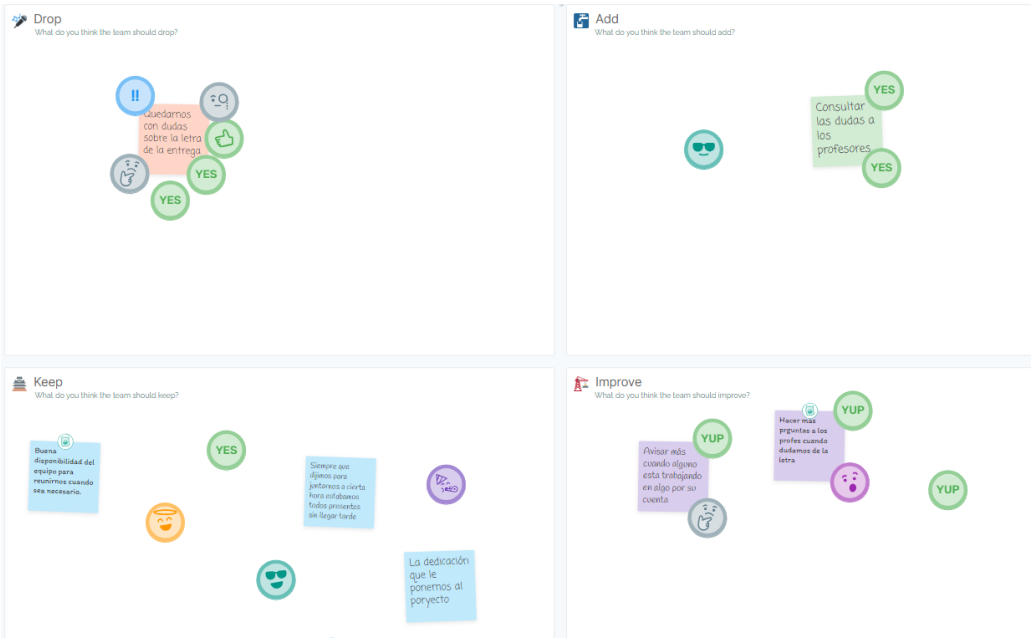
Conclusiones

En esta entrega se lograron definir bugs e issues para solucionar en futuras entregas. Se logró realizar un análisis de deuda técnica que nos ayudó a entender mejor el sistema y a comprender su código. Además, se plantearon métodos y estrategias a utilizar para futuras entregas con respecto al versionamiento y uso de repositorio.

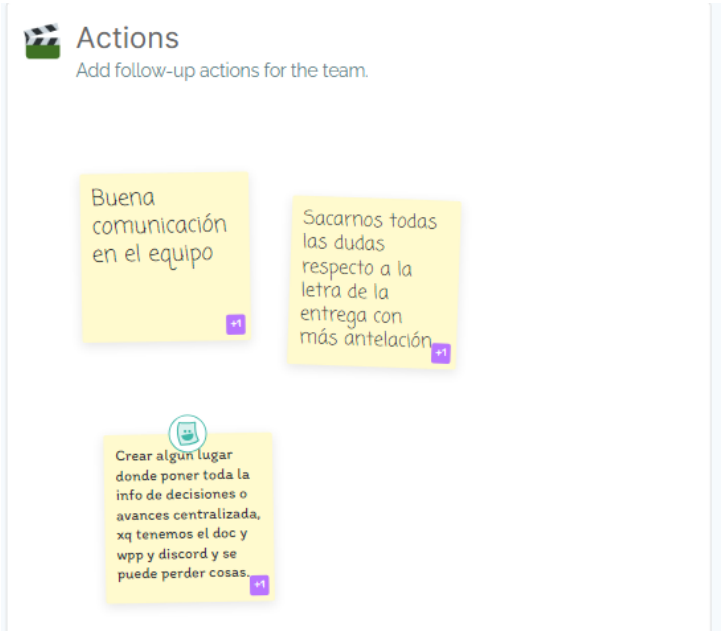
Retrospectiva

Con el objetivo de mejorar nuestro trabajo en cada iteración se realizó una retrospectiva con el formato DAKI (Drop, Add , Keep , Improve) en la herramienta Metro Retro.

Se genero el siguiente tablero:



A partir del cual se eligieron las siguientes acciones como mejoras a implementar en futuras entregas:



Link al video de la retrospectiva:
https://youtu.be/atiIBP0hBr8