Universidad ORT Uruguay

Facultad de Ingeniería

Informe académico

Ingeniería de Software Ágil 2

La guía de despliegue e instalación se puede encontrar dentro del repositorio, en la carpeta Documentación/Entrega 5 o en el siguiente link

Melissa Molinari – 229896

María Agustina Lamanna – 223040

Sebastian Daners - 255927

Profesores: Álvaro Ortas, Carina Fontán

Grupo: M7A

Noviembre 2023

Índice

Gestión del proyecto

Actividades realizadas y productos entregados

El proyecto se dividió en 4 entregas. Para la primera entrega, como actividad principal se realizó un análisis de deuda técnica, siguiendo la guía planteada en el anexo de la letra para la clasificación de los issues encontrados. Además, se realizó una primera definición del marco general de KANBAN y la primera versión del proceso de ingeniería. Al finalizar el plazo de dos semanas se entregó un listado de todos los issues/bugs encontrados junto con la clasificación de los mismos. Los mismos llegaron a 23 issues reportados, que se pueden observar en el siguiente link.

En la segunda entrega, se seleccionaron tres bugs (los de mayor severidad y prioridad) de los hallados en la etapa anterior, para implementar su solución utilizando TDD. Para ello, se tuvo que redefinir el proceso de ingeniería planteado anteriormente, para su correcta implementación. Además se creó un tablero en GitHub, con las columnas correspondientes a las actividades a realizar para solucionar dichos bugs y se configuró el pipeline. Una vez finalizada la implementación de las soluciones, se realizaron reuniones de retrospectiva para el equipo y de revisión con el Product Owner. Finalmente, una vez transcurrida la semana pautadas se entregaron el código funcionando, con los bugs solucionados y un informe académico. En esta entrega, destacamos la implementación del pipeline de deployment como una herramienta valiosa para efectivizar el proceso de desarrollo.

En la tercera etapa, se realizó una actualización del proceso de ingeniería para que este contemplara la metodología de desarrollo que íbamos a utilizar, BDD. También se creó una nueva versión del tablero. La actividad principal de esta etapa, fue el desarrollo de nuevas funcionalidades. Para ello se aplicaron nuevas metodologías (BDD y One Piece Flow) y herramientas (Specflow). Durante esta etapa, pudimos comprobar las ventajas del uso de One Piece Flow. Esto nos llevó a evitar la aparición de cuellos de botella, esperable al utilizar herramientas nuevas para el equipo. Al finalizar las dos semanas pautadas, se entregaron las nuevas funcionalidades correctamente implementadas.

Para la cuarta y última entrega, se automatizaron los tests de integración para las funcionalidades nuevas implementadas en la entrega anterior. Para ello se utilizó la herramienta Selenium. Por otra parte, se calcularon las métricas DevOps con los datos extraídos en las entregas 2 y 3. Las mismas arrojaron resultados muy favorables con respecto a la entrega 3. Observamos que se logró un valor de Flow Efficiency elevado, concluímos que esto se debió a la utilización de la metodología One Piece Flow.

Antes de la fecha establecida para cada una de las entregas, se realizaron las correspondientes reuniones de revisión y retrospectiva con el Product Owner y Scrum Master establecidos para cada una de las etapas. Además se entregaron los correspondientes informes de avance de etapa.

Marco de gestión utilizado

A lo largo de este proyecto, utilizamos el marco de gestión ágil KANBAN. El mismo fue adaptado a las necesidades que implicó este proyecto. Por ello, decidimos conveniente el uso de roles, como plantea el marco de gestión SCRUM, como Product Owner y Scrum Master. Estos roles fueron esenciales al momento de realizar las reuniones de revisión y retrospectiva por el papel que tienen ambos roles en los mismos. El Product Owner fue el encargado de aceptar o rechazar los cambios o funcionalidades implementadas, a nivel de código. Aunque, particularmente para este proyecto, al encontrarse enmarcado en un proceso de aprendizaje, priorizamos la toma de decisiones en equipo. Por otra parte, el Scrum Master se encargaba de guíar las reuniones de retrospectiva, de esta forma lográbamos utilizar el método DAKI de forma más organizada.

A nivel de tablero, nos basamos en las ideas que plantea el marco de gestión KANBAN. Para cada tablero, se creó un tablero con las columnas correspondientes a cada una de las tareas que se iban a realizar en esa etapa del proyecto. Además, para cada una de las tareas o actividades que se iban a realizar se agregó una tarjeta. La misma se iba moviendo por cada una de las columnas, logrando informar de forma clara al equipo qué actividad se estaba realizando con respecto a dicha tarea. En la última entrega se planteó un tablero que contenía todas las columnas que se fueron utilizando a lo largo del proyecto. Consideramos que este último sería un buen tablero para utilizar en un proyecto ya que logra mostrar de manera visual todas las actividades a realizarse a lo largo de un proyecto, además de las actividades comprendidas por la metodología de desarrollo BDD, la cuál aporta grandes beneficios al momento de escribir código.

Métricas del proyecto

A partir de los datos recolectados durante las entregas 2 y 3, se calcularon las siguientes métricas DevOps: Lead Time, Cycle Time, Flow Efficiency. Dichas métricas fueron calculadas durante la entrega 4, disponible en el siguiente link.

Lead Time

El *Lead Time* es una métrica encargada de medir la duración total desde el inicio (se agrega a *To Do*) hasta la finalización de una tarea (cuando pasa a *Done*). La misma se mide en alguna unidad de tiempo (horas, días, meses, etc.). Por ello, esta métrica sirve para evaluar la eficiencia de los procesos y la velocidad de entrega de un equipo de trabajo.

En la siguiente tabla, se pueden observar los valores promediados de *Lead Time* obtenidos para cada una de las entregas. Estos valores fueron obtenidos al promediar los valores de *Lead Time* obtenidos para cada uno de los bugs solucionados o funcionalidades implementadas.

	Lead Time
Entrega 2	2 días
Entrega 3	2,25 días

Por lo que se puede observar de los valores obtenidos, el equipo mantuvo una velocidad de entrega similar en ambas entregas. Sin embargo, el leve aumento que se puede observar de la entrega 3 podría deberse a la complejidad adicional que significó para el equipo el uso de la herramienta Specflow. Se puede considerar que ambas entregas tienen un *Lead Time* relativamente corto, en comparación al tiempo establecido para la entrega, lo que indica una mayor eficiencia en el proceso de entrega.

Cycle Time

Esta métrica se utiliza para medir el tiempo real que transcurre desde que se inicia una tarea, hasta que se completa. A diferencia del *Lead Time*, el *Cycle Time* se enfoca específicamente en el tiempo activo de una tarea. Es decir, desde el momento en que pasa a la columna *In Progress* (o primera columna luego del *To Do*, dependiendo de cómo se configuró el tablero) hasta que se finaliza, llega a la columna *Done*. Se mide en alguna unidad de tiempo, (horas, días, meses).

Al igual que con el *Lead Time*, la siguiente tabla muestra los valores de *Cycle Time* promediados para cada entrega. Estos valores fueron obtenidos al promediar los valores de *Cycle Time* obtenidos para cada uno de los bugs solucionados o funcionalidades implementadas.

	Cycle Time
Entrega 2	1 día
Entrega 3	2,25 días

Al igual que con el *Touch Time*, la diferencia (en este caso más notoria) entre ambas entregas nos indica que la entrega 2 se completó más rápidamente en términos de tiempo activo de trabajo en comparación a la entrega 3.

Flow Efficiency

Esta métrica mide el porcentaje de tiempo que se dedica realmente al trabajo.

En la siguiente tabla, se muestran los valores de *Flow Efficiency* promediados para cada entrega. Estos valores fueron obtenidos al promediar los valores de *Flow Efficiency* obtenidos para cada uno de los bugs solucionados o funcionalidades implementadas.

	Flow Efficiency
Entrega 2	50%
Entrega 3	100%

Se puede concluir que el valor obtenido para la entrega 2, indica que solo el 50% del tiempo total del proceso se dedicó al trabajo valioso y productivo. Esto se debe, a que en dicha entrega se arreglaron bugs encontrados durante el análisis de deuda técnica. Debido a ello, no se tenían las users stories correspondientes ni sus criterios de aceptación. Por ello, se tomó la decisión en el equipo de considerar la tarea como finalizada cuando el Product Owner aceptara la solución implementada generando este valor de *Flow Efficiency*.

En el caso de la entrega 3, el valor elevado se debe a que se utilizó la metodología de trabajo conocida como *one piece flow* debido a que se estaba trabajando una metodología nueva (BDD), con herramientas desconocidas para el equipo (Specflow). Por ello, todas las user stories debían ser completamente implementadas antes de comenzar con la user story anterior. Esto hizo que no se generarán esperas al momento de avanzar con las tareas.

En ambos casos, se puede concluir que los valores obtenidos no son indicadores de que se debieran realizar cambios urgentes en la forma de trabajo del equipo.

Throughput

Esta métrica mide la cantidad de tiempo completado en una unidad de tiempo específica.

	Throughput
Entrega 2	3 bugs en 2 días.
Entrega 3	4 funcionalidades en 7 días

Esfuerzo total

El *Esfuerzo total* se refiere a la cantidad total de trabajo, en este caso media en horas-persona.

	Esfuerzo total
Entrega 2	19,58 horas-persona
Entrega 3	37,7 horas-persona

Reflexiones sobre el aprendizaje

Aplicación del marco de gestión ágil - KANBAN

El uso de la metodología KANBAN durante el proyecto se puede considerar mayormente exitoso. En este sentido, lo que hicimos bien y deberíamos continuar haciendo fue nuestra habilidad para adaptar el tablero a nuestras necesidades específicas, las cuales fueron cambiando en cada iteración. A su vez, en cada entrega se asignaron los roles de Product Owner y Scrum Master rotativamente lo cual nos permitió realizar reviews y retrospectivas consistentemente durante todo el proyecto.

En cuanto a lo que podríamos añadir en el futuro, se podría considerar la implementación de herramientas de seguimiento automático para el tablero KANBAN. Esto no solo facilita la actualización en tiempo real, sino que también nos permitiría analizar datos históricos para tomar decisiones más informadas en futuros proyectos. Además, podríamos explorar la posibilidad de establecer reuniones periódicas específicas para revisar el progreso del tablero y abordar cualquier problema de manera proactiva.

Finalmente, lo que consideramos que se debería mejorar es la actualización del Tablero. En ciertas ocasiones, el progreso de las tareas no se veía reflejado en el Tablero, principalmente porque nuestro equipo es pequeño y a menudo nos comunicamos sobre los avances a través de otros canales. Para abordar este desafío, podríamos implementar un enfoque más riguroso en cuanto a la actualización del Tablero, estableciendo políticas claras sobre cómo y cuándo se deben registrar los avances, de está forma se aumenta la visibilidad y transparencia permitiendo la autogestión para cada integrante.

Uso de nuevas herramientas y estrategias

Análisis de deuda técnica

En la entrega 1, realizamos un análisis de deuda técnica para identificar defectos en la aplicación. Valoramos positivamente la práctica de utilizar varias herramientas para el análisis estático del código, ya que cada una destaca diferentes aspectos a mejorar. Dado que estas herramientas eran desconocidas para nosotros, el equipo colaboró de manera conjunta para comprenderlas, evitando posibles cuellos de botella al esperar resultados. En el análisis exploratorio, todos contribuimos con distintas ideas para revelar problemas en las funcionalidades, lo que enriqueció aún más nuestra evaluación. Mantener esta colaboración y diversidad de enfoques en futuros proyectos seguirá siendo fundamental para un análisis de deuda exhaustivo y una mejora continua en nuestra calidad de trabajo.

Como mejora, sería útil explorar maneras de automatizar ciertos aspectos del análisis podría ayudarnos a ser más eficientes y a centrarnos en solucionar los problemas identificados en lugar de gastar tiempo en procesos manuales.

Implementación de repositorio y procedimientos de versionado

Sobre el repositorio y los procedimientos, durante el proyecto mantuvimos satisfactoriamente la metodología de *Trunk-based*, como resultado la colaboración entre los miembros del equipo fue más fácil y estructurada.

Como mejora, reconocemos que deberíamos haber sido más rigurosos en la revisión y aprobación antes de fusionar cambios en la rama principal. A pesar de nuestras intenciones, hubo ocasiones en las que se necesitaron hotfixes en la rama main para abordar problemas críticos.

Otro aspecto a refinar es el formato de los mensajes de nuestros commits ya que en general se siguió el estándar predefinido pero no fuimos tan rigurosos en está parte como podríamos haberlo sido.

GitHub Actions - pipeline

La creación del pipeline con GitHub Actions presentó desafíos al intentar incorporar pruebas de Specflow y Selenium. A pesar de añadir las pruebas de TDD correctamente, nos enfrentamos a obstáculos debido a la falta de conocimiento técnico en estas herramientas. Como mejora, reconocemos la importancia de profundizar en la investigación de los errores encontrados al ejecutar las pruebas, permitiéndonos una implementación más efectiva en futuros proyectos.

A pesar de las dificultades, investigamos y obtuvimos el código necesario para integrar Specflow y Selenium en GitHub Actions, en caso de resolver los problemas. Este enfoque proactivo es un atributo positivo que debemos mantener. Además, el compromiso del equipo para abordar de inmediato cualquier error detectado por la ejecución del pipeline es una práctica que ha contribuido a mantener la integridad del repositorio y debe mantenerse en futuros proyectos.

Generar escenarios de testing desde la perspectiva del usuario

Con el objetivo de generar tests desde la perspectiva del usuario se aprendieron a usar nuevas herramientas. Lo que consideramos que hicimos bien y deberíamos seguir haciendo fue crear tests completos que evalúen no solo los casos de éxito sino que también realizamos exhaustivos casos de error con la meta de aumentar la calidad del código.

Por otra parte reconocemos que inicialmente el aprendizaje relacionado al uso de la herramienta Specflow fue un gran contratiempo. El uso de la metodología de One Piece Flow resultó útil ya que implicó que todos los miembros del equipo se dedicaron completamente a aprender la herramienta lo cual aceleró el avance. Sin embargo, consideramos que se podría mejorar el tiempo que lleva integrar nuevas tecnologías al proyecto investigando sobre otras metodologías o designando tiempo extra para el aprendizaje.

Por último, algo que no hicimos y deberíamos, es reconsiderar la estrategia de pruebas para no depender de la base de datos real, optando por el uso de mocks. Esta adaptación permitiría una ejecución más eficiente en un pipeline y mejoraría la eficacia de nuestras pruebas.

Automatización del testing funcional

En general, como equipo demostramos un manejo eficaz de Selenium para automatizar el testing funcional o de caja negra. La rápida familiarización con esta herramienta resultó beneficiosa. Continuaremos utilizando Selenium de manera efectiva y aprovecharemos sus capacidades para pruebas más avanzadas y mantenimiento eficiente.
Identificamos como posible mejora evitar que las pruebas modifiquen directamente la base de datos. Deberíamos dejar de depender de pruebas que alteran la base de datos real y, en su lugar, considerar la implementación de mocks para asegurar que las pruebas no tengan impacto en la integridad de los datos de producción.

Reflexiones sobre DevOps.

Como equipo evaluamos que nuestra ejecución de la metodología DevOps ha sido exitosa dado nuestro conocimiento adquirido durante el proyecto. Destacamos la automatización de tests y la retroalimentación asociada, la cual ha mejorado la calidad del software y acelerado nuestros procesos de desarrollo. La elección de trunk-based para la integración continua ha demostrado ser beneficiosa al reducir conflictos durante la fusión de ramas, lo que ha contribuido a una mayor eficiencia y facilitado la entrega continua. Además, el script de deploy creado en está iteración pretende agilizar nuestras implementaciones, siguiendo la idea de DevOps de Infraestructura como código.
Como mejoras a lo trabajado durante el proyecto contemplamos la automatización de tests, particularmente en la integración con GitHub Actions. Explorar soluciones para solucionar las dificultades encontradas durante la integración de pruebas con herramientas como BDD y Selenium en GitHub Actions. Además, se podría mejorar la organización de los canales de comunicación utilizados por fuera del tablero. Establecer un protocolo claro para la comunicación podría aumentar la eficacia y evitar posibles malentendidos.
Finalmente, exploramos la posibilidad de ampliar los tipos de retroalimentación obtenida a través de la automatización de pruebas. En el inicio del proyecto, se llevaron a cabo análisis estáticos de código, y se podría evaluar la posibilidad de incorporarlos a GitHub Actions para mejorar la detección temprana de errores. Además, se propone considerar prácticas como la revisión de código en pares para asegurar la coherencia y calidad del código que se fusiona en el repositorio. Estas iniciativas no solo fortalecerán la robustez del proceso de desarrollo, sino que también contribuirían a mantener altos estándares de calidad en el código del proyecto.

Lecciones aprendidas

A continuación se detallan varias lecciones aprendidas durante este proyecto. Consideramos que las mismas podrían ser de utilidad para futuros proyectos similares.

Marco de trabajo ágil - Kanban

Si quieren aplicar el marco de trabajo KANBAN de forma efectiva, entonces es necesario acordar las políticas de trabajo que se van a llevar a cabo.

Debido a la alta flexibilidad que permite este marco de trabajo ágil, es necesario que el equipo se reúna en conjunto y acuerde las tareas a realizar para las fechas establecidas de entrega. Como el marco de trabajo no establece fechas límites, sino actividades a llevar a cabo, pero como este obligatorio sí implicaba fechas de entrega, vimos

beneficioso realizar reuniones en dónde los desarrolladores acordaron de antemano las actividades que realizan y los plazos que debían manejar teniendo en cuenta dichas entregas pautadas.

Desarrollo Guiado por Pruebas (TDD)

Si quieren mejorar la calidad del código, entonces utilicen TDD.

El uso de TDD permitió que el desarrollo de cada funcionalidad sea más fluido, dado que se podía observar en tiempo real si los cambios efectuados cumplían con los requisitos y expectativas del sistema asegurando la calidad del producto. Además el uso de esta metodología de desarrollo permite visualizar los errores de forma temprana, logrando resolverlos en etapas tempranas de desarrollo.

Desarrollo Basado en Comportamiento (BDD)

Si quieren aplicar BDD a su proyecto, entonces tengan en cuenta su curva de aprendizaje.

Esta metodología de trabajo incluye actividades y artefactos específicos, como lo son la definición de requerimientos y el artefacto resultante de la misma, la user story. Esta última tiene un formato específico para su descripción y sus criterios de aceptación. Notamos que adaptarnos a escribir en dicho formato y lograr definir cuáles eran los criterios de aceptación para cada requerimiento nos llevó más tiempo del esperado. Sobre todo, adaptarse a las herramientas que esto implica (como en nuestro caso fue Specflow).

Esto hizo que el equipo invirtiera más tiempo del esperado en lograr adaptarse a las nuevas herramientas y metodologías.

One Piece Flow

Si quieren reducir el tiempo de ciclo, entonces utilicen One Piece Flow

El uso de One Piece Flow, nos permitió movernos de una etapa a otra de manera continua y sin esperas, lo cual eliminó los cuellos de botella y redujo significativamente el tiempo total necesario para completar un producto o servicio.

Si quieren aplicar One Piece Flow, entonces tengan en cuenta que pueden surgir obstáculos inesperados, como limitaciones tecnológicas o falta de conocimiento en ciertas áreas.

En ocasiones, el equipo quedó atascado durante un período prolongado debido a un problema específico, lo que impidió avanzar en otras tareas. En estos casos, es importante ser flexible y encontrar un equilibrio entre mantener el flujo constante y abordar los desafíos técnicos.

Otros Consejos

Si quieren realizar mejoras en el proceso entonces implementen retrospectivas frecuentes.

Las retrospectivas promovieron un ambiente de reflexión donde cualquier miembro del equipo pudo compartir sus inquietudes o sugerencias libremente. Esta transparencia y colaboración han sido fundamentales para identificar áreas de mejora y aplicar soluciones de manera ágil.

Conclusiones

Para concluir con lo antes expuesto, a lo largo de este proyecto el equipo aprendió valiosas lecciones sobre la aplicación de las metodologías ágiles, como KANBAN, las métricas que se pueden generar, así como metodologías de desarrollo como lo es BDD y la automatización que está ligado a ambos conceptos.

La utilización de la metodología ágil KANBAN nos permitió experimentar de primera mano la visualización más clara de las actividades que están realizando los integrantes del equipo y el uso de los tableros que describen esta metodología. Logrando adaptar las actividades que él mismo describe, a las necesitadas por nuestro proyecto.

Por otra parte, las mediciones de las métricas de DevOps nos permitió obtener valiosa información sobre la eficiencia que tuvo el equipo en tareas de desarrollo. Esto nos permitió visualizar de primera mano las grandes ventajas que aporta la utilización de la metodología de desarrollo one-piece flow, sobre todo en casos en dónde se utilizan herramientas nuevas. Debido a ello, se lograron valores de métricas satisfactorios. Gracias a ello podremos aplicar estos nuevos conocimientos para futuros proyectos, así como la utilización de las métricas para la toma de decisiones informadas.

La implementación de BDD nos permitió definir criterios de aceptación desde un comienzo, asegurando de esta forma la correcta implementación de la funcionalidad a la que referían ya que nos brindó una comprensión más clara de los requisitos. Por otra parte, la utilización de la herramienta Specflow para la implementación del código referente a dichos criterios de aceptación, nos permitió ver de forma aún más clara los grandes aportes que nos brindan los criterios de aceptación. Además, de la facilidad de la implementación del código una vez que los criterios de aceptación fueron definidos.

Junto con ello, la implementación de pruebas automatizadas nos permitió detectar errores de forma rápida y temprana. Además, los conocimientos adquiridos sobre GitHub Actions y la configuración del pipeline, nos permitió asegurarnos que todo lo que fuera introducido a la rama *main* del repositorio lograba cumplir satisfactoriamente las pruebas codificadas. Logrando de esta forma agilizar y asegurar la calidad del proceso de desarrollo.

En conclusión, este proyecto nos permitió poner en práctica las metodologías aprendidas en clase, fortaleciendo nuestro entendimiento de ellas. Por otra parte, la utilización de nuevas herramientas nos ayudó a lograr una mayor eficiencia al momento de desarrollar código. Finalmente, logramos reafirmar el valor que poseen las técnicas de gestión de proyectos.