

Universidad ORT Uruguay
Facultad de Ingeniería

Obligatorio Ingeniería de Software Ágil 2

Entregado como requisito para la obtención del título de
Ingeniero en Sistemas

Milena dos Santos – 254813

Guzmán Dupont – 230263

Julieta Sarantes – 251105

Tutores: Alvaro Ortas, Carina Fontán

2023

ÍNDICE

Presentación.....	3
Entrega 1.....	3
Entrega 2.....	4
Entrega 3.....	5
Entrega 4.....	5
Reflexiones.....	7
Lecciones aprendidas.....	8
Conclusiones.....	10
Guía de Instalación:.....	11

Presentación

A continuación presentamos las reflexiones, las lecciones y las decisiones tomadas, que fueron surgiendo a lo largo de todo el Proyecto DevOps, comenzando por la entrega número uno, hasta la entrega número cuatro.

Entrega 1

El objetivo de esta entrega fue introducirnos al marco general de Kanban, familiarizarnos con el proyecto entregado por los docentes y determinar los bugs presentes en la aplicación.

Para completar estos objetivos creamos los siguientes artefactos:

- Definición/uso del proceso de ingeniería en el contexto de KANBAN,
- Explicación del tablero y su vínculo con el proceso de ingeniería,
- Creación y posterior mantenimiento del repositorio y Documento principal.
- Tablero Kanban

Todos estos artefactos fueron utilizados para llevar un control del estado del proyecto y documentar las decisiones y experiencias durante el mismo.

Los bugs encontrados fueron once, y las tareas que fuimos registrando fueron seis. Debemos aclarar que las tareas, si bien se encuentran en la columna In Progress, sí fueron completadas pero no fueron llevadas a la columna Done del tablero, consideramos que este es un error por nuestra parte.

Dejamos el link al documento principal de la entrega 1 donde se puede llegar a los demás artefactos de la entrega: [link](#)

Dentro de las reflexiones rápidas que podemos mencionar están:

- Comenzar el trabajo con tiempo es esencial.
- Hacer una especie de sprint planning para definir las cosas por hacer.
- Mantener una buena comunicación es fundamental.
- El trabajo es más difícil y exigente de lo que uno puede pensar en un principio.

Entrega 2

Nos enfrentamos a la resolución de tres bugs que surgieron durante la exhaustiva búsqueda de estos en el proyecto entregado por los profesores. Utilizamos la metodología de TDD (Desarrollo Guiado por Pruebas). Este enfoque implica escribir las pruebas antes de corregir errores existentes, lo que tuvimos que hacer en esta etapa.

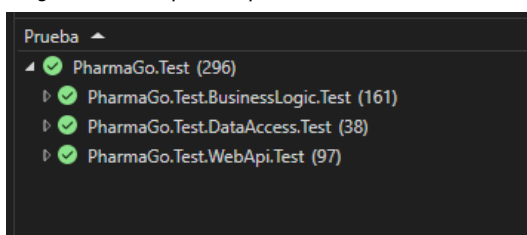
El proceso TDD nos permitió identificar y solucionar los bugs de una manera sencilla y más sistemática, asegurándonos que nuestras correcciones no introdujera nuevos problemas en el código existente. A su vez, esto aumentó la calidad de nuestro código, creando una base sólida para el desarrollo, logrando un código fácil de mantener a lo largo del ciclo de vida del proyecto.

En esta entrega también configuramos el pipeline de integración continua utilizando Github Actions. integramos un job que tenía la tarea de automatizar la integración del backend cada vez que se hacía un push o un pull request a nuestra rama develop o main.

Un aspecto a destacar de esta entrega es que cumplimos de manera satisfactoria con los objetivos establecidos. Pudimos resolver de forma exitosa los tres bugs mediante la implementación de la metodología mencionada con anterioridad, y además implementar el pipeline de forma correcta, lo que nos llevó a generar una entrega completa y lograr un buen proceso de entrega en esta iteración.

Se entregó el proyecto en principio con 291 test, le agregamos 5 para la entrega 2.

Imagen donde se puede apreciar todos los test en verde:



Dejamos el link al documento principal de la entrega 2 donde se puede llegar a los demás artefactos de la entrega: [link](#)

Entrega 3

Esta entrega tenía como objetivo el desarrollo de algunas funcionalidades nuevas que debían ser implementadas al proyecto utilizando la técnica de BDD (Behavior Driven Development). Estas nuevas funcionalidades consistían en la implementación de alta, baja y modificación de productos de farmacia, como por ejemplo shampoo, perfume, etc, y además la compra de dichos productos.

La técnica de BDD consiste en la programación basada en comportamiento es decir debemos definir la narrativa y los escenarios, también llamados requisitos de aceptación, de las funcionalidades a implementar. La narrativa se expresa con las siguientes palabras clave:

- Como
- Quiero
- Para

Dentro del “Como” se define el actor del requisito, en el “Quiero” se define lo que se desea hacer y el “Para” es la justificación del requisito.

Los escenarios se definen con:

- Dado
- Cuando
- Entonces

Donde el “Dado” establece el contexto o las condiciones iniciales del escenario, el “Cuando” describe la acción o el evento que se va a probar y el “Entonces” se especifica el resultado o el comportamiento esperado después de que se haya ejecutado la acción descrita en la sección “Cuando”.

Para desarrollar esto a nivel de código utilizamos la herramienta Specflow.

Debemos decir que no nos sentimos conformes ni orgullosos con los resultados obtenidos en esta entrega ya que no fuimos capaces de terminar las *features* requeridas en el tiempo establecido. La principal razón de esto es que no dedicamos el tiempo necesario para la definición de escenarios. Más razones de esto se explican detalladamente en la sección Reflexiones de este mismo documento.

Dejamos el link al documento principal de la entrega 3 donde se puede llegar a los demás artefactos de la entrega: [link](#)

Entrega 4

El objetivo de esta entrega fue desarrollar los test de integración con la herramienta Selenium. Este procedimiento se hace definiendo los test que prueban la aplicación a nivel de backend y frontend de manera conjunta. Para la creación de estos tests se definen ciertos pasos que Selenium sigue, simulando el comportamiento de un

usuario que ejecuta las funcionalidades, y si se devuelve el resultado esperado se marca el test como correcto.

Dado que no logramos implementar las funcionalidades de la entrega pasada, mucho del tiempo invertido en esta entrega fue para terminar aquello que no habíamos logrado terminar. De las funcionalidades pasadas logramos implementar el alta y baja de producto. Decidimos enfocarnos en completar algunas de las funcionalidades anteriores para no perder tanto tiempo ya que el objetivo de la entrega era implementar las pruebas de integración con Selenium. Si bien logramos implementar un test de Selenium no nos quedamos nada conformes con el resultado.

Para esta entrega también se pidieron algunas métricas que son una herramienta fundamental para medir y evaluar el rendimiento del proyecto por lo que es importante conocerlas y controlarlas. Las métricas analizadas fueron las siguientes: Lead time, Cycle time, Flow Efficiency y Throughput.

Para la entrega llegamos a calcular las métricas de las entregas 2 y 4, y obtuvimos ciertos resultados pero no llegamos a hacer el análisis correspondiente, que es una parte fundamental en el estudio de las mismas.

Los resultados del estudio de la entrega 2 son los siguientes: el bug de crear invitación entró al tablero el día 20/09 y entregada al cliente el 27/09, 7 días para completar el bug. El Cycle time fue de 3 días. Elegimos la unidad de tiempo días ya que es la que más se ajusta a los tiempos de la entrega. Luego el Flow Efficiency fue de $3/7 = 0,42$ (42%) que es un valor deseable ya que indica que el tiempo dedicado a la tarea se gastó en trabajo productivo. El bug de invitación con nombre de usuario en el sistema llevó de Lead time 3 días y de cycle time 2 días. El Flow efficiency fue de 0,7 (70%) que es un buen valor dado que se encuentra por encima de la heurística del 40%. El bug de comprar medicamento tuvo un Lead time de 3 días y un Cycle time de 2 días. El Flow efficiency fue de 0,7 (70%) como en el bug anterior. En conclusión el throughput fue de 3 bugs entregados por semana.

Para la entrega 3 no llegamos a completar nuevos requerimientos.

Los resultados de la entrega 4 son los siguientes: la *feature* alta de producto tuvo un Lead time de 7 días, el Cycle time fue de 5 días por lo que el Flow efficiency fue de $7/5 = 0,71$ (0,71). Cometimos algunos errores en los cálculos de las métricas de baja de producto, modificación de producto y compra de producto, ya que estos requerimientos no llegaron a la columna Done. **Los valores indicados en el documento Métricas de la entrega 4 son producto de una confusión.**

Dejamos el link al documento principal de la entrega 4 donde se puede llegar a los demás artefactos de la entrega: [link](#)

En términos generales, los resultados de la entrega no cumplieron con los objetivos establecidos. Los requisitos de modificación y compra de producto no se llegaron a implementar y el test de integración contempla únicamente un criterio de aceptación.

Nos sentimos en la obligación de tomar lo sucedido como lección, reflexionar sobre las decisiones tomadas y aprender de nuestros errores para superar los posibles retos a los que nos podamos enfrentar en el futuro.

Reflexiones

Ahora presentamos algunas reflexiones más elaboradas en formato DAKI:

¿Qué hicimos mal?:

Consideramos que no tomamos los trabajos con suficiente responsabilidad, en otras palabras, no supimos dimensionar bien las propuestas y esto nos perjudicó mucho, pensamos que las tareas eran más sencillas de lo que en realidad fueron.

Otra de las cosas que hicimos mal fue comenzar las tareas tarde. El error más presente en todo el proyecto. consideramos que hay varias razones para esto, una de ellas es la mencionada en el párrafo anterior, otra razón fue la cantidad de otras entregas y obligaciones exigidas por otras materias que desviaron la atención de este obligatorio.

¿Qué no hicimos y deberíamos comenzar a hacer?:

Una de las cosas que no hicimos fue comenzar las tareas temprano. De volver a hacer el obligatorio, consideramos que empezar de inmediato es fundamental, por lo menos con el fin de sacar dudas y enfrentarse a la herramienta desconocida con antelación y poder evacuar dudas en las siguientes clases.

¿Qué hicimos bien y deberíamos continuar haciendo?

Consideramos que hicimos bien en hacer llamadas y trabajar en equipo al mismo tiempo. Esto fue un tema recurrente en las retro y creemos que fue muy positivo. También es verdad que no podemos irnos hacia el otro extremo y trabajar únicamente cuando estamos todos los integrantes presentes. Trabajar de manera independiente permite avanzar con más velocidad y llegar a las fechas de entrega con comodidad.

Pensamos que hemos hecho un buen balance entre trabajar en grupo e individualmente, ya que las llamadas servían mucho para cuando nos bloqueábamos con alguna tarea y nuestros compañeros nos ayudan a eliminar ese obstáculo para poder continuar trabajando de manera individual.

Consideramos que el manejo del repositorio con el desarrollo de Trunk-Base development fue un punto a favor considerando que era una técnica nueva que

consideramos oportuna probar, aprovechando la característica de entrega y feedback continuos.

¿Qué hicimos relativamente bien, pero podríamos mejorar?

El uso del tablero fue algo que utilizamos bien pero sin duda podríamos mejorar. Si bien el tablero fue de gran ayuda al momento de definir las nuevas funcionalidades con su narrativa y sus respectivos escenarios, muchas veces nos pasó que olvidábamos mover las tarjetas del tablero y al terminar o avanzar en una tarea, la tarjeta seguía en una columna intermedia y no en la columna correspondiente como debería. Esto se debió a que no estamos del todo acostumbrados a tener un tablero Kanban y por momentos nos olvidamos de su existencia.

La creación del pipeline fue algo que hicimos bien ya que llegamos a implementarlo a nivel de backend y brindar feedback de cada integración al que se hacía al backend, podríamos mejorarlo implementándolo a nivel de frontend, que fue algo que no pudimos hacer por errores en las rutas de archivos y con el fin de no perder tiempo lo mantuvimos solo en el back.

Lecciones aprendidas

De las anteriores reflexiones podemos comentar algunas lecciones aprendidas.

Lección n°1: Hacer sprint planning al comienzo de la iteración.

Una sprint planning al comienzo de la iteración es fundamental para que todos los integrantes del equipo sepan en donde estamos parados y hacia dónde debemos ir. Una vez que se tienen claros los objetivos de la iteración, cada integrante puede trabajar a su ritmo con tranquilidad y avanzar sabiendo que el próximo integrante podrá seguir el camino definido al comienzo del sprint.

Particularmente nos pasó, al comienzo de la tercera entrega, que por no hacer una llamada grupal, no definimos los escenarios correctamente y esto conllevó a modificarlos el día antes de la entrega porque no estaban bien definidos retrasando el avance del código de Specflow y todas las tareas subsiguientes.

Lección n°2: No considerar al trabajo más sencillo de lo que en realidad es.

Durante la cuarta entrega subestimamos la dificultad de Selenium y es una de las razones por las que implementamos un solo test de Selenium. No es una herramienta difícil de implementar pero sin duda alguna conlleva un tiempo de

aprendizaje que no consideramos. Esto sucedió por no investigar la herramienta con suficiente profundidad, los ejemplos practicados en clase nos dieron una sensación de confianza que se desvaneció rápidamente al enfrentarnos a la tarea real.

Al testear con Selenium tuvimos problemas para poder agregar los pasos (click) que luego debía seguir, por lo que algunos escenarios no pudieron ser testeados con esta herramienta.

Lección n°3: Aprovechar el tablero Kanban.

Como lo indicamos en las reflexiones, el tablero Kanban se utilizó de manera correcta pero no de la manera más eficiente. El tablero es una herramienta muy útil que permite la comunicación con los demás integrantes del equipo y llevar un control del estado del proyecto.

Nos encontramos en muchas ocasiones con tarjetas perdidas por el tablero, tarjetas desactualizadas y a veces incompletas. Debemos aprovechar al máximo la utilidad y comodidad que brinda el tablero para administrar mejor el trabajo. Consideramos que la razón es la inexperiencia con esta herramienta ya que ningún integrante del equipo trabaja con alguna herramienta similar. En conclusión, el tablero es la herramienta del proceso DevOps por excelencia y debe ser tomado en cuenta, revisado y actualizado en todo momento.

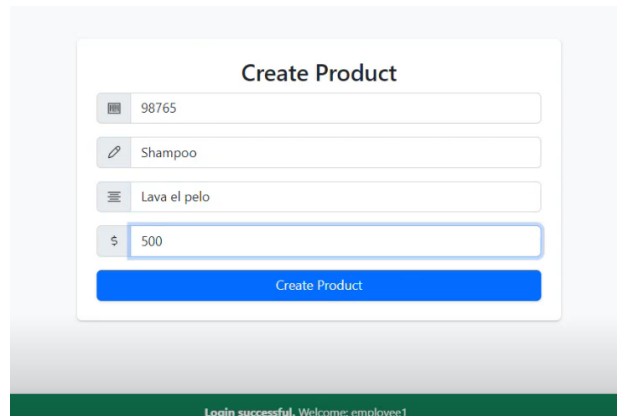
Lección n°4: No dejar el trabajo para último momento.

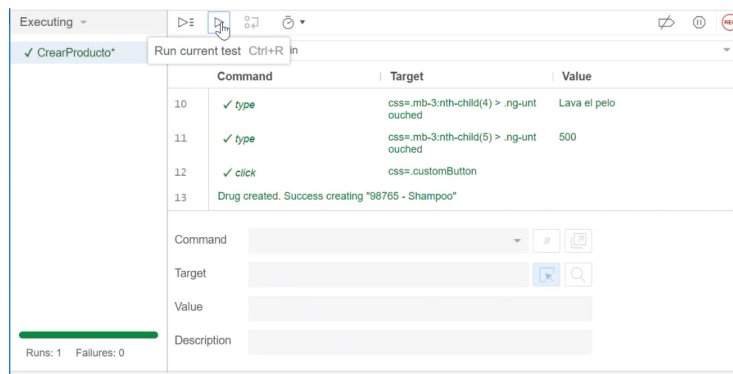
Será la última lección pero no la menos importante. Sin duda que el factor tiempo fue un tema recurrente en las retrospectivas y gran parte de las cosas que no llegamos a completar fueron ocasionadas por no saber administrarlo correctamente.

Los tests de integración con Selenium son testigo de nuestra incapacidad para administrar el tiempo ya que no logramos desarrollar más tests (alta de producto inválido y baja de productos) por estar programando a último momento, el día de la entrega, los escenarios en Selenium. Tuvimos que dejar de hacer los tests porque de seguir usando la herramienta no habríamos logrado entregar en tiempo y forma. Dejamos algunos casos que amplían lo anteriormente mencionado:

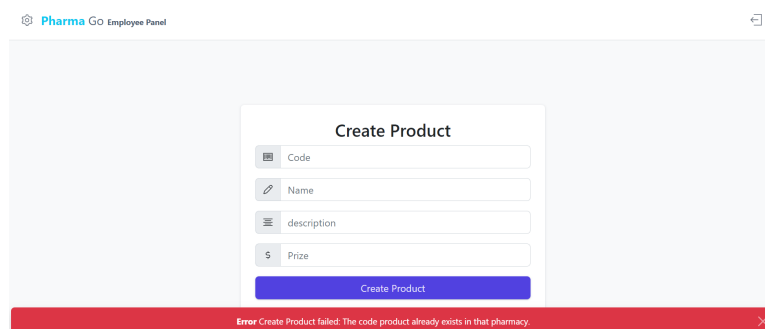
- **Alta producto:** en este caso pudimos testear bien el caso correcto, pero no llegamos a testear el caso incorrecto.

Caso correcto:





El caso que no se pudo testear fue producto ya existente:



- **Baja producto:** Llegamos al paso de *Integration Test* pero se nos dificultó a la hora de agregar los pasos con click, ya que nos tiraba errores y no nos dábamos cuenta de que era lo que estaba fallando.

Conclusiones

Las conclusiones obtenidas de este proyecto son de vital importancia para nuestro equipo. A pesar de todos los desafíos que experimentamos y del retraso inicial en las respectivas entregas, en la implementación de las prácticas como BDD (Desarrollo Basado en el Comportamiento) y la integración de Selenium, los resultados obtenidos pueden no ser perfectos pero fueron de gran enseñanza en la forma de trabajar para cada uno de nosotros, pensando en no cometer los mismos errores nuevamente en procesos de desarrollo futuros.

Hemos aprendido lecciones muy valiosas y adquirido conocimientos fundamentales los cuales nos han permitido obtener un gran progreso en la aplicación y comprensión de los conceptos teóricos y prácticos vistos en clase. La utilización de las distintas herramientas como SpecFlow, Github Actions y Selenium ha demostrado ser esencial para la automatización y mejora de nuestras prácticas de desarrollo y pruebas, no solo nos ayudaron a optimizar nuestro tiempo y esfuerzo, sino que también han contribuido a elevar la calidad del software entregado.

En cuanto a la colaboración y comunicación interna del equipo, tanto las dailies como las retrospectivas desempeñaron un papel fundamental en la identificación rápida y eficiente de problemas, generando un aumento significativo en la productividad. Algo a destacar en nuestro equipo es el optimismo y la fuerza de voluntad por entregar lo mejor que se pudo lograr, en ningún momento a pesar de los fallos se pensó en abandonar. La comunicación se mantuvo presente a lo largo de todas las iteraciones con todos los miembros del equipo.

A pesar de los desafíos que presentamos al no poder utilizar las herramientas como BDD y Selenium de manera óptima, es importante destacar que esta experiencia ha sido un valioso aprendizaje para el equipo. El reconocimiento de nuestras limitaciones en la implementación a tiempo de estas herramientas nos brinda una perspectiva valiosa sobre las áreas que debemos hacer hincapié para lograr la mejora continua.

Este proceso de aprendizaje nos permitió identificar no solo las oportunidades de mejora técnica, sino también las áreas en las que podemos fortalecer la planificación y gestión del tiempo. Aunque no pudimos aprovechar plenamente estas herramientas en este proyecto específico, estamos comprometidos a aplicar estos conocimientos para mejorar nuestros enfoques en proyectos futuros.

Estamos conscientes de las limitaciones que tuvimos durante las entregas, las cuales nos impulsan a buscar soluciones efectivas y eficientes para la implementación en proyectos futuros de las herramientas mencionadas anteriormente.

Guía de Instalación:

Para la instalación se deben seguir los siguientes pasos:

- 1) Para abrir la base de datos se deben seguir los siguientes pasos: Obtener el archivo *.bak*. Se copia a la siguiente ruta: *C:\Program Files\Microsoft SQL Server\MSSQL15.SQLEXPRESS\MSSQL\Backup*. Luego abrir Microsoft SQL Server allí hacer click izquierdo en la carpeta Database de la base de datos y ahí se va a la opción de restore database, luego se selecciona device, los tres puntitos y luego en la nueva ventana presionamos Add, se selecciona *PharmaGoDb.bak* y damos Ok a todo hasta terminar el proceso.
- 2) Para ejecutar el backend de la aplicación debemos seguir los siguientes pasos: Debemos ir a la carpeta *Código, Backend, PharmaGo.WebApi, bin, Release, net6.0* y buscamos *PharmaGo.exe* que es el ejecutable de la aplicación.

- 3) Para la ejecución del frontend debemos seguir los siguientes pasos: En la consola del sistema abrimos la carpeta del proyecto, luego nos movemos hacia la carpeta frontend y ejecutamos el comando `ng serve`.
- 4) Para la ejecución de las pruebas se siguen los siguientes pasos: En la consola `bash` nos movemos a la carpeta Código, Backend, luego `SpecFlowProject2` y ejecutamos el comando `dotnet test`.

```
mile2@DESKTOP-561DLP9 MINGW64 ~/OneDrive/Escritorio/Semestre7/Ingenieria de Software Agil 2/Obligatorio/230263-251105-254813/Codigo/Backend/SpecFlowProject2 (develop|MERGING)
```

```
Con error! - Con error: 5, Superado: 8, Omitido: 0, Total: 13, Duración: 6 s - SpecFlowProject2.dll (net6.0)
```

Estos errores que encontramos pueden deberse a que ese dato específico ya existe en la base de datos. La manera de corregir estos errores es cambiar el dato en los ejemplos del archivo `features.spec` de la prueba en `specflow`.