



Universidad ORT Uruguay

Facultad de Ingeniería

Informe de avance 4

Ingeniería de Software Ágil 2

[Repositorio en Github](#)

Los distintos documentos que forman parte de la entrega se encuentran dentro del repositorio Documentación/Entrega 4

Melissa Molinari – 229896

María Agustina Lamanna – 223040

Sebastian Daners - 255927

Profesores: Álvaro Ortas, Carina Fontán

Grupo: M7A

Octubre 2023

Índice

Índice	2
Definición del proceso de ingeniería en el contexto de KANBAN	2
Designación de roles	3
Explicación del Tablero y su vínculo con el proceso de ingeniería	3
Nuevas funcionalidades	3
Resolución de bugs	4
Tareas	4
Nueva versión del pipeline en GitHub Actions	4
Casos de prueba con Selenium	6
Ejecución de las pruebas	6
Métricas	7
Heurística Flow efficiency	7
Bugs	7
Exportador - archivo generado	7
Funcionalidad Login/SignUp no accesible desde Home	7
Solicitudes de stock negativas	7
Métricas promediadas	8
Consideraciones respecto a la entrega	8
Nuevas funcionalidades	9
Alta producto	9
Baja producto	9
Modificación producto	9
Compra producto	9
Métricas promediadas	9
Conclusiones	10
Review	10
Retrospectiva	11
Conclusiones	12

Definición del proceso de ingeniería en el contexto de KANBAN

Designación de roles

Para esta entrega, al igual que para las anteriores, se definieron los roles que asumiría cada uno de los integrantes.

Se definió el rol de SM (Scrum Master), el mismo lo adoptó Sebastián Daners. Para esta entrega, al igual que las anteriores, el Scrum Master realizó tareas de moderador en la *ceremonia de retrospectiva*. Fue el encargado, junto con el resto del equipo, de determinar las mejoras a realizar para las próximas entregas, a nivel de equipo.

Por otra parte, se definió el rol de PO (Product Owner), el mismo lo adoptó Melissa Molinari. Para esta entrega se realizó una *ceremonia de revisión*, en donde se le mostró al PO la implementación de las nuevas funcionalidades. De acuerdo con los criterios de aceptación definidos para cada una de ellas, el PO se encargó de aceptar aquellos requerimientos que cumplieran satisfactoriamente con dichos criterios.

Explicación del Tablero y su vínculo con el proceso de ingeniería

Con el fin de unificar los distintos tipos de actividades que se pueden realizar dentro de un proyecto (tareas, resolución de bugs e implementación de nuevas funcionalidades) se creó un tablero que agrupa las columnas que recorren las tarjetas correspondientes. Para la creación del mismo, se reunieron las columnas utilizadas en las distintas iteraciones del proyecto. El mayor beneficio de este nuevo tablero es que nos permite visualizar en el mismo lugar todas las actividades que se desarrollan durante el proyecto.

Para comenzar se tiene una columna *Backlog* donde se crean las tarjetas correspondientes las tareas, bugs y nuevas funcionalidades.

Nuevas funcionalidades

Luego, existe un grupo de columnas para las funcionalidades desarrolladas por medio de **BDD** las cuales son:

Requirement definition: aquí se escriben las narrativas y casos de uso siguiendo los pasos CCC (Card specification, Conversation y Confirmation). Tiene como output la User Story correspondiente al requerimiento, con su narrativa y criterios de aceptación.

Test cases implementation: aquí se generan los casos de prueba. Tomando como ejemplo lo realizado en entregas anteriores, se debe traducir los criterios de aceptación a formato cucumber para poder ser ingresados en la herramienta Specflow.

Application implementation: aquí se codifica e implementa la nueva funcionalidad.

Automation testing: aquí se ejecutan los tests implementados en el paso de Test cases implementation.

Refactoring: aquí se posicionan las tarjetas que al ejecutar los tests de Automation testing alguno de ellos quedó con estado/color rojo. En esos casos, en los que se encontraron bugs/cosas que no funcionaban correctamente, se comienza el ciclo nuevamente.

Done: aquí se posicionan las tarjetas correspondientes a las user stories que están correctamente implementadas y finalizadas.

Resolución de bugs

El proceso de **arreglar bugs** requiere que las tarjetas creadas para ellos pasen desde el **Backlog** a la columna **TDD** y luego a **Done**. La columna *TDD* significa la implementación del código necesario para crear los test y la funcionalidad siguiendo el enfoque TDD.

Tareas

Por último, las tareas hacen un recorrido en el tablero por las columnas **To Do** , **In Progress** y **Done**. Por tareas nos referimos a actividades por fuera del desarrollo de funcionalidades como tal sino que actividades como la documentación, review y retrospectiva.

Nueva versión del pipeline en GitHub Actions

Para esta entrega no se realizaron cambios al pipeline. Si se quisiera, se podría agregar una tarea para que se ejecutarán automáticamente las pruebas de Selenium. En nuestro caso, esto no es posible ya que no hicimos deploy del sistema, por lo tanto se carecen de los datos necesarios para que se ejecuten automáticamente dichas pruebas.

En caso de que se hiciera deploy del sistema, se podrían agregar las siguientes líneas de código al script para que las mismas se ejecutarán de forma automática:

- name: Setup Selenium and Run Tests

run: |

sudo apt-get update

sudo apt-get install -y chromium-browser

```
sudo apt-get install -y chromium-chromedriver
npm install -g selenium-side-runner

selenium-side-runner -c "goog:chromeOptions.args=[--no-sandbox, --headless,
--disable-gpu, --disable-dev-shm-usage]"
Código/Backend/PharmaGo.Selenium/PharmaGo.Selenium.side
working-directory: Código/Backend
```

Además a los paths se debe agregar:

```
- '**.side'
```

Para que pueda ejecutar los archivos con esa extensión.

Casos de prueba con Selenium

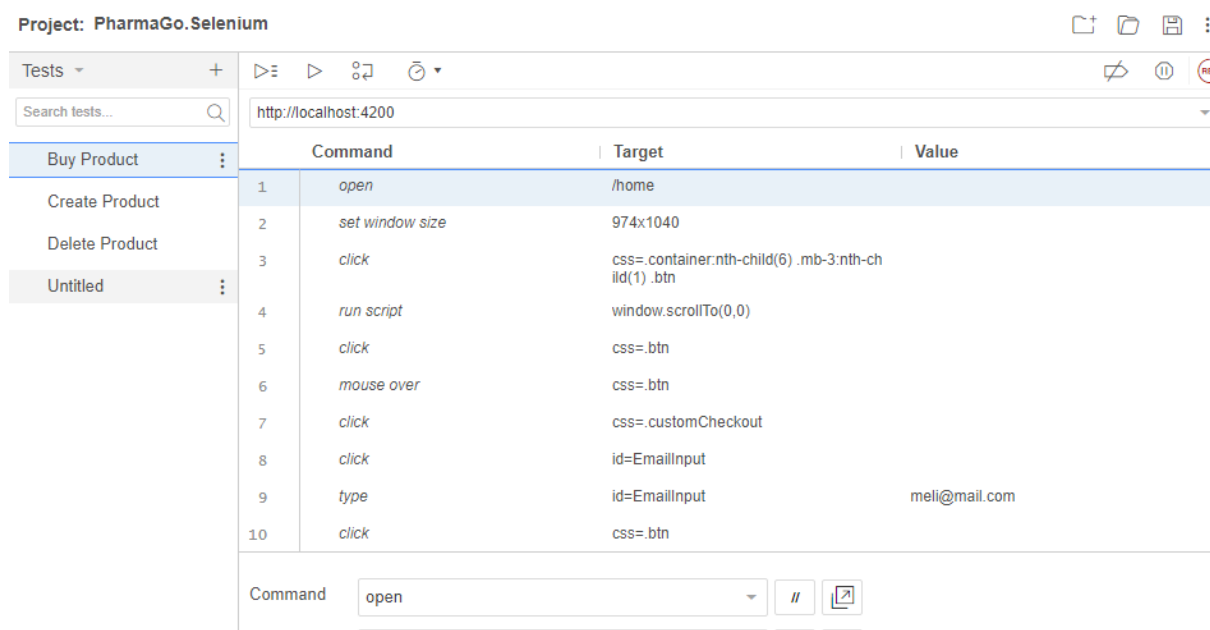
Para cada una de las funcionalidades implementadas en la entrega 3, se crearon los casos de prueba utilizando la herramienta *Selenium*. En el siguiente [link](#), se encuentra un video evidenciando la ejecución de las pruebas creadas.

Ejecución de las pruebas

En la siguiente ruta se encuentra un archivo donde se contienen las pruebas de Selenium.

Código\Backend\PharmaGo.Selenium\PharmaGo.Selenium.side

Una vez ubicado el archivo con las pruebas se debe abrir utilizando la extensión de Chrome de Selenium. Luego de abrir el archivo se verá lo siguiente:



Arriba la izquierda donde dice *Tests* se deber seleccionar *Test Suites* en esa sección se encontrarán colecciones de tests de productos,hay 4 suites, la primera incluye los casos donde se realizan las acciones de Creación, Modificación, Eliminación y compra Correctamente. Las otras 3 colecciones incluyen los casos de error para Creación, Modificación y Compra. Eliminación no tiene casos de errores desde el front por cómo se diseñó la página.

Métricas

Con los datos tomados en las entregas 2 y 3 se calcularon las siguientes métricas. Las mismas se encuentran identificadas con el título *Bugs*, correspondientes a los bugs solucionados en la entrega 2. Las nuevas funcionalidades implementadas en la entrega 3, se encuentran identificadas con dicho nombre.

Heurística Flow efficiency

Para esta métrica, no existen valores estrictos para considerar como buenos o malos. Sino que dependen del proyecto y del análisis de cómo el valor va progresando a lo largo del proyecto. Sin embargo, hay algunos valores que se toman como indicadores para tomar acciones de mejoría. Así lo establece el artículo “The Flow Efficiency Chart”, en dónde establece que valores menores al rango de 35%-40% son indicadores de que se deben realizar mejoras en el equipo.

En el artículo “Our survey says... uncovering the real numbers behind flow efficiency”, se establecen rangos en dónde consideran el flow efficiency como bajo (casos en donde dicho valor es menor a 20%), medio (entre 20%-40%) y muy alto (mayor a 60%). Pero el mismo no establece si hay algún valor mejor que otro.

Fuentes consultadas: [The Flow Efficiency Chart – Kanbanize Knowledge Base](#) , [Our survey says...uncovering the real numbers behind flow efficiency | by Nick Brown | ASOS Tech Blog | ASOS Tech Blog \(medium.com\)](#)

Bugs

Exportador - archivo generado

Lead Time: 2 días

Cycle Time: 1 día

Flow efficiency (Cycle time / Lead Time) : 50%

Funcionalidad Login/SignUp no accesible desde Home

Lead Time: 2 días

Cycle Time: 1 día

Flow efficiency (Cycle time / Lead Time) : 50%

Solicitudes de stock negativas

Lead Time: 2 días

Cycle Time: 1 día

Flow efficiency (Cycle time / Lead Time) : 50%

En esta entrega, se implementaron las soluciones para los bugs en un mismo día. Asimismo, cada una de las tarjetas correspondientes también fue creada el mismo día. El lead time de estas actividades coinciden ya que al tratarse de bugs, no teníamos la US correspondiente a cada uno. Por lo tanto, al no tener los criterios de aceptación correspondientes, tomamos la decisión de que para lograr considerar el bug como solucionado (es decir, pasarlo a la columna *Done*), se debía tener la validación del Product Owner. Es por ello, que todas las tarjetas fueron trasladadas a la columna *Done* durante la Review. Por lo tanto al haber sido creadas las tarjetas el mismo día (es decir puestas en la columna *To Do*) y haber sido validadas en el mismo momento (durante la *Review*) todas poseen el mismo valor para la métrica *Lead Time*.

Métricas promediadas

Se promediaron los valores de cada uno de los bugs, para llegar a métricas más generalizadas sobre la entrega.

Lead Time: 2 días

Cycle Time: 1 día

Flow efficiency (Cycle time / Lead Time) : 50%

Con el valor de flow efficiency al que se llegó en la entrega, teniendo en cuenta las heurísticas planteadas anteriormente, podemos considerar que dicho valor es medio. Además nos permite concluir que no es necesario aplicar mejoras inmediatas en la metodología de trabajo del equipo, ya que el valor no se encuentra dentro del rango de 35%-40%. De igual manera se debería continuar monitoreando los valores del equipo para continuar con dicho análisis.

Throughput: En esta entrega, se resolvieron 3 bugs en 2 días. Por lo tanto, se le entregó al cliente 3 funcionalidades implementadas correctamente.

Consideraciones respecto a la entrega

Por otra parte, en la entrega 2 que fue dónde se resolvieron estos bugs se incluyeron unas métricas de análisis en dónde se incluían todas las tareas realizadas durante esa entrega. En este caso, nos estamos centrando en los bugs específicos mencionados anteriormente. Por lo tanto, esto nos generó una diferencia en las métricas a las que llegamos.

Nuevas funcionalidades

Alta producto

Lead Time: 5 días

Cycle Time: 5 días

Flow efficiency (Cycle time / Lead Time) : 100%

Para esta entrega estuvimos varios días aprendiendo a utilizar la herramienta Specflow. Esto nos generó un cuello de botella por las demoras que esto nos generó. Además por la metodología de trabajo que elegimos (one-piece flow) no podíamos comenzar con nuevas actividades si la previa no había sido finalizada. Por lo tanto se generaron demoras para finalizar las tareas.

Baja producto

Lead Time: 2 días

Cycle Time: 2 días

Flow efficiency (Cycle time / Lead Time) : 100%

Modificación producto

Lead Time: 1 día

Cycle Time: 1 día

Flow efficiency (Cycle time / Lead Time) : 100%

Compra producto

Lead Time: 1 día

Cycle Time: 1 día

Flow efficiency (Cycle time / Lead Time) : 100%

Al haber utilizado la metodología *one piece flow*, y haber creado las US en el paso *Requirement Definition* nos llevó a tener que finalizar con una tarea para poder comenzar con la siguiente. Por ello, el Lead Time de cada una de las funcionalidades coincide con su Cycle Time.

Métricas promediadas

Se promediaron los valores de cada una de las nuevas funcionalidades implementadas, para llegar a métricas más generalizadas sobre la entrega.

Lead Time: 2.25 días

Cycle Time: 2.25 días

Flow efficiency (Cycle time / Lead Time) : 100%

Throughput: Se añadieron 4 nuevas funcionalidades en 7 días.

Conclusiones

El uso de one-piece flow nos permitió llegar a un valor de flow efficiency elevado, según las heurísticas investigadas, ya que se trabajó en una tarea por vez asegurando así que no se podían comenzar a implementar otras funcionalidades hasta que no se terminará con la actual. Por lo tanto no se generaron esperas para otras funcionalidades para avanzar y el Lead Time quedó igual al Cycle Time en todas las funcionalidades.





Review

Se realizó una reunión en donde los desarrolladores le mostraron al Product Owner, en este caso Melissa Molinari, cómo se implementaron las pruebas de integración utilizando la herramienta Selenium. En este caso, se ejecutaron todos los tests y estos fueron aceptados por la PO ya que lograban mostrar todos los casos de uso para las funcionalidades deseadas.

En el siguiente [link](#) se puede encontrar el video de la reunión.

Retrospectiva

Se realizó una reunión con todos los integrantes del equipo para llevar a cabo la retrospectiva. Para ello, se utilizó la herramienta MetroRetro con su plantilla del método DAKI. Luego de un tiempo, los integrantes plantearon las siguientes ideas:

 Drop What do you think the team should drop?	 Add What do you think the team should add?
 Keep What do you think the team should keep?	 Improve What do you think the team should improve?

Comunicacion por fuera del grupo

encontrar mejores canales de comunicacion con los profes o preguntar mas dudas en clase

Entregar todo a tiempo

Consultar las dudas a los profesores

coordinar mejor las reuniones

Mejorar la organización para reunirnos

La comunicación en el equipo esta última entrega no fue muy buena

Cada integrante tuvo su momento para explicar las ideas que planteó y en conjunto se llegaron a las siguientes acciones para implementar, mejorar y mantener durante la próxima entrega:



En el siguiente [link](#) se puede encontrar el video de la reunión.

Conclusiones

Se logró cumplir satisfactoriamente con los objetivos planteados al inicio de la iteración, durante la etapa de planificación. Se lograron crear satisfactoriamente todos los tests de integración utilizando la herramienta Selenium. Además, se verificó que estas cumplieran y contemplaran todos los criterios de aceptación determinados para cada funcionalidad, durante la entrega pasada.

En este caso, no se encontraron dificultades al momento de familiarizarnos con la herramienta Selenium ya que su interfaz permite un uso efectivo de la misma. Por lo tanto, la creación de los tests de integración fue un proceso más ameno y rápido de lo esperado por los integrantes del equipo. El aprendizaje sobre la utilización de esta herramienta fue uno de los mayores aprendizajes de esta entrega.

Por otra parte, se calcularon todas las métricas correspondientes a los bugs solucionados durante la entrega 2 y las funcionalidades implementadas durante la entrega 3. Pudimos observar que las métricas referentes a la entrega 3, en dónde se utilizó la metodología one piece flow generó que se lograra un Flow efficiency alto. Esto se debe a que one piece flow establece que no se puede comenzar a implementar una nueva funcionalidad hasta no haber terminado la actual. Por lo tanto, generó que el Lead Time fuera igual al Cycle Time.

En comparación con la entrega 2, en dónde utilizamos una metodología de trabajo diferente ya que se implementaron las soluciones a los bugs de forma independiente. Podemos observar que en este caso, el Flow Efficiency al que llegamos no es tan elevado ya que el trabajo en los mismos de forma independiente llevó a que el Lead Time se viera modificado ya que consideramos que la tarea estaba finalizada cuando el Product Owner la validaba.

Como aprendizaje general de esta última parte de mediciones de métricas, podemos concluir que al utilizar la metodología de trabajo one piece flow se logró un valor de Flow Efficiency mayor que cuando utilizamos otra metodología de trabajo. Por lo tanto, como esta métrica mide cuán eficiente es el pasaje por cada una de las columnas de una user story, podríamos considerar que la metodología de trabajo one piece flow fue más eficiente.