Universidad ORT Uruguay
Facultad de Ingeniería

Obligatorio - Ingeniería de Software Ágil 2

Link al repositorio

Entrega 4

Integrantes:
Danilo Biladoniga - 231749
Tomás Núñez - 257564
Germán Oller - 242312

Índice

Definición del proceso de ingeniería en el contexto de KANBAN

Roles dentro del equipo

Mantuvimos los mismos roles de la anterior entrega.

Miembro	Rol/es
Danilo Biladóniga	Developer, Tester
Tomás Núñez	Developer, Tester, Product Owner
Germán Oller	Developer, Tester

Ceremonias

Mantuvimos los 3 tipos de ceremonias realizadas en la anterior entrega, las standup, la retrospective y la review.

Explicación del tablero y su vínculo con el proceso de ingeniería

Para esta última entrega, modificamos el tablero para reflejar todo el proceso de desarrollo.

To Do	RD	TCI	AI	AT	Ref	TDD	IT	Doing	Done

Este tablero es el resultado de unir todos los tableros utilizados a lo largo del proyecto. El tablero puede tener 3 tipos de elementos, según los cuales realizarán cierto recorrido por el mismo.

- Nuevas features (User Stories): se desarrollan siguiendo BDD, y luego se realiza el testing de integración, por lo que el recorrido es:

To Do -> RD (Requirement Definition) -> TCI (Test Cases Implementation) -> AI (Aplication Implementation) -> AT (Automatization Testing) -> Ref (Refactoring) -> IT (Integration Testing) -> Done

- Bug Fixes: utilizamos TDD para arreglar los bugs, y posteriormente realizamos testing de integración, resultando en este recorrido:

To Do -> TDD -> IT -> Done

- Tareas Auxiliares: estas son otras tareas que no implican codificación (cómo puede ser hacer un video o realizar un informe), lo que lleva a este recorrido:

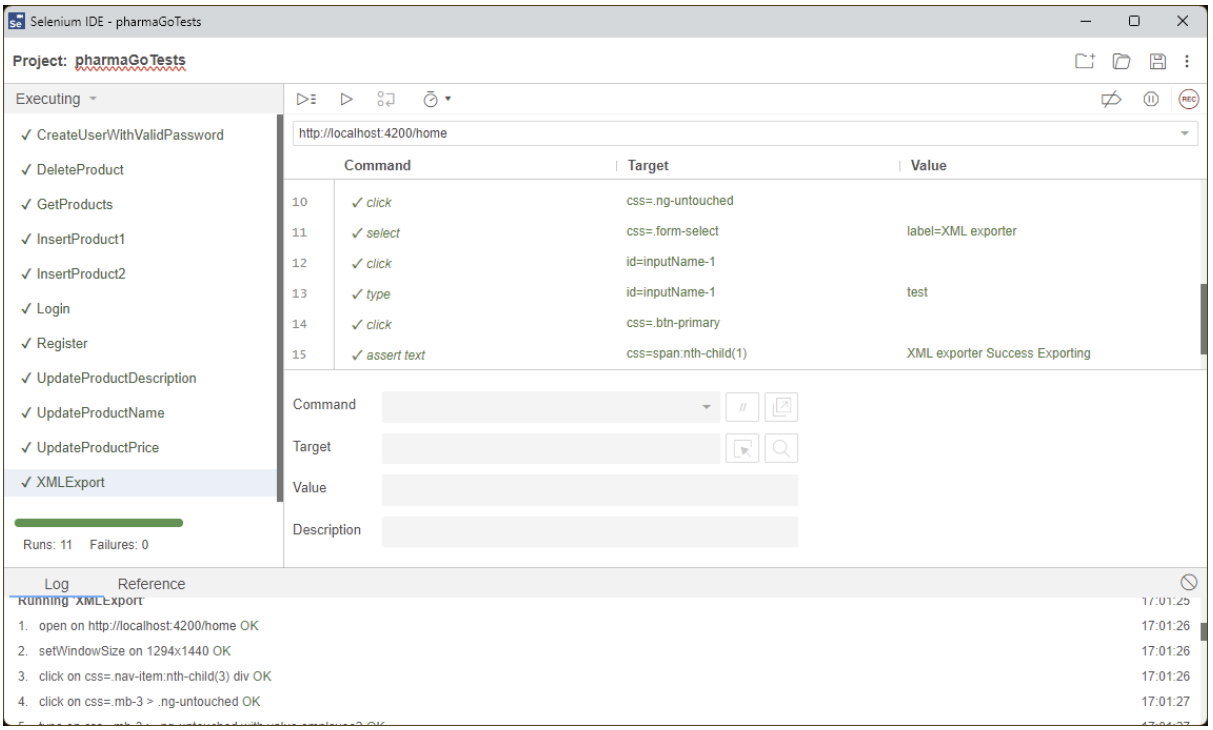
To Do -> Doing -> Done

Configuración del pipeline y su vínculo con el tablero

No modificamos la configuración de nuestro pipeline. Al realizar un Pull Request, es necesario que tanto el backend cómo el frontend compilen, y los test unitarios del backend deben arrojar resultados positivos. En caso de que alguna de las condiciones no se cumpla, no es posible realizar un merge.
El objetivo de fijar estas condiciones en el pipeline es minimizar la posibilidad de cometer errores al agregar código nuevo. Idealmente, se realiza el Pull Request en el momento que la issue pasa a la columna Done, habiendo recorrido previamente todo el ciclo de BDD y el testing de integración (en caso de ser una nueva feature) o TDD y testing de integración (en caso de ser un bug fix).
Nos planteamos agregar tanto los test de SpecFlow cómo los de Selenium al pipeline. Sin embargo, debido a la complejidad que tiene relacionada con el uso de la base de datos, decidimos descartar esta idea.

Evidencia de ejecución de los casos de prueba

A continuación se muestra una imagen de los casos de prueba ejecutados usando Selenium IDE los cuales se corresponden con los escenarios planteados en specflow.



Consideraciones previas para la ejecución de las pruebas:

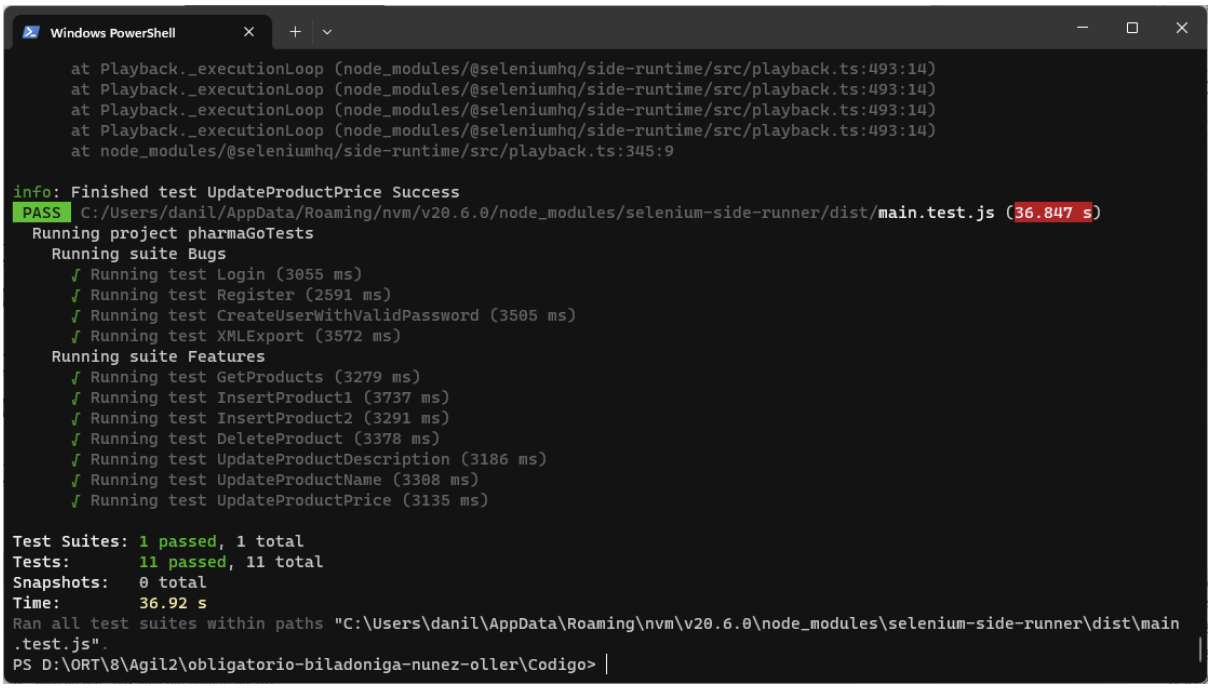
1. Tener selenium IDE instalado
2. Tener los drivers instalados para el buscador correspondiente

Pasos para ejecutar las pruebas:

1. Ejecutar API en forma local
2. Ejecutar aplicación cliente en forma local
3. Abrir una terminal e ir al directorio de “Codigo” en la raíz del proyecto
4. Escribir el siguiente comando en la terminal y ejecutarlo
    a. selenium-side-runner -c "browserName=chrome acceptInsecureCerts=true" .\pharmaGoTests.side

(la flag acceptInsecureCerts=true se agrega para permitir acceder a direcciones http de la API)

Resultado del comando:



Métricas de DevOps

Analizaremos las métricas sobre las nuevas features y los bug fixes.

Lead Time

Esta métrica corresponde al tiempo desde que se crea una tarea (es decir, se coloca en la columna To Do del tablero) hasta que se finaliza (llega a Done).

- Fix Bug ID 7: 2 días
- Fix Bug ID 9: 5 días
- Fix Bug ID 18: 4 días

Promedio para Bug fixes = 4 días

- Alta de producto: 21 días
- Baja de producto: 19 días
- Modificación de producto: 21 días

Promedio para nuevas features: 20 días

Cómo podemos observar, hay una diferencia notoria en los lead times de las nuevas features comparado con el de los bug fixes. Esto se debe principalmente a la tecnología asociada; para reparar bugs, simplemente fue necesario reproducir el mismo y aplicar las correcciones pertinentes, utilizando TDD (técnica con la que el grupo ya estaba familiarizado). Sin embargo, para el desarrollo de nuevas features utilizamos la técnica BDD, con la herramienta de SpecFlow. El equipo necesitó de cierto tiempo para estudiar y aprender a utilizar esta herramienta, antes de comenzar con el desarrollo de las funcionalidades.

Cycle Time

Refiere al tiempo desde que se comienza a trabajar en una tarea (In progress) hasta que se completa (Done).

- Fix Bug ID 7: 1:30 horas
- Fix Bug ID 9: 1:30 horas
- Fix Bug ID 18: 1:30 horas

Promedio para Bug fixes = 1:30 horas

- Alta de producto: 5 días
- Baja de producto: 5 días
- Modificación de producto: 6 días

Promedio para nuevas features: 5 días

Al igual que con el Lead Time, vemos grandes diferencias entre el Cycle Time de los bug fixes y el de las nuevas funcionalidades. El motivo es el mismo que para la métrica anterior, las tecnologías asociadas a las tareas.

Flow Efficiency

La fórmula general de Flow Efficiency es Touch Time / Lead Time. Sin embargo, cómo nosotros no tenemos el cálculo del Touch Time, excepcionalmente utilizaremos la siguiente fórmula: Flow Efficiency = Cycle Time / Lead Time.

Flow Efficiency para los bugs:

- Flow Efficiency = 1.5 / (4 * 24) = 0.015625

Flow Efficiency para las features:

- Flow Efficiency = 5 / (20 * 24) = 0.010416666666666666

Multiplicando los valores anteriores por 100 para visualizar el Flow Efficiency en porcentajes:

Flow Efficiency para los bugs:

- Flow Efficiency = 1.5 / (4 * 24) = 1.5625

Flow Efficiency para las features:

- Flow Efficiency = 5 / (20 * 24) = 1.0416666666666666

Esto conlleva a que el 98.5% en el caso de bugs y 99% en el caso de las features aproximadamente fue tiempo del ciclo en el cual hemos estado esperando algo para poder trabajar.

Throughput

El Throughput mide la cantidad de unidades de trabajo terminadas en un periodo específico. En nuestro caso usaremos como períodos a cada una de las entregas, es decir Entrega 1, Entrega 2, Entrega 3 y Entrega 4. No estamos ante periodos estáticos, esto se debe a que en situaciones laborales un cliente nos puede dar plazos distintos para cada entrega, en este caso los docentes nos asignaron periodos de entregas que variaron entre 1 y 3 semanas de duración.

<u>Entrega 1:</u>
Throughput para los bugs: 0
Throughput para las funcionalidades: 0

<u>Entrega 2:</u>
Throughput para los bugs: 3 (se arreglaron 3 bugs en la entrega 2)
Throughput para las funcionalidades: 0

Entrega 3:
Throughput para los bugs: 0
Throughput para las funcionalidades: 3 (se añadieron 3 funcionalidades, alta, baja y modificación de productos).

Entrega 4:
Throughput para los bugs: 0
Throughput para las funcionalidades: 0