Universidad ORT Uruguay

Facultad de Ingeniería

Informe de avance de etapa: Entrega 4

Ingeniería de Software Ágil 2

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

Sofia Montero - 250618

Romina Pardo - 253445

Andres Wilchinski - 193694

Tutor: Carina Fontan y Alvaro Ortas

2023

https://github.com/IngSoft-ISA2-2023-2/250618_193694_253445

Índice

Índice **1**
Información sobre la entrega **3**
Objetivos del Proyecto 3
Entregas y Evaluación 3
Registro de actividades 3
Definiciones Iniciales **4**
Definición de Kanban 4
Roles asignados 4
Trunk Based 5
Explicación del tablero y su vínculo con el proceso de ingeniería 5
Uso de tableros por entrega 5
Proceso de ingeniería basado en BDD 5
Proceso de ingeniería basado en reparar bugs 6
Proceso de ingeniería basado en tareas 6
Explicación del pipeline y su vínculo con el proceso de ingeniería 7
Test exploratorios basado en KANBAN 7
Uso de Selenium 7
Casos de prueba 8
Bug 3: Rechazo Compras 8
Bug 4: Exportación de medicamentos 8
Bug 6: Código de invitaciones Administrador 8
User Story 1: Alta Producto 8
User Story 2: Baja Producto 10
User Story 3: Modificación de producto 10
User Story 4: Compra Producto 11
Resultados obtenidos 12
Estructura y Mantenimiento del repositorio 14
Creación del repositorio 14
Mantenimiento del repositorio 14
Uso de Documentos de Google 14
Uso de Hojas de Cálculos de Google 14
Métricas **15**
Podrá ver el seguimiento de esfuerzo y métricas realizadas aquí. **15**
Cycle Time, Lead Time, Flow Efficiency y Throughput 15
Bugs 15
User Stories 15
Comparativas 15
Conclusión 16
Kanban system lead time chart 17

Registro del trabajo y Esfuerzo 17
Análisis del trabajo **18**
Registro de las daily’s 18
24/10/2023: Mensaje de texto 18
26/10/2023: Mensaje de texto 18
28/10/2023: Llamada de teams 18
Registro de la Review 18
Registro de retrospectiva 18
Drop 19
Add 19
Keep 20
Improve 21
Registro de avance **23**
Resultados obtenidos en el período 23
Dificultades encontradas y formas de solución 23
Lecciones aprendidas y mejoras en el proceso 23
Con respecto a la entrega anterior 23

Información sobre la entrega

Objetivos del Proyecto

- Aplicar un marco de gestión ágil.
- Analizar la deuda técnica.
- Implementar un repositorio y procedimientos de versionado.
- Crear un pipeline con eventos y acciones.
- Integrar prácticas de QA en el pipeline y gestionar el feedback.
- Generar escenarios de testing desde la perspectiva del usuario.
- Automatizar el testing funcional o de caja negra.
- Reflexionar sobre DevOps

Entregas y Evaluación

El proyecto se divide en cinco entregas. Cada una se evaluará en función de su cumplimiento y calidad. Además, se llevarán a cabo retrospectivas después de cada entrega para reflexionar sobre el proceso y realizar mejoras continuas.

Informe Final

Al final del proyecto, se presentará un informe académico que documentará todas las actividades realizadas, lecciones aprendidas y conclusiones obtenidas a lo largo del proyecto.

Registro de actividades

En cada entrega se aplicarán ceremonias de gestión ágil que el equipo deberá incorporar y adaptar.

Se espera que cada integrante dedique un mínimo de 5 horas semanales a las actividades de ingeniería del proyecto (sin incluir gestión y retrospectivas). Antes de cada entrega, el equipo realizará una retrospectiva y confeccionará un informe de avance, que debe incluir:

- Registro de las actividades realizadas (fecha, horas, integrante).
- Resultados obtenidos en el período.
- Dificultades encontradas y formas de solución.
- Lecciones aprendidas y mejoras en el proceso.

Se debe realizar un informe académico que detalle las actividades realizadas y justifique las decisiones tomadas para cumplir con estos objetivos. Es esencial fundamentar todas las decisiones tomadas durante el proyecto.

Definiciones Iniciales

Definición de Kanban

Para este proyecto se decidió utilizar la metodología Kanban y su artefacto Kanban board. El mismo está formado por 2 ceremonias obligatorias, el stand up diario y la retrospectiva. En esta metodología no hay roles de product owner o scrum master como si lo hay en scrum pero como también la metodología se tiene que adaptar a las necesidades del equipo y no al revés decidimos agregar dichos roles ya que nos favorecía. El stand up diario se realiza todos los días y dura aproximadamente 15 minutos, en esta instancia los desarrolladores del equipo y cada uno da un breve resumen de en qué trabajaron el día anterior, en que van a trabajar hoy y también dicen si tienen alguna dificultad. Como no trabajabamos todos los días en este proyecto decidimos realizar stand ups cada 2 o 3 días. Otra ceremonia es la retrospectiva, que se realiza con el objetivo de que el equipo mejore. Aquí se aplica un sistema DAKI y además de ir el equipo de desarrollador, se agrega el scrum master, el cual conduce dicha ceremonia. Además de estas 2 ceremonias decidimos agregar la review, en esta le mostraremos al product owner como resolvimos los bugs o cuales son las nuevas funcionalidades agregadas al proyecto. Como mencionado anteriormente, el artefacto más importante de la metodología es la Kanban board, en este se puede visualizar fácilmente qué tareas o user stories fueron ya realizadas, se están haciendo o todavía no se han comenzado a hacer. Hay varios tipos de tableros, dependiendo de las necesidades y especificaciones que necesitemos según el tipo de tareas necesarias para cada entrega.

Kanban se basa en 6 principios bien claros.

- Visualizar el flujo de trabajo: esto es posible gracias al kanban board
- Limitar el WIP (work in progress): permite evitar los cuellos de botella y defectos rápidamente.
- Gestionar y medir el flujo: medir para ver si se mejora y en base a eso tomar medidas acordes.
- Implementar ciclos de feedback: para este principio las retrospectivas son clave.
- Explicar políticas y procedimientos.
- Mejora continua mediante la colaboración.

Roles asignados

Roles Posibles: SM, DESA, TES, PO

Todos los integrantes del equipo deben poseer los roles de DESA y TES. Por otro lado, en esta cuarta instancia, se ve pertinente que sean asignados los siguientes roles: Desarrollador, Tester, Product Owner y Scrum Master. Esta decisión fue tomada teniendo en cuenta la letra de la entrega y los artefactos recomendados para la misma. En el caso de Scrum Master, lo consideramos necesario debido a que una de las ceremonias, la retrospectiva, es liderada por el. Decidimos que el rol de Scrum Master rotará dependiendo de las entregas. A su vez, el product owner es necesario debido a que es necesario para el video de la revisión del sistema.

- **Sofia Montero**: Desarrollador, Tester, Product Owner
- **Romina Pardo**: Desarrollador, Tester, Scrum Master
- **Andres Wilchinski**: Desarrollador, Tester

Trunk Based

Debido a que en esta entrega comenzamos a codificar, creímos necesario definir la forma de trabajo en git. Para el proyecto, hemos decidido trabajar con trunk based development, una práctica de gestión del control de versiones en la que los desarrolladores incorporan pequeñas actualizaciones frecuentes a la rama principal. Al usar trunk based se optimiza para la productividad del equipo ya que todos trabajan en una misma área y los merge constantes hacen que los problemas de merge sean más concretos y simples de resolver. Debido al alcance del proyecto y de cada entrega los commits se realizan todos los días en los que el código es trabajado, no diariamente. Aunque trabajar con esta práctica puede resultar más complicado ya que todos los desarrolladores estamos acostumbrados a usar Git Flow consideramos que el esfuerzo extra vale la pena debido a los beneficios que trae.

Explicación del tablero y su vínculo con el proceso de ingeniería

Para la creación del tablero hemos realizado algunas mejoras y ajustes de las versiones anteriores del tablero. Esta cuarta versión consta de las siguientes columnas: Sprint Backlog, Requirements Definition, Test Cases Implementation, App Implementation, Automation Testing, Refactoring, TDD, Integration Testing, Doing, Done.

El objetivo del tablero es juntar en un solo tablero el camino de los distintos ítems que están involucrados en el proyecto: User stories, Bugs y Tareas. Los caminos de cada uno de los items están detallados en sus procesos de ingeniería.

Uso de tableros por entrega

Se creará un tablero por entrega utilizando la herramienta git project. La división de tablero por entrega nos permitirá poseer un control registrado del estado del proyecto en cada entrega. Podrá acceder al tablero de está aquí.

Proceso de ingeniería basado en BDD

El Behavior-Driven Development (BDD) se enfoca en el comportamiento del software desde la perspectiva del usuario, fomentando una colaboración estrecha entre desarrolladores, probadores y expertos en el dominio. Utiliza "historias de usuario" para describir el comportamiento deseado de la aplicación en un lenguaje comprensible, las cuales se convierten en pruebas automatizadas para verificar los criterios de aceptación, garantizando que el software se desarrolle teniendo en cuenta las necesidades del cliente. Las historias de usuario de BDD se representan como tarjetas en un tablero

Kanban, permitiendo un flujo de trabajo transparente y una respuesta ágil a cambios en los requisitos del cliente. También esto nos permite mantener una documentación actualizada del sistema.

El flujo de trabajo según el tablero de las User Stories es el siguiente: inicialmente, todas las user stories se agregan a la columna Sprint Backlog. Cuando se comienza a trabajar se pasa a la columna "Requirements Definition", donde se definen los requerimientos. Cuando comienza la implementación de los test las User Stories se pasan a la columna Test Cases Implementation. Luego, se comienza a programar el frontend y backend y se pasa a la columna App Implementation. Una vez que la implementación o la tarea es terminada se pasa a la columna Refactoring. Luego la user story pasa a la columna de integration testing, donde se testea la funcionalidad de punta a punta y como se integra con el sistema desde el front end para darla por terminada. Al completarse y pasar los test de integración, se coloca en la columna Done.

Proceso de ingeniería basado en reparar bugs

A la hora de reparar bugs el proceso es muy diferente ya que en lugar de BDD se usa TDD. Test Driven Development es una metodología de desarrollo de software que se centra en escribir pruebas automatizadas antes de escribir el código real de la aplicación. En resumen, el proceso de TDD sigue estos pasos:

1. **RED**: El desarrollador comienza escribiendo una prueba que describe una funcionalidad específica que desea implementar. Esta prueba inicial generalmente falla, ya que la funcionalidad aún no está presente.
2. **GREEN**: Escribir el código mínimo para pasar la prueba: El desarrollador escribe la cantidad mínima de código necesaria para que la prueba pase. Esto a menudo implica la implementación de la funcionalidad requerida.
3. **REFACTOR**: Una vez que la prueba pasa, el desarrollador puede mejorar el código manteniendo la confianza de que si las pruebas siguen pasando, la funcionalidad aún funciona correctamente.

En el caso de los Bugs, por ende, el flujo de trabajo en el tablero sigue un camino totalmente diferente. Inicialmente el Bug en cuestión se ingresa al Sprint Backlog, al comenzar a trabajar en la resolución del bug este pasa a la columna de TDD donde pasa por el proceso previamente mencionado. Luego pasa a Integration Testing para ser testeado en conjunto con el resto del sistema. Cuando el proceso está terminado pasa a la columna Done

Proceso de ingeniería basado en tareas

En el caso de las Tareas, nuevamente, el flujo de trabajo en el tablero sigue un camino diferente. Como los otros ítems, inicialmente se ingresa al Sprint Backlog. Al comenzar a trabajar en la tarea esta pasa a la columna de Doing. Finalmente, cuando la tarea está terminada pasa a la columna Done.

Explicación del pipeline y su vínculo con el proceso de ingeniería

Para la pipeline corregimos para que en github actions solo se corran los unit tests en las actions. Se podría agregar para que también se corran los test de specflow pero para esto era necesario correr el .jar o .exe de la aplicación y se decidió no complicarse con esto para los test automáticos en github. Estos test se corren al igual que antes cada vez que se pushea un commit o sino al crear una pull request. Se corren los test en los 3 sistemas operativos, windows, linux y ios.

Estos test nos permiten identificar si algún cambio rompió una funcionalidad pasada. Lo malo es que dichos test solo chequean que las funciones del backend hagan lo que deberían hacer, no se testea ni el front end, ni test de integración. Para esto lo que hicimos fueron los test de selenium que testean la integración desde el front hasta el final. Estos test de selenium no los agregamos a la pipeline de github action pero entendemos que podía ser una buena mejora para agregarle al sistema en un futuro.

Test exploratorios basado en KANBAN

Los Test Exploratorios basados en Kanban son una técnica de prueba de software que fusiona los principios de los Test Exploratorios con la metodología Kanban. En este enfoque, se utiliza un tablero visual para gestionar el flujo de trabajo de las pruebas exploratorias, permitiendo una visualización clara de su progreso.

Se aplican límites de trabajo en curso (WIP) para evitar la sobrecarga y se priorizan las pruebas en función de las necesidades actuales del proyecto. Esto proporciona flexibilidad y la capacidad de adaptarse a los cambios de manera continua.

La técnica fomenta la creatividad y la exploración activa del software en busca de problemas no documentados previamente.

Uso de Selenium

Selenium es una suite de herramientas de código abierto ampliamente utilizada para la automatización de pruebas en aplicaciones web y permite automatizar interacciones con navegadores web.

La combinación de Selenium con Test Exploratorios basados en Kanban puede mejorar la eficiencia de las pruebas de software al permitir la automatización de pruebas en aplicaciones web, acelerando el proceso de pruebas y manteniendo la calidad del software.

Ver el archivo utilizado aquí.

Casos de prueba

Bug 3: Rechazo Compras

Si en una compra no se encuentra el medicamento en la farmacia indicada esta queda como pendiente ya que no se permite aceptarla o rechazarla, debería permitir rechazarla o que el sistema la rechace de forma automática

Solución: Actualmente se puede rechazar una compra de un medicamento borrado. También se cambió el mensaje de error si se trata de aceptar la compra de un medicamento que fue borrado

- loguearse como empleado, asegurarse de haber comprado algún medicamento y borrado
- Mostrar que no se puede aceptar la compra pero si rechazarla debido a que el medicamento fue borrado

Bug 4: Exportación de medicamentos

Al exportar los medicamentos de una farmacia se encuentran los medicamentos de la farmacia, al igual de aquellos que fueron previamente eliminados de la misma

- Asegurarse de que estoy logueado como empleado de una farmacia con medicamentos eliminados, exportar los medicamentos y verificar que los medicamentos eliminados no son mostrados

Bug 6: Código de invitaciones Administrador

Al hacer el update de una invitación, se permite cambiar el código a un código de menos de 6 dígitos

- Ingresar como administrador, intentar modificar una
- Editar el código a mano, colocando un número de menos de 6 dígitos (ejemplo 22)
- Presionar el botón de "update invitation"
- No te deja y muestra error en pantalla
- Probar lo mismo con más de 6 dígitos
- Probar con 6 dígitos exacto, ahi si se actualiza correctamente

User Story 1: Alta Producto

Scenario: Create product with correct data

- **Given** the code "12347"
- And the name "Shampoo Sedal 200 ml"
- And the description "Dale vida a tu pelo con el nuevo shampoo Sedal"
- And the price 80,00
- And the pharmacy with id 1
- **When** press the create button
- **Then** the product should be registered correctly with code 200

Scenario: Create product with wrong code length

- **Given** the code "123478"
- And the name "Shampoo Sedal 200 ml"
- And the description "Dale vida a tu pelo con el nuevo shampoo Sedal"
- And the price 80,00
- And the pharmacy with id 1
- **When** press the create button
- **Then** the product should not be registered with code 400

Scenario: Create product with wrong code, no digits

- **Given** the code "1234A"
- And the name "Shampoo Sedal 200 ml"
- And the description "Dale vida a tu pelo con el nuevo shampoo Sedal"
- And the price 80,00
- And the pharmacy with id 1
- **When** press the create button
- **Then** the product should not be registered with code 400

Scenario: Create product with wrong name, no alphanumeric

- **Given** the code "12345"
- And the name "Shampoo Sedal 200 ml%%$"
- And the description "Dale vida a tu pelo con el nuevo shampoo Sedal"
- And the price 80,00
- And the pharmacy with id 1
- **When** press the create button
- **Then** the product should not be registered with code 400

Scenario: Create product with wrong name, length longer than 30 characters

- **Given** the code "12345"
- And the name "123456789 123456789 123456789 0”
- And the description "Dale vida a tu pelo con el nuevo shampoo Sedal"
- And the price 80,00
- And the pharmacy with id 1
- **When** press the create button
- **Then** the product should not be registered with code 400

Scenario: Create product with wrong description, non alphanumeric

- **Given** the code "12345"
- And the name "Shampoo Sedal 200 ml"
- And the description "Dale vida a tu pelo con el nuevo shampoo Sedal &"

- And the price 80,00
- And the pharmacy with id 1
- **When** press the create button
- **Then** the product should not be registered with code 400

Scenario: Create product with wrong description, length longer than 70 characters

- **Given** the code "12345"
- And the name "Shampoo Sedal 200 ml"
- And the description "123456789 123456789 123456789 123456789 123456789 123456789 123456789 1"
- And the price 80,00
- And the pharmacy with id 1
- **When** press the create button
- **Then** the product should not be registered with code 400

User Story 2: Baja Producto

Scenario: Delete product correctly

- **Given** the product registered in my pharmacy with id
- **When** press the delete button for that product
- **Then** the product should be deleted correctly

User Story 3: Modificación de producto

Scenario: Successful Modification

- **Given** the product with id 2
- And code "12349"
- And name "Antiperspirant Roll"
- And description "cream for underarm care"
- And price as 70.00
- **When** I press the "Modify" button for that product, I should be able to modify the name to "Antitranspirante RollOn" and the description to "With moisturizing cream for underarm care"
- **Then** the product should be edited correctly with code 200

Scenario: Modification with Incorrect Code

- **Given** the product with id 2
- And code "12349"
- And name "Antitranspirante RollOn"
- And description "With moisturizing cream for underarm care"
- And price as 70.00

- **When** I press the "Modify" button for that product, I should be able to modify the code to "123"
- **Then** the product should be edited correctly with code 400

Scenario: Modification with Incorrect Code (Different Product)

- **Given** the product with id 2
- And code "12349"
- And name "Antitranspirante RollOn"
- And description "With moisturizing cream for underarm care"
- And price as 70.00
- **When** I press the "Modify" button for that product, I should be able to modify the code to "12347"
- **Then** the product should be edited correctly with code 500

Scenario: Modification with Incorrect Name

- **Given** the product with id 2
- And code "12349"
- And name "Antitranspirante RollOn"
- And description "With moisturizing cream for underarm care"
- And price as 70.00
- **When** I press the "Modify" button for that product, I should be able to modify the name to "The best Antitranspirante RollOn" and the description to "With moisturizing cream for underarm care"
- **Then** the product should be edited correctly with code 400

Scenario: Modification with Incorrect Description

- **Given** the product with id 2
- And code "12349"
- And name "Antitranspirante RollOn"
- And description "With moisturizing cream for underarm care"
- And price as 70.00
- **When** I press the "Modify" button for that product, I should be able to modify the name to "Antitranspirante RollOn" and the description to "Dove Original Roll On antiperspirant with moisturizing cream for underarm care"
- **Then** the product should be edited correctly with code 400

User Story 4: Compra Producto

Scenario: Compra correcta

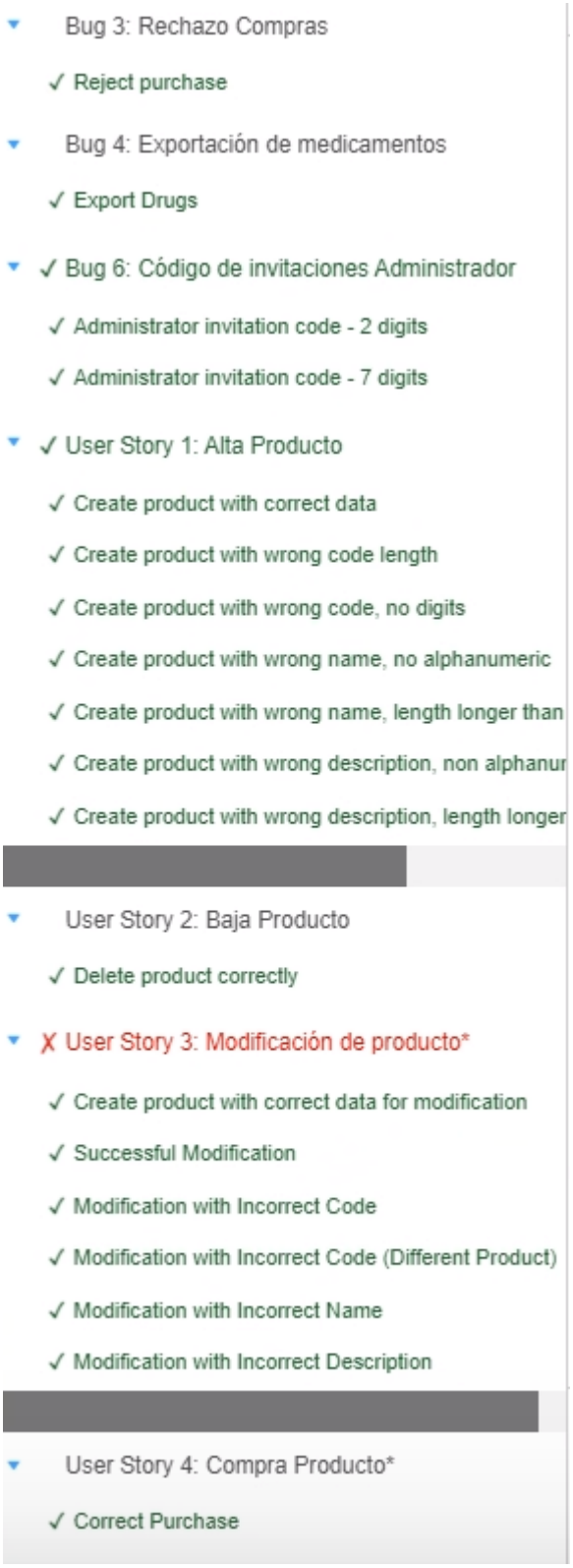
- **Given** a product of a pharmacy
- And a positive quantity greater than 0 of it, for example 5

- **When** I select add to cart
- **Then** the product is added to the purchase with that quantity

Resultados obtenidos

Para la confección de los tests en Selenium se crea un proyecto que llamamos Entrega 4 donde creamos varias test Suites uno para cada caso de prueba.

Luego de que todas las pruebas corrieron este es el resultado:



Video de todos los test siendo ejecutados (se pueden ejecutar las testSuite en ese orden):

- Crear producto
- Modificar producto
- Eliminar producto

- Intentar aprobar una compra con producto eliminado
- Exportar medicamentos
- Modificar código de una invitación
- Hacer compra

Estructura y Mantenimiento del repositorio

Creación del repositorio

El repositorio fue creado en la plataforma github. Esto nos permite tener un control de versiones sobre el proyecto, lo cual significa poseer control de las modificaciones realizadas sobre el proyecto.

Para hacer esto, primero se creó un equipo en la plataforma, donde se encontraran presentes cada uno de los mails de los integrantes. Después se agregó a la organización de la materia un nuevo repositorio agregándolo como perteneciente al equipo creado con anterioridad.

Después de eso, se agregó el contenido del proyecto a analizar y modificar al repositorio. Este fue encontrado en la página de aulas de la materia. Después de haberlo descargado, se usó git bash para incorporarlo a la rama “main” del repositorio.

Cuando el proyecto DevOps estuvo incorporado, se decidió agregar una descripción resumida de los objetivos de este obligatorio a su vez del mecanismo de evaluación.

Mantenimiento del repositorio

Para realizar el mantenimiento del repositorio se decidió utilizar las herramientas de Google Hojas de Cálculos y Documentos. A su vez, todos los archivos creados con estas plataformas, serán guardados en una carpeta compartida de Drive. Podrá acceder a esta carpeta aquí.

Uso de Documentos de Google

Esta plataforma nos permitirá crear documentos de informes en formato pdf. Estos informes serán los que poseerán toda la información de análisis de cada entrega. Se creará un documento para cada entrega y luego se resumen todos para la entrega final.

Uso de Hojas de Cálculos de Google

Esta plataforma nos permitirá crear libros de trabajo donde registramos el esfuerzo con tareas y subtareas por entrega. Esto significa que se creará un libro de trabajo para cada entrega.

Métricas

Podrá ver el seguimiento de esfuerzo y métricas realizadas aquí.

Cycle Time, Lead Time, Flow Efficiency y Throughput

Se pide el análisis de métricas sobre los bugs resueltos y user stories.

Bugs

Se utilizan los bugs de la entrega 2.

Bugs (entrega2)

Nro sub tarea	Tipo	Descripción	Horas/Esfuerzo persona	Responsable	Fecha In Todo
6	Bug	Fix Bug 3	1:30:00	Andres	20/9/2023
7	Bug	Fix Bug 4	1:30:00	Sofia	20/9/2023
8	Bug	Fix Bug 6	1:00:00	Romina	20/9/2023

Nro sub tarea	Cycle Time (dias)	Lead Time (dias)	Flow efficiency	Throughput	Duración de la entrega (dias)
6	0.00	3.00	0	0.33	9
7	1.00	3.00	0.3333333333		
8	0.00	3.00	0		
	0.33	3.00	0.1111111111		

User Stories

Se utilizan las user stories de la entrega 3.

User Story (entrega 3)

Nro sub tarea	Tipo	Descripción	Horas/Esfuerzo persona	Responsable	Sprint Backlog
6	User Story	User Story 1 - Backend y Frontend	15:00:00	Todos*	11/10/2023
7	User Story	User Story 2 - Backend y Frontend	3:30:00	Sofia	11/10/2023
8	User Story	User Story 3 - Backend y Frontend	10:00:00	Romina	11/10/2023
9	User Story	User Story 4 - Backend y Frontend	5:00:00	Andy	11/10/2023

Nro sub tarea	Cycle Time (dias)	Lead Time (dias)	Flow efficiency	Throughput	Duración de la entrega (dias)
6	8.00	11.00	0.7272727273	0.1666666667	24
7	9.00	12.00	0.75		
8	8.00	11.00	0.7272727273		
9	9.00	12.00	0.75		
	8.50	11.50	0.7386363636		

Comparativas

Cycle Time: Los Bugs tienen un Cycle Time mucho más corto en comparación con las User Stories. Los Bugs se resuelven en cuestión de horas (0.33 días), mientras que las User Stories toman varios días (8.50-9 días).

Lead Time: El Lead Time de las User Stories es significativamente más largo que el de los Bugs, con una diferencia de alrededor de 8 días. Esto va relacionado con lo dicho en la parte de cycle time.

Flow Efficiency: Los Bugs tienen un Flow Efficiency mucho más bajo (11.11%) en comparación con las User Stories (73.87%). Un Flow Efficiency más alto indica que las User stories se están moviendo de manera más eficiente a través del proceso. Esta métrica nos indica que en proporción los bugs pasan más tiempo en el ToDo que las user stories

Throughput: La Throughput de los Bug Fixes es más alta en comparación con las User Stories, esto significa que se demora menos en entregar el arreglo de un bug, que la implementación de una nueva user story. Las User Stories toman alrededor de 11.50 días para completarse, en comparación con los 3 días de los arreglos de Bugs. Esto es porque el esfuerzo que implica corregir un bug es menor que el de implementar una nueva user story. Esto se puede ver en los registros de esfuerzos de las correcciones de bugs que fueron de aproximadamente 1.5 horas persona en cambio las user stories aunque variaban mucha oscilaban entre 8.5 horas persona.

Una observación relevante a hacer es que a pesar de que en kanban se trata de que todas las user stories requieran aproximadamente el mismo esfuerzo, como éramos un equipo nuevo y era la primera vez que usamos specflow, entonces en la primera user story que hicimos (user story 1) nos tomó bastante más tiempo y esfuerzo debido a la curva de aprendizaje que cualquier nueva tecnología conlleva. Esta es la razón por la cual las user stories tomaron valores de esfuerzos muy distintos entre ellas, ya que como se puede ver en la tabla de las user stories de arriba, se observa que hay user stories que tomaron 15 horas persona (la primera user story que hicimos) y otras como el delete product (user story 2) que eran muy parecido al create o al modify y tomó 3.5 horas persona. Si analizamos lo mismo para los bugs se puede ver que ahí si las cards eran de tamaños similares.

Conclusión

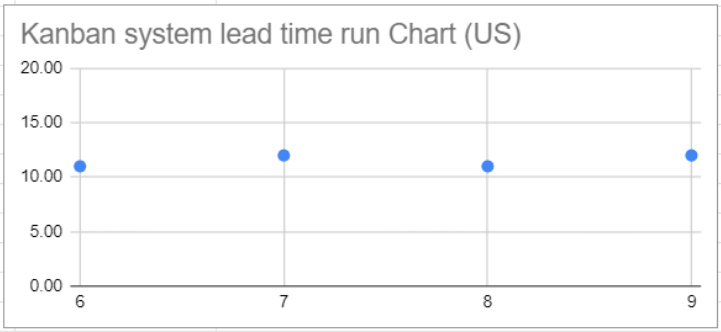
Como conclusión, debido a la gran diferencia de esfuerzo entre los Bugs y las User Stories sus métricas no son comparables para la obtención de un resultado concluyente respecto a la performance del equipo. A su vez, el periodo de realización de las mismas también varía lo que afirma que estas no son comparables.

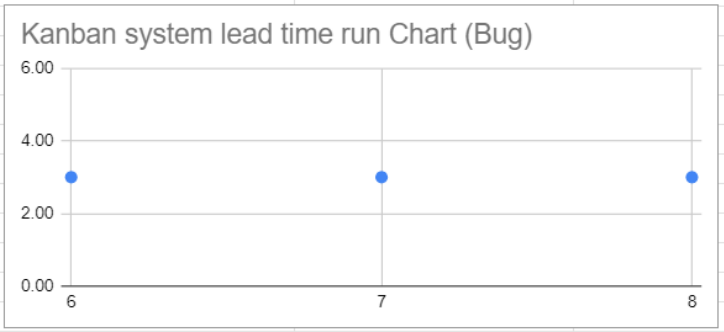
Kanban system lead time chart

Kanban system lead time run Chart

Nro User Story		Dia de ingreso de tarea	Dia de finalización	Tiempo
6		11/10/2023	22/10/2023	11.00
7		11/10/2023	23/10/2023	12.00
8		11/10/2023	22/10/2023	11.00
9		11/10/2023	23/10/2023	12.00
				11.50

Nro Bug		Dia de ingreso de tarea	Dia de finalización	Tiempo
6		20/9/2023	23/9/2023	3.00
7		20/9/2023	23/9/2023	3.00
8		20/9/2023	23/9/2023	3.00
				3.00





En esta gráfica se puede observar el tiempo que llevó cada user story y bug desde que se definió hasta estar completa. Se puede decir que el promedio de resolución de los bugs es de 3 días a diferencia de las user stories que son entre 11 y 12 días. Por lo tanto, las user stories requieren más esfuerzo en ser resueltas que los bugs.

Registro del trabajo y Esfuerzo

Nro sub tarea	Tipo	Descripción	Horas/Esfuerzo persona	Responsable
1	Tarea	Actualizacion de la definicion de KANBAN (doc)	0:30:00	Sofia
2	Tarea	Proceso de ingeneiria en conjunto	1:00:00	Sofia
3	Tarea	Text exploratorios basado en KANBAN (doc)	6:30:00	Todos
4	Tarea	Cuarta version del tablero y actualizacion	0:30:00	Todos
5	Tarea	Configuracion del pipeline (doc y gitAction)	1:00:00	Andy
6	Tarea	Explicacion del uso del tablero (doc)	0:30:00	Sofia
7	Tarea	Asignacion de roles	0:30:00	Sofia
8	Tarea	Reparacion de errores de la entrega anterior	5:00:00	Todos
9	Tarea	Metricas de Us y Bugs	1:30:00	Romi
			17:00:00	

Para esta entrega contamos con 9 tareas y el total del registro de esfuerzo fue de 17 horas. El registro de esfuerzo de los bugs y user stories se encuentran en los informes de las entregas anteriores y en las tablas insertadas anteriormente.

Análisis del trabajo

Registro de las daily's

24/10/2023: Mensaje de texto

P1: ¿Alguien está trabajando en algo que no está en el tablero?

- No

P2: Como equipo , ¿qué estamos buscando finalizar?

- Estamos buscando organizar lo que se debe hacer

P3: ¿Existen cuellos de botella u otros impedimentos al flujo de trabajo?

- No

26/10/2023: Mensaje de texto

P1: ¿Alguien está trabajando en algo que no está en el tablero?

- No

P2: Como equipo , ¿qué estamos buscando finalizar?

- Estamos buscando distribuir tareas y tener certeza de cómo hacer las cosas correctamente

P3: ¿Existen cuellos de botella u otros impedimentos al flujo de trabajo?

- No

28/10/2023: Llamada de teams

P1: ¿Alguien está trabajando en algo que no está en el tablero?

- No

P2: Como equipo , ¿qué estamos buscando finalizar?

- Estamos buscando corregir errores de la entrega pasada y empezar con Selenium

P3: ¿Existen cuellos de botella u otros impedimentos al flujo de trabajo?

- Algunos problemas con SpecFlow que fueron solucionados

Registro de la Review

Se realizó un único video verificando todas las funcionalidades. Podrá verlo aquí.

Registro de retrospectiva

Este registro de retrospectiva se basa en el método DAKI y se utiliza para documentar los aprendizajes y resultados de la retrospectiva.

También se creó un tablero en retrometro para acompañar la retrospectiva.

Formato de la retrospectiva:

- 5 minutos iniciales para responder todas las secciones. Esto se hace de forma anónima.
- Abrir las notas para que todos puedan ver lo que se escribió.
- Dialogar sobre cada sección y discutir para llegar a una respuesta conjunta.
- Luego, para finalizar, se eligiran las ideas más importantes de las planteadas para cumplir en la próxima entrega.

Podrá ver el tablero aquí y el video de la retrospectiva aquí.

Drop

En esta sección, identificamos las cosas que hicimos mal en esta entrega y que deberíamos dejar de hacer en el futuro.

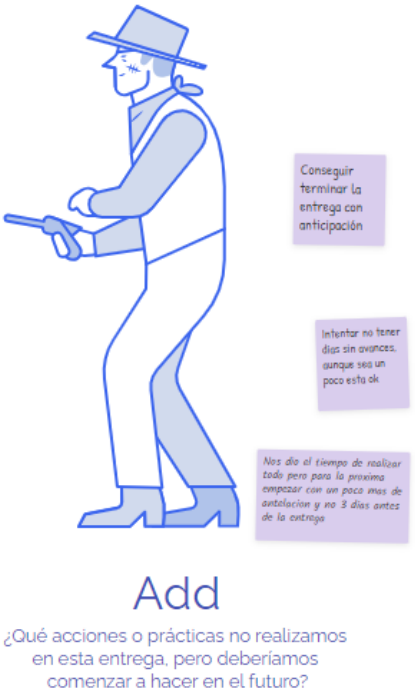
Pregunta: ¿Qué cosas hicimos mal en esta entrega que deberíamos dejar de hacer en el futuro?

La idea más importante destacada fue: “Arrancar antes la entrega”. A pesar de que es algo que hemos mencionado en entregas anteriores creemos que de igual manera en este aspecto hemos mejorado.

Add

Aquí, enumeramos las acciones o prácticas que no realizamos en esta entrega, pero que deberíamos comenzar a hacer en el futuro.

Pregunta: ¿Qué acciones o prácticas no realizamos en esta entrega, pero deberíamos comenzar a hacer en el futuro?



La idea más importante destacada fue: “No tener días sin avances”

Keep

En esta sección, destacamos las cosas que hicimos bien en esta entrega y que deberíamos seguir haciendo en el futuro.

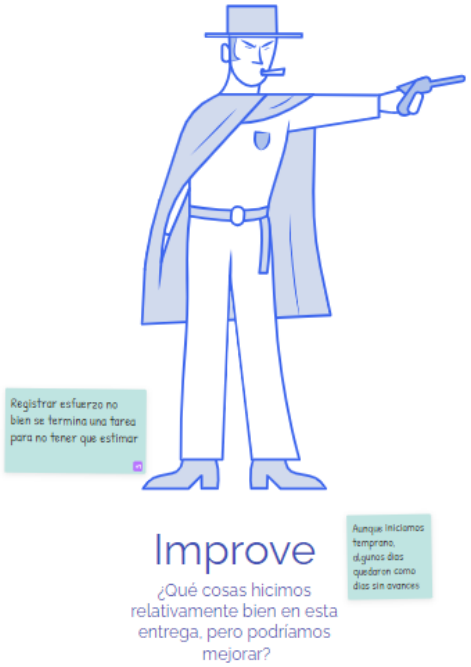
Pregunta: ¿Qué cosas hicimos bien en esta entrega que deberíamos seguir haciendo en el futuro?

La idea más importante destacada fue: “Buena organización, documentación y comunicación”. Esto se repite respecto a retrospectivas anteriores.

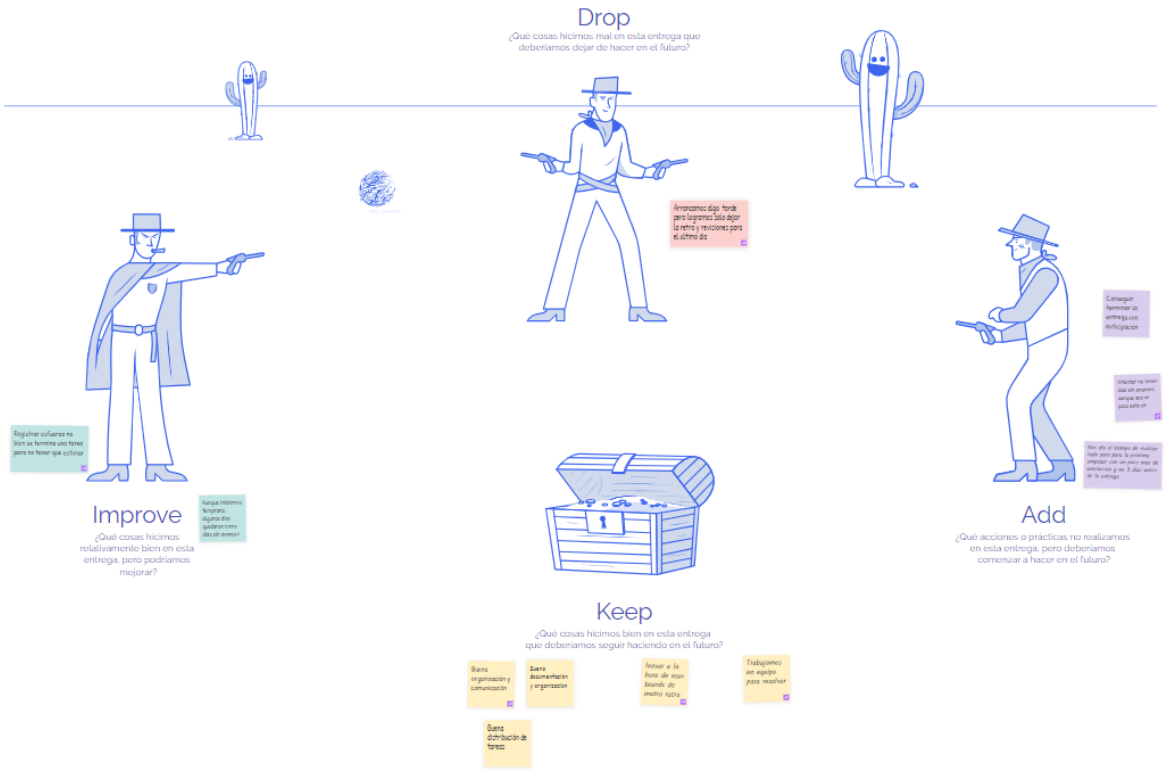
Improve

Finalmente, identificamos las cosas que hicimos relativamente bien en esta entrega, pero que podríamos mejorar.

Pregunta: ¿Qué cosas hicimos relativamente bien en esta entrega, pero podríamos mejorar?



La idea más importante destacada fue: “Registrar esfuerzo no bien se termine la tarea y no estimar”



Registro de avance

Resultados obtenidos en el período

El objetivo de esta entrega fue verificar el correcto funcionamiento del sistema utilizando Selenium con BDD. No se implementaron nuevas funcionalidades, en su lugar, se trabajó con las funcionalidades previamente implementadas y se corrigieron los errores conocidos.

Los resultados obtenidos en este período incluyen:

- Implementación de pruebas de Selenium utilizando BDD para cubrir las funcionalidades existentes.
- Verificación exitosa del funcionamiento de las funcionalidades previamente implementadas.
- Obtención de métricas que proporcionaron una representación de la cobertura de las pruebas y la calidad del sistema.

Dificultades encontradas y formas de solución

Durante esta fase, se enfrentaron dificultades en la creación de las pruebas de Selenium con BDD. Se encontraron desafíos técnicos en la creación de pruebas de Selenium utilizando BDD, lo que resultó en retrasos en el proceso.

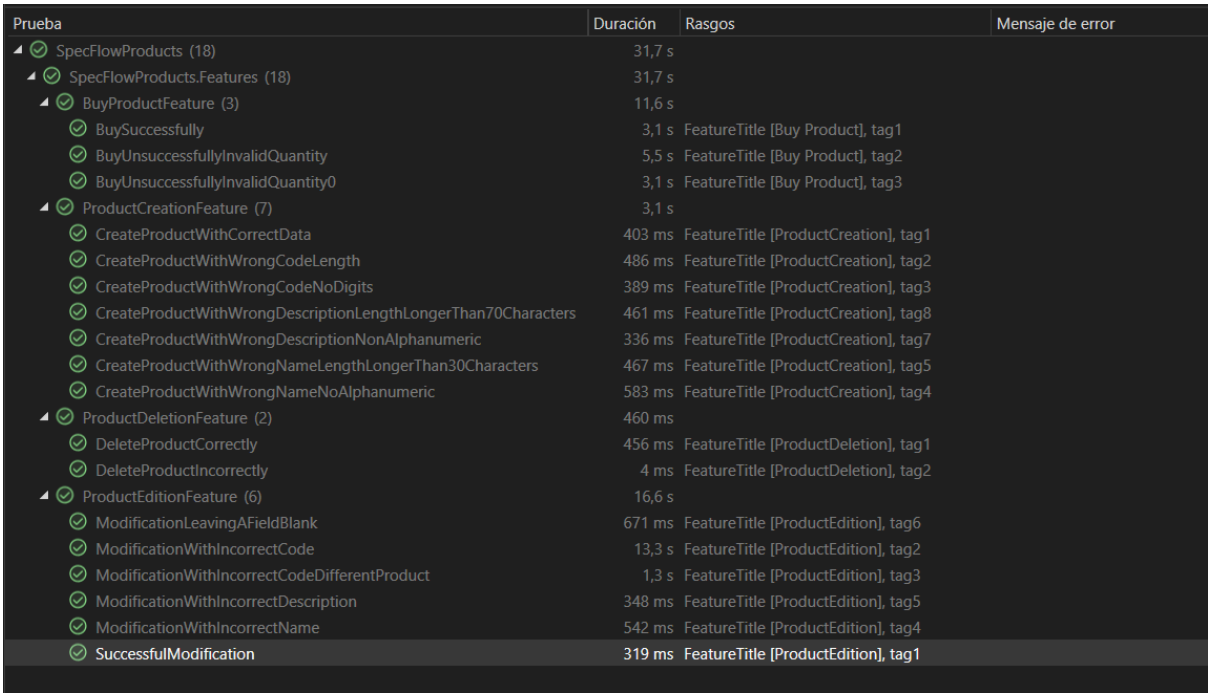
Lecciones aprendidas y mejoras en el proceso

Se adquirió experiencia en la utilización de Selenium con BDD para realizar pruebas automatizadas que verifican el comportamiento del sistema. También se comprendió la importancia de la automatización de pruebas en el proceso de desarrollo para garantizar la calidad y el funcionamiento correcto del sistema.

Con respecto a la entrega anterior

Con respecto a la entrega anterior, en esta hemos corregido varios errores observados en la documentación, código y en el registro de esfuerzo. Entre estos fueron corregidos los errores de Specflow, los registros de esfuerzo de la User Story 1 y 4, y errores varios de arrastre o ortografía de entregas anteriores.

Evidencia de que los tests fueron arreglados:



Prueba	Duración	Rasgos	Mensaje de error
SpecFlowProducts (18)	31,7 s		
SpecFlowProducts.Features (18)	31,7 s		
BuyProductFeature (3)	11,6 s		
BuySuccessfully	3,1 s	FeatureTitle [Buy Product], tag1	
BuyUnsuccessfullyInvalidQuantity	5,5 s	FeatureTitle [Buy Product], tag2	
BuyUnsuccessfullyInvalidQuantity0	3,1 s	FeatureTitle [Buy Product], tag3	
ProductCreationFeature (7)	3,1 s		
CreateProductWithCorrectData	403 ms	FeatureTitle [ProductCreation], tag1	
CreateProductWithWrongCodeLength	486 ms	FeatureTitle [ProductCreation], tag2	
CreateProductWithWrongCodeNoDigits	389 ms	FeatureTitle [ProductCreation], tag3	
CreateProductWithWrongDescriptionLengthLongerThan70Characters	461 ms	FeatureTitle [ProductCreation], tag8	
CreateProductWithWrongDescriptionNonAlphanumeric	336 ms	FeatureTitle [ProductCreation], tag7	
CreateProductWithWrongNameLengthLongerThan30Characters	467 ms	FeatureTitle [ProductCreation], tag5	
CreateProductWithWrongNameNoAlphanumeric	583 ms	FeatureTitle [ProductCreation], tag4	
ProductDeletionFeature (2)	460 ms		
DeleteProductCorrectly	456 ms	FeatureTitle [ProductDeletion], tag1	
DeleteProductIncorrectly	4 ms	FeatureTitle [ProductDeletion], tag2	
ProductEditionFeature (6)	16,6 s		
ModificationLeavingAFieldBlank	671 ms	FeatureTitle [ProductEdition], tag6	
ModificationWithIncorrectCode	13,3 s	FeatureTitle [ProductEdition], tag2	
ModificationWithIncorrectCodeDifferentProduct	1,3 s	FeatureTitle [ProductEdition], tag3	
ModificationWithIncorrectDescription	348 ms	FeatureTitle [ProductEdition], tag5	
ModificationWithIncorrectName	542 ms	FeatureTitle [ProductEdition], tag4	
SuccessfulModification	319 ms	FeatureTitle [ProductEdition], tag1	

Para que funcionen los test de modificar se debe poseer en la base de datos un objeto con id 1 y otro con el código 45678.