



Universidad ORT Uruguay

Facultad de Ingeniería

Informe de avance 3

Ingeniería de Software Ágil 2

[Repositorio en Github](#)

Los distintos documentos que forman parte de la entrega se encuentran dentro del repositorio Documentación/Entrega 3

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	3
Uso de BDD	3
One Piece Flow	3
Designación de roles	4
Tablero con BDD	4
Configuración del pipeline	6
Requirement definition	7
Evidencia de ejecución de casos de prueba	11
Alta producto	11
Ejecución de las pruebas	12
Baja producto	12
Ejecución de las pruebas	12
Modificación producto	13
Ejecución de las pruebas	13
Compra producto	14
Ejecución de las pruebas	14
Review	15
Retrospectiva	15
Conclusiones	16

Definición del proceso de ingeniería en el contexto de KANBAN

Uso de BDD

Para esta iteración trabajo en base a BDD (Behaviour Driven Development), BDD consta en la definición de los criterios de aceptación previa a la implementación de la funcionalidad.

Dadas las funcionalidades brindadas por los docentes, comenzamos por definir las User Stories para dichos requerimientos nuevos, utilizando el formato “Como-Quiero-Para”. Una vez creadas las user stories se definieron los criterios de aceptación mediante distintos escenarios con el formato “Dado-Cuando-Entonces”. Los criterios de aceptación permiten establecer las condiciones bajo las cuales se considera que la funcionalidad funciona correctamente y de acuerdo a las necesidades de los usuarios definidas por el product owner.

Una vez definidos se comienza con la etapa de implementación de pruebas y desarrollo. Dado que no se aplicaran pruebas al frontend se comenzó por el desarrollo del mismo. Una vez listo el frontend, se continuó por definir las pruebas usando la herramienta SpecFlow. Se creó un archivo .feature donde se escriben las user stories y los criterios de aceptación y un archivo Step Definitions donde se implementa el código que corresponde a cada prueba.

Luego de creados los test se comenzó con el desarrollo del código del backend utilizando los test creados previamente para validar su correcto funcionamiento.

Finalmente, se llevó a cabo una revisión conjunta con el Product Owner para validar la funcionalidad, asegurando que cumple con los requisitos establecidos en las historias de usuario y los criterios de aceptación. Este enfoque permite asegurarnos que el desarrollo de funcionalidades se realiza acorde a la necesidades de los usuarios.

One Piece Flow

Al utilizar una herramienta nueva como lo es *Specflow*, decidimos adoptar la metodología *One Piece Flow*. Debido a ello, se desarrolló una única funcionalidad a la vez. De esta forma logramos mejorar la eficiencia mediante la eliminación de cuellos de botella y tiempos de espera por desconocimientos de las herramientas ya que nos encontrábamos los tres desarrolladores trabajando en conjunto sobre la misma pieza de código.

Comenzamos implementando la funcionalidad en el frontend, para continuar traduciendo los criterios de aceptación para ser ingresados a Specflow. Finalmente, se

desarrollo el backend de forma tal que los tests creados con Specflow pudieran ser ejecutados.

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 Agustina Lamanna. 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 Sebastián Daners. 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.

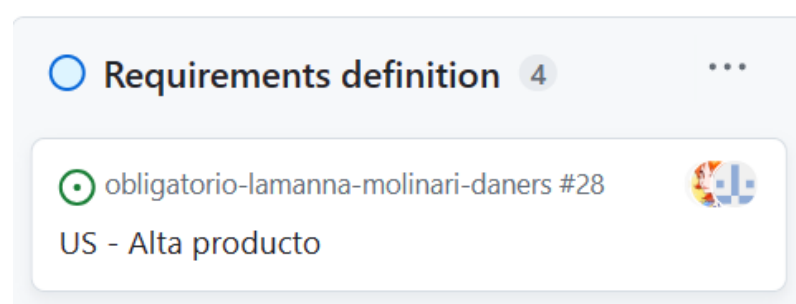
Asimismo, los tres integrantes adoptaron roles de desarrolladores y testers.

Tablero con BDD

El tablero se organizó de forma tal que pudiera considerar tanto las tareas como las actividades relacionadas a BDD.

Para el caso de las tareas, se definieron las columnas *To do* en donde se pusieron todas las tareas relacionadas con esta entrega que no estaban relacionadas a modificaciones de código. Por ejemplo, se agregaron allí las tarjetas de *Planificación* en donde se planificó las actividades a realizar durante esta entrega, así como la metodología a utilizar y la repartición de roles entre los integrantes. Luego se definió la columna *In progress* en donde se posicionaban las tarjetas que corresponden a tareas que se estaban realizando. Finalmente, en la columna *Done* se encuentran todas aquellas tareas finalizadas.

En el caso de las actividades relacionadas a BDD, se decidió crear una columna para cada una de las actividades del ciclo de BDD dentro del *Plan, Do, Check*. En la columna *Requirement definition* se escribieron las narrativas y los casos de uso siguiendo los pasos CCC (Card specification, Conversation y Confirmation). Allí se generan las user stories y se ingresaron la narrativa y los criterios de aceptación dentro de un *issue*. En la siguiente imagen se puede observar cómo se fueron posicionando las tarjetas correspondientes a cada US, ejemplificado con la tarjeta correspondiente al alta de producto.



En la siguiente columna, *Test cases implementation* se pasaron los criterios de aceptación escritos en español a formato *cucumber* para poder ser ingresados a *Specflow*. Para todos los issues en este punto se decidió realizar una separación de la user story en **frontend** y **backend**.

Luego en la columna *App Implementation & Automation testing* se juntaron las actividades de BDD en una misma columna para que quedara un tablero con actividades similares a las de TDD. En este paso se realizó la codificación e implementación de la aplicación, así como la escritura y ejecución de los tests implementados en el paso anterior de *Test cases implementation*. En resumen, se escribieron los códigos en angular y C#. En la columna *Refactoring* se posicionan aquellas tarjetas que al ejecutar los tests de *Automation testing* de la parte anterior, quedó algún test con estado/color rojo. En esos casos, en los que se encontraron bugs/cosas que no funcionaban correctamente, se comienza el ciclo nuevamente a partir de *Requirements definition*.

Finalmente, en la columna *Integration testing* corresponde a los test de integración. Allí se verifica que el frontend y backend estén integrados correctamente y que todo funcione según lo esperado. Esto nos permite identificar y corregir problemas que puedan ocurrir en la integración, antes de que el producto sea considerado como finalizado.

En la siguiente tabla se encuentran resumidos los roles, artefactos y métodos involucrados en cada una de las ceremonias:

Ceremonias	Roles	Artefactos	Método
<i>Plan</i>	Equipo	Plan	Estimación / SP por cada US
<i>Requirement definition</i>	Usuario + equipo	Requerimientos	CCC + US
<i>Test Cases Implementation</i>	Equipo	Test cases	Gherkin
<i>Application implementation</i>	Equipo	Código	OOP (Object Oriented Programming)
<i>Automation testing</i>	Equipo	Test Report	Diseño de scripts de prueba, desarrollarlos y ejecutarlos

<i>Refactoring</i>	Equipo	Code	Comenzar el proceso nuevamente
<i>Integration test</i>	Equipo + tester	Test Result	Ejecución del sistema como usuario
<i>Review</i>	Equipo + PO	Requerimientos implementados	Muestra al PO de los requerimientos implementados
<i>Retrospectiva</i>	Equipo + SM	Listado de ideas para implementar y cosas a continuar haciendo	DAKI

Configuración del pipeline

En esta oportunidad no se realizaron cambios al pipeline en Github Actions.

Se tomó la decisión de realizar las pruebas correspondientes a la metodología BDD mediante el uso de la herramienta *Specflow* de forma que las pruebas interactúan directamente con los controladores y repositorios/base de datos. Por lo tanto, esto no nos permitió automatizar su ejecución con el pipeline ya que se accede directamente a la base de datos. Si se hubiera codificado las pruebas de Specflow de forma tal que se mockeara el comportamiento de los controladores y de la base de datos, dichas pruebas se podrían ejecutar de forma automática desde el pipeline definido en el github actions.

Se mantuvo al igual que la entrega anterior, la automatización mediante el pipeline de la ejecución de las pruebas realizadas con la metodología TDD.

A continuación dejamos las líneas de código que se deberían agregar al pipeline definido, si funcionara:

```
- name: Install SpecFlow.Tools.MsBuild.Generation
  run: dotnet tool install --global SpecFlow.Tools.MsBuild.Generation --version 3.9.74

- name: Generate SpecFlow files
  run: dotnet msbuild /t:SpecFlowGenerateAll C:\Users\Administrador\Desktop\obligatorio-lamanna-molinari-daners\Código\Backend\PharmaGo.sln

- name: Run SpecFlow Tests
  run: dotnet test C:\Users\Administrador\Desktop\obligatorio-lamanna-molinari-daners\Código\Backend\PharmaGo.SpecFlow\PharmaGo.SpecFlow.csproj
```

Requirement definition

En la etapa de *requirement definition* se definió para cada una de las nuevas funcionalidades requeridas, las siguientes narrativas y criterios de aceptación (con sus correspondientes escenarios). Los mismos cuentan además con un título que identifica la funcionalidad a la que se refieren.

Para la escritura de las user stories se siguieron los siguientes pasos:

¿Quién lo escribe?		¿Cuándo se escribe?
PO	Nombre de la US Narrativa	En el momento que se realiza la card specification
Team	Criterios de aceptación	Luego del paso de conversation (relevamiento) el equipo se dispone a escribir los criterios de aceptación
Team	Tamaño	En la <i>planning meeting</i>

Para este caso, al encontrarnos en un proceso de aprendizaje consideramos que sería más beneficioso realizar todos los pasos en conjunto en lugar de que el PO determinara el nombre y la narrativa de la user story por su cuenta.

A continuación se detallan las user stories creadas:

Título: Alta producto

Narrativa:

Como empleado

Quiero dar de alta un nuevo producto

Para que el producto esté en el sistema

Criterios de aceptación:

Escenario 1 - alta de producto satisfactoria

Dado que al ingresar nombre, descripción y precio válidos en el formulario

Cuando presiono el botón *Añadir*

Entonces el sistema muestra un mensaje de éxito

Escenario 2 - nombre de producto con largo excedido

Dado que al ingresar un nombre con más de 30 caracteres en el formulario

Cuando presiono el botón *Añadir*

Entonces el sistema muestra un mensaje de error "Nombre excede el largo permitido"

Escenario 3 - campo nombre de producto sin completar

Dado que se deja sin completar el nombre del producto en el formulario

Cuando presiono el botón *Añadir*

Entonces el sistema muestra un mensaje de error “Debe completar todos los datos”

Escenario 4 - descripción con largo excedido

Dado el ingreso de una descripción con más de 70 caracteres en el formulario

Cuando presiono el botón *Añadir*

Entonces el sistema muestra un mensaje de error “Descripción excede el largo permitido”

Escenario 5 - campo descripción sin completar

Dado el ingreso de una descripción vacía en el formulario

Cuando presiono el botón *Añadir*

Entonces el sistema muestra un mensaje de error “Debe completar todos los datos”

Escenario 6 - precio menor a cero

Dado el ingreso de un precio menor a cero en el formulario

Cuando presiono el botón *Añadir*

Entonces el sistema muestra un mensaje de error “El Precio no puede ser negativo”

Escenario 7 - campo precio sin completar

Dado el ingreso de un precio vacío en el formulario

Cuando presiono el botón *Añadir*

Entonces el sistema muestra un mensaje de error “Debe completar todos los datos”

Título: Baja Producto

Narrativa:

Como empleado

Quiero dar de baja un producto existente

Para que el producto ya no este en el sistema

Criterios de aceptación:

Escenario 1 - baja de producto satisfactoria

Dado la selección de un producto, de la lista dada

Cuando presiono el botón *Eliminar*

Entonces el sistema muestra un mensaje, “Producto eliminado correctamente”

Título: Modificación de un Producto

Narrativa:

Como empleado

Quiero modificar los datos de un producto existente

Para que estos datos de producto se modifiquen en el sistema

Criterios de aceptación:

Escenario 1 - modificación de producto satisfactoria

Dado que al ingresar nombre y/o descripción y/o precio válidos en el formulario

Cuando presiono el botón *Modificar*

Entonces el sistema muestra un mensaje de éxito

Escenario 2 - nombre de producto con largo excedido

Dado que al ingresar un nombre con más de 30 caracteres en el formulario

Cuando presiono el botón *Modificar*

Entonces el sistema muestra un mensaje de error "Nombre excede el largo permitido"

Escenario 3 - descripción con largo excedido

Dado el ingreso de una descripción con más de 70 caracteres en el formulario

Cuando presiono el botón *Modificar*

Entonces el sistema muestra un mensaje de error "Descripción excede el largo permitido"

Escenario 4 - precio menor a cero

Dado el ingreso de un precio menor a cero en el formulario

Cuando presiono el botón *Modificar*

Entonces el sistema muestra un mensaje de error "El Precio no puede ser negativo"

Escenario 5 - código con largo menor al válido

Dado el ingreso de un código con un largo menor a 5 dígitos en el formulario

Cuando presiono el botón *Modificar*

Entonces el sistema muestra un mensaje de error "Código inválido"

Escenario 6 - código con largo mayor al válido

Dado el ingreso de un código con un largo mayor a 5 dígitos en el formulario

Cuando presiono el botón *Modificar*

Entonces el sistema muestra un mensaje de error "Código inválido"

Escenario 7 - código ya existente dentro de la farmacia

Dado el ingreso de un código perteneciente a otro producto dentro del sistema

Cuando presiono el botón *Modificar*

Entonces el sistema muestra un mensaje de error "Código inválido"

Título: Compra Productos

Narrativa:

Como cliente

Quiero realizar una compra que incluye productos

Para que la compra quede registrada en el sistema

Criterios de aceptación:

Escenario 1 - compra satisfactoria

Dado que al ingresar la cantidad de productos deseados al carrito

Cuando presiono el botón *Proceed to checkout*

Entonces muestra la pantalla *Checkout*

Evidencia de ejecución de casos de prueba

Para cada uno de los escenarios detallados anteriormente, se creó su correspondiente traducción a *cucumber* para su utilización en *Specflow*. Para cada una de las funcionalidades se definió un archivo de feature con su correspondiente step definition.

Alta producto

En este caso, se decidió crear dos grandes escenarios.

Para el caso de la correcta creación del producto se creó el siguiente escenario:

Scenario: Successful product creation

Given I am an authorized employee

When I add a new product with the name "<name>"

And the description "<description>"

And the code "<code>"

And the price "<price>"

Then the response status should be "<codeResponse>"

Tomamos la decisión de plantear el escenario como esquema con la correspondiente tabla en dónde se van especificando diferentes valores para los datos variables (los que se encuentran identificados con <>). Consideramos que esto era altamente beneficioso ya que permite a futuro agregar nuevos casos de prueba sin tener que modificar el código ya existente.

Se creó el siguiente escenario para cuando no se logra crear el producto ya que no cumple con alguno de los criterios de aceptación establecidos:

Scenario: UnSuccessful product creation

Given I am an authorized employee

When I add a new product with the name "<name>"

And the description "<description>"

And the code "<code>"

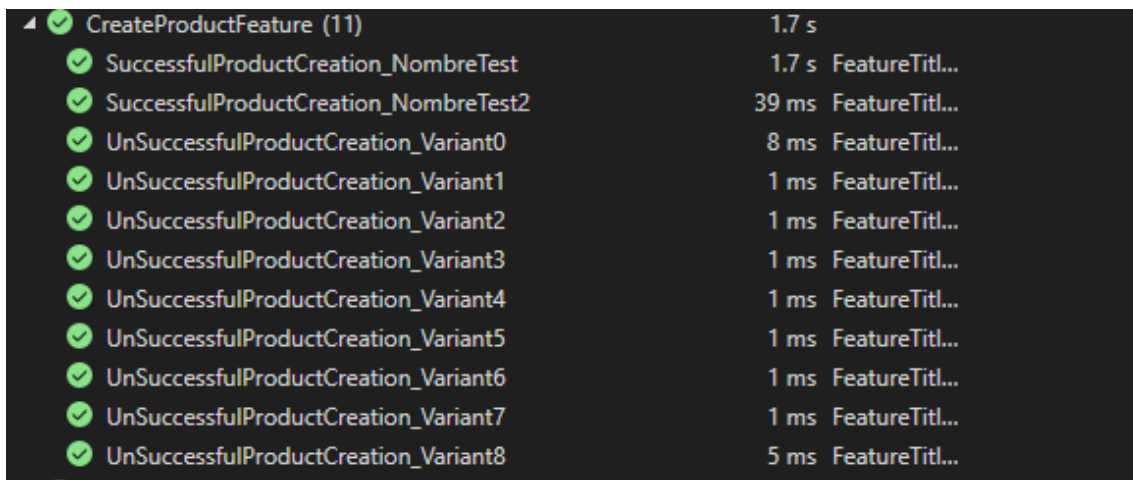
And the price "<price>"

Then the response message should be "<codeMessage>"

Al igual que lo mencionado anteriormente, la utilización del escenario como esquema permite agregar nuevos casos de prueba a futuro de forma sencilla sin tener que modificar el código existente.

Ejecución de las pruebas

En la siguientes imágenes se deja en evidencia la correcta ejecución de los casos de prueba:

A screenshot of a test runner interface showing a list of test cases for the 'CreateProductFeature' feature. The interface has a dark background with green checkmarks indicating successful test results. The table lists 11 test cases, including one successful creation test and eight unsuccessful creation variant tests, with their respective execution times and feature titles.

▲	✓	CreateProductFeature (11)	1.7 s	
	✓	SuccessfulProductCreation_NombreTest	1.7 s	FeatureTitl...
	✓	SuccessfulProductCreation_NombreTest2	39 ms	FeatureTitl...
	✓	UnSuccessfulProductCreation_Variant0	8 ms	FeatureTitl...
	✓	UnSuccessfulProductCreation_Variant1	1 ms	FeatureTitl...
	✓	UnSuccessfulProductCreation_Variant2	1 ms	FeatureTitl...
	✓	UnSuccessfulProductCreation_Variant3	1 ms	FeatureTitl...
	✓	UnSuccessfulProductCreation_Variant4	1 ms	FeatureTitl...
	✓	UnSuccessfulProductCreation_Variant5	1 ms	FeatureTitl...
	✓	UnSuccessfulProductCreation_Variant6	1 ms	FeatureTitl...
	✓	UnSuccessfulProductCreation_Variant7	1 ms	FeatureTitl...
	✓	UnSuccessfulProductCreation_Variant8	5 ms	FeatureTitl...

Baja producto

Para esta funcionalidad, se creó un único escenario. Esto se debe a cómo se implementó el frontend, en dónde simplemente se muestra la lista de productos y el usuario selecciona un botón para eliminar un producto determinado. Por lo tanto, al no tener que ingresar datos no se debe validar los mismos. El escenario desarrollado es el siguiente:

Scenario: Successful product deletion

Given I am an authorized employee deleting a product

When I choose the product with code "<codeProduct>" to be deleted

Then the response status code should be "<codeResponse>"

Ejecución de las pruebas

En la siguientes imágenes se deja en evidencia la correcta ejecución de los casos de prueba:

A screenshot of a test runner interface showing test results for the 'DeleteProductFeature' feature. The interface has a dark background with green checkmarks indicating successful test results. The table lists three test cases: two successful product deletions and one additional successful deletion, with their respective execution times and feature titles.

▲	✓	DeleteProductFeature (2)	20 ms	
	✓	SuccessfulProductDeletion_12370	14 ms	deleteProd...
	✓	SuccessfulProductDeletion_12371	6 ms	deleteProd...

Modificación producto

Al igual que con el alta de producto, en este caso creamos dos escenarios. El correspondiente a la correcta modificación de un producto y al caso en que por algún error de validación de datos no se pueda modificar el producto.

Los escenarios son los siguientes:

Scenario: Successful product update

Given I am an authorized employee who selected the product with code "<oldCode>"

When I update a product with the new name "<name>"

And the new description "<description>"

And the new code "<code>"

And the new price "<price>"

Then the response code should be "<codeResponse>"

Scenario: UnSuccessful product update

Given I am an authorized employee who selected the product with code "<oldCode>"

When I update a product with the new name "<name>"

And the new description "<description>"

And the new code "<code>"

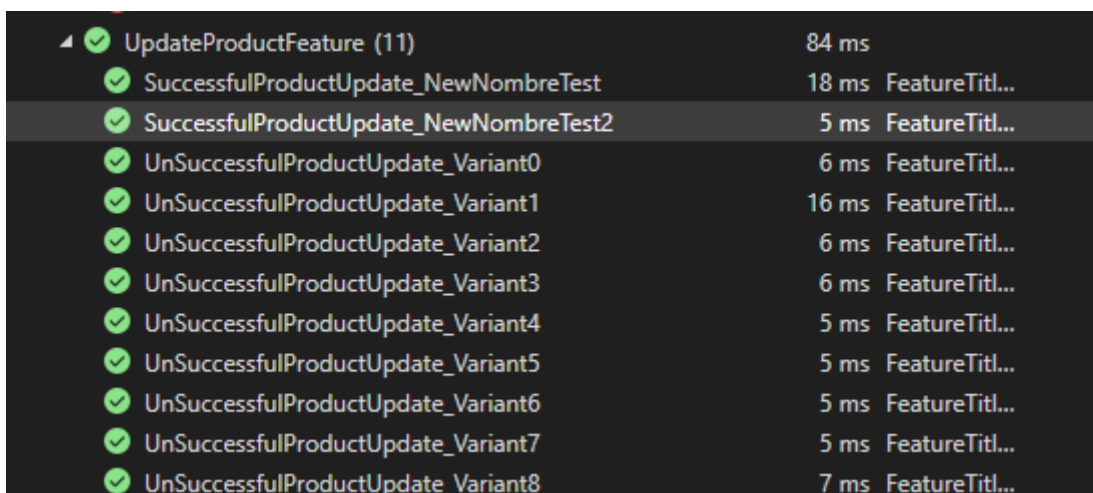
And the new price "<price>"

Then the error response message should be "<codeMessage>"

Al igual que las pruebas de alta de un producto se utilizaron escenarios como esquemas para permitir la extensibilidad de las pruebas, sin implicar la modificación del código ya existente.

Ejecución de las pruebas

En la siguientes imágenes se deja en evidencia la correcta ejecución de los casos de prueba:



UpdateProductFeature (11)	84 ms	
SuccessfulProductUpdate_NewNombreTest	18 ms	FeatureTitl...
SuccessfulProductUpdate_NewNombreTest2	5 ms	FeatureTitl...
UnSuccessfulProductUpdate_Variant0	6 ms	FeatureTitl...
UnSuccessfulProductUpdate_Variant1	16 ms	FeatureTitl...
UnSuccessfulProductUpdate_Variant2	6 ms	FeatureTitl...
UnSuccessfulProductUpdate_Variant3	6 ms	FeatureTitl...
UnSuccessfulProductUpdate_Variant4	5 ms	FeatureTitl...
UnSuccessfulProductUpdate_Variant5	5 ms	FeatureTitl...
UnSuccessfulProductUpdate_Variant6	5 ms	FeatureTitl...
UnSuccessfulProductUpdate_Variant7	5 ms	FeatureTitl...
UnSuccessfulProductUpdate_Variant8	7 ms	FeatureTitl...

Compra producto

En este caso, al igual que ocurre con la baja de un producto, se le muestra al usuario una lista de todos los productos existentes en el sistema y allí el mismo selecciona la cantidad que desea comprar de cada uno. Por lo tanto, se creó un único escenario que corresponde a la correcta compra de los productos.

El escenario planteado es el siguiente:

Scenario: Successful purchase

Given I'm a client

When I enter a purchase request

Then the response code should be "200"

En este caso, consideramos que el uso de un único escenario con datos concretos sería más beneficioso que el uso de un escenario como esquema.

Ejecución de las pruebas

En la siguiente imagen se deja en evidencia la correcta ejecución del caso de prueba:



✔ PurchaseProductFeature (1)	1.2 s
✔ SuccessfulPurchase	1.2 s FeatureTitl...

Review

Se realizó una reunión en donde los desarrolladores le mostraron al PO, en este caso Sebastián Daners, cómo se implementaron las nuevas funcionalidades y cómo las mismas cumplían con los criterios de aceptación determinados.

En este caso, todas las soluciones fueron aceptadas por el PO ya que cumplían con todos los escenarios planteados en los criterios de aceptación. Además, se logra completar satisfactoriamente las acciones requeridas por las funcionalidades.

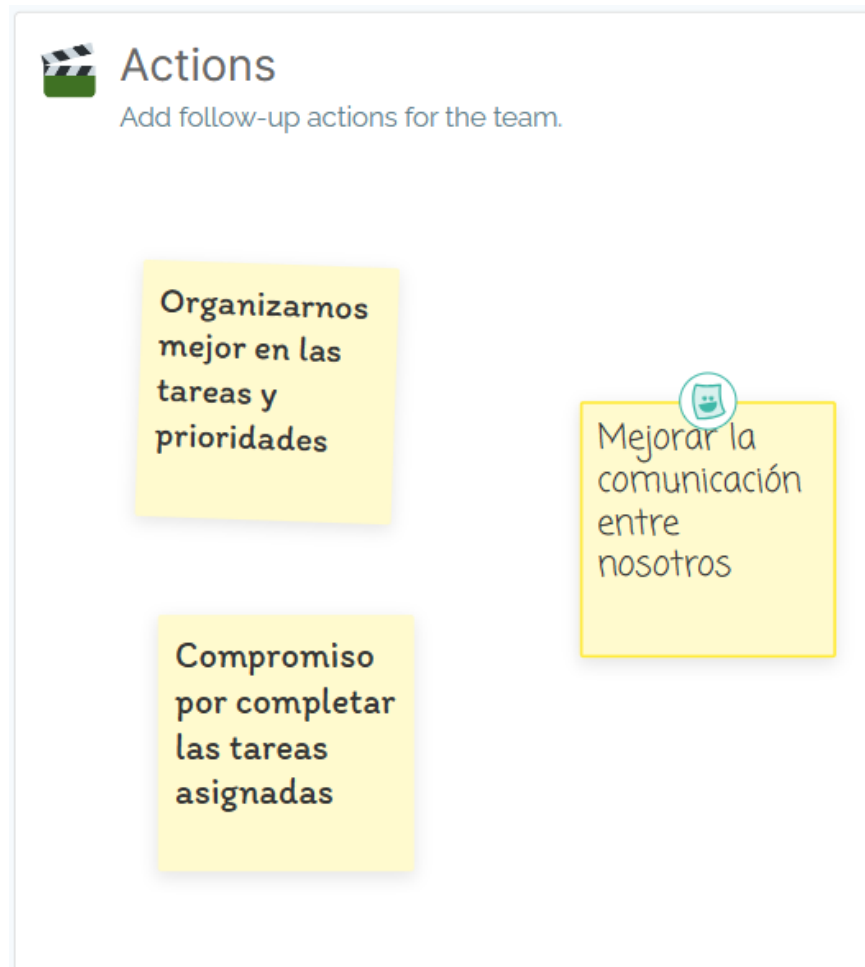
En el siguiente [link](#) se puede encontrar el video de la reunión.

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?
<div>No sacarnos las dudas con los profesores</div> <div>Empezar tarde</div>	<div>Organizarnos mejor en las tareas y prioridades</div>
Keep What do you think the team should keep?	Improve What do you think the team should improve?
<div>Compromiso por completar las tareas asignadas</div> <div>Seguir comenzando las entregas con tiempo.</div>	<div>Tuvimos mas tiempo para esta entrega de lo normal y eso termino haciendo que dejaramos mas para ultimo momento terminarla</div> <div>Mejorar la comunicación entre nosotros</div>

Cada uno fue explicando las ideas que planteó y en conjunto se llegaron a las siguientes ideas principales para implementar, mejorar y mantener para las próximas entregas:



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 implementaron satisfactoriamente todas las funcionalidades nuevas, se corroboró además que las mismas cumplieran con los criterios de aceptación establecidos en la etapa de *Requirement Definition*.

Por otra parte, se utilizó una nueva herramienta como lo es Specflow. Esto nos requirió un tiempo por fuera de lo relacionado directamente al proyecto para comprender la herramienta y aprender a utilizarla. La misma nos permitió aplicar BDD como metodología de desarrollo para los nuevos requerimientos, en el backend.