Universidad ORT Uruguay

Facultad de Ingeniería

Informe de avance de etapa: Entrega 2

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

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: PO, SM, DESA, TES

Todos los integrantes del equipo deben poseer los roles de DESA y TES. Por otro lado, en esta segunda 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. En cambio, la necesidad del Product Owner se debe a que otro artefacto recomendado es el video revisión de bugs con el product owner. Debido a que cada uno de nosotros arreglo un bug decidimos nombrar a dos product owners para que el bug que cada

uno arreglo sea revisado por otro integrante. A su vez, decidimos que el rol de Scrum Master rotará, por lo tanto este fue asignado a alguien distinto en comparación a la primera entrega.

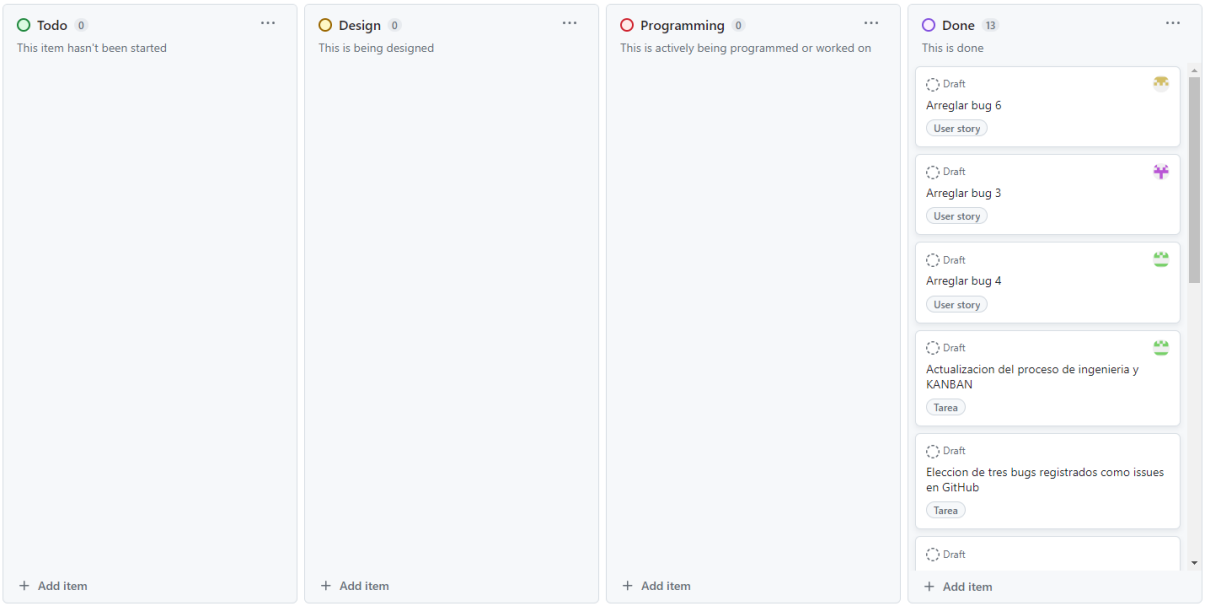
- **Sofia Montero**: Desarrollador, Tester, Product Owner
- **Romina Pardo**: Desarrollador, Tester, Product Owner
- **Andres Wilchinski**: Desarrollador, Tester, Scrum Master, Product Owner

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 la primera versión del tablero. Esta segunda versión consta de las siguientes columnas: ToDo, Design, Programming y Done. Para esta versión se mantiene la clasificación de tarea o user story.



Cada tarea y user story en el tablero tendrá la siguiente estructura:

- **Título** → Nombre corto descriptivo.

- **Descripción** → Descripción de lo que se debe hacer.
- **Fecha In ToDo** → Fecha en la que la tarea o user story se añadió a la columna ToDo.
- **Fecha In Diseño** → Fecha en la que la tarea o user story se mueve a la columna Diseño.
- **Fecha In Programming** → Fecha en la que la tarea o user story se mueve a la columna Programming.
- **Fecha In Done** → Fecha en la que la tarea o user story se marca como completada y se traslada a la columna Done.
- **Esfuerzo** → Estimación del esfuerzo necesario para completar la tarea o user story (opcional).
- **Responsable** → Nombre del responsable de la tarea o user story.

El flujo de trabajo en este nuevo tablero es el siguiente: inicialmente, todas las tareas se agregan a la columna ToDo. Cuando se comienza a trabajar en una tarea, en el caso de precisar un diseño previo, se mueve a la columna Design. Luego, cuando comienza la programación o cuando se está trabajando en la tarea o user story, esta se mueve a la columna Programming. Finalmente, al completarse, se coloca en la columna Done. El registro de fechas en las diferentes etapas del proceso permite un seguimiento preciso del progreso de cada tarea y facilita la gestión del proyecto.

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.

Explicación del pipeline y su vínculo con el proceso de ingeniería

Como en esta entrega el objetivo era solucionar bugs y realizar algunas tareas, decidimos cambiar un poco el tablero como se menciona anteriormente. El tablero ahora contiene las columnas ToDo, Design, Programing y Done. Decidimos separar la columna doing de la entrega pasada en Design y Programing para poder diferenciar mejor y saber cuánto tiempo pasaba cada tarea en cada una de estas áreas y poder saber mejor el estado actual de la tarea cuando esta se estaba realizando. Al ser código, como el ejemplo solucionar un bug, este consta de 3 pasos, el diseñar la solución, codificar y testear. (El análisis de relevamientos en el caso de corregir un bug no es necesario.) Como decidimos utilizar la estrategia de TDD (test driven development), el testing se hace en paralelo con el desarrollo. En TDD primero se hace el test y después se crea el código mínimo que pase el test y luego se refactoriza. Debido a esto decidimos unir las columnas desarrollo y test en una sola la cual llamamos programming.

Pipeline con Git Actions

Como decidimos utilizar trunk base lo cual implica mergear a main constantemente y tener en main una versión siempre deployable y lista, decidimos crear una pipeline con github action que haga el build y corra los test de nuestro proyecto para así tener un mayor grado de confianza de que los cambios nuevos

que se introducen a main no rompen nada y el sistema va a seguir funcionando con normalidad. Como primer paso solo hicimos que se genere el build del backend y se corran los test del mismo ya que era algo fácil de hacer porque ya poseíamos los test del backend debido a que el desarrollo del mismo se hizo mediante el uso de TDD. La pipeline se dispara cada vez que se pushea un commit nuevo o a la hora de realizar una pull request a la branch main en la cual se haya cambiado algún archivo .cs o el .csproj. Decidimos probar correrlo en los 3 sistemas operativos windows, macOS y ubuntu, para asegurarnos que el backend podría correr sin problema en cualquiera de estos 3 sistemas operativos. La pipeline realiza 3 acciones. Primero descargar el código de github e instala las dependencias necesarias, luego hace el build de la aplicación , esto nos asegura que ningún cambio introducido hace que el código deje de compilar. Por último corre los test del backend, lo cual nos da bastante certeza de que los cambios introducidos no rompen nada y el código sigue funcionando como uno esperaría.

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.

Sobre los Issues

Registro de Issues

Para registrar los issues utilizamos la plataforma de registro de issues de github. Cada una asociada a distintas labels, donde se registra que tipo de issue, el tipo de severidad y de prioridad. A su vez, nos permite poder ver los issues ya resueltos y realizar hilos de conversación dentro de el apartado de esa issue.

Puede ver más aquí.

Forma de trabajo para elección y resolución de los Issues

Para esta entrega se decidió en tres issues a resolver, estos fueron divididos entre los desarrolladores, que deben resolverlos con TDD en una rama aparte que debe ser mergeada con la rama principal. Elegimos resolver los Bugs 3, 4 y 6 porque eran los bugs con mayor severidad y prioridad.

Bug 3:

Severidad: Mayor, función principal que no cumple los requisitos y se comporta de manera distinta de lo esperado.

Prioridad: Media, plazo máximo un par de semanas.

Descripción: Permitir rechazar una compra cuando el medicamento no se encuentra en la farmacia, debido a que este fue borrado, ya que en este momento no se permite aceptar o rechazar la compra de dicho medicamento porque no se encuentra disponible. El sistema debería permitir rechazar dicho pedido o que el sistema lo rechace de forma automática.

Bug 4:

Severidad: Mayor, función principal que no cumple los requisitos y se comporta de manera distinta de lo esperado.

Prioridad: Media, plazo máximo un par de semanas.

Descripción: Al exportar los medicamentos de una farmacia se encuentran los medicamentos de la farmacia, al igual que aquellos que fueron previamente eliminados de la misma. Un medicamento eliminado no debería aparecer a pesar de que el borrado se haya implementado de manera lógica y no

sea realmente un borrado físico. Para el usuario el borrado debería funcionar como si fuese borrado de verdad físico.

Bug 6:

Severidad: Mayor, función principal que no cumple los requisitos y se comporta de manera distinta de lo esperado.

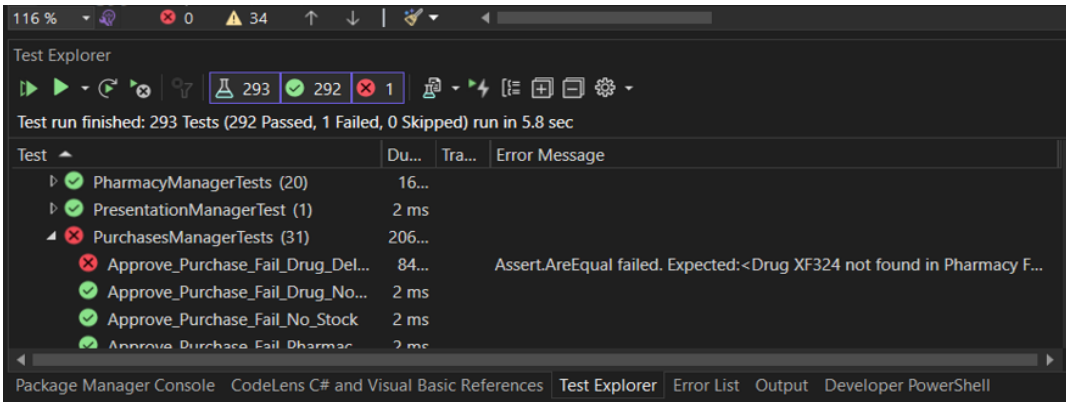
Prioridad: Alta, plazo máximo 48 hs.

Descripción: Al hacer el update de una invitación, se permite cambiar el código a un código de menos de 6 dígitos, lo cual no debería pasar nunca según la letra.

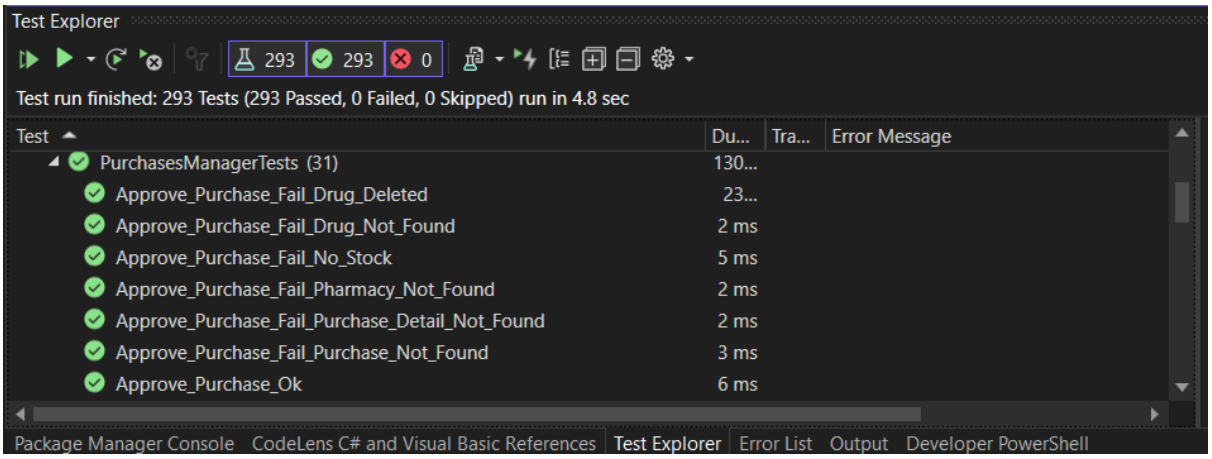
Uso de TDD

Bug 3

En la imagen a continuación se puede ver el uso de TDD, ya que al principio se agregó el test de que al aprobar una compra de un medicamento borrado de mensaje nuevo y eso todavía no estaba cambiado en el código.

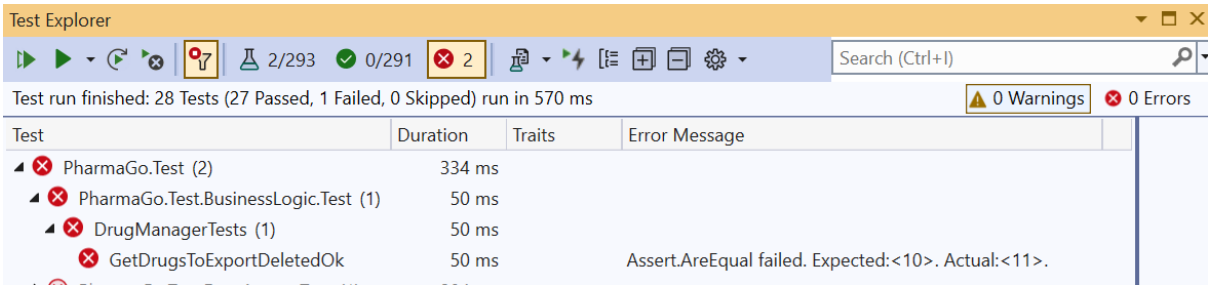


Luego del cambio en el código:

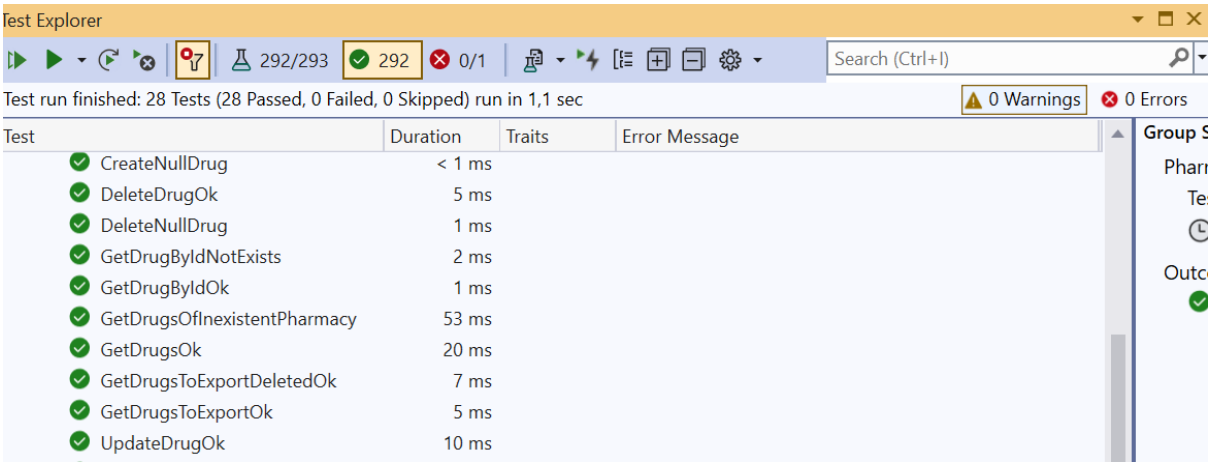


Bug 4

Para realizar TDD primero se crea el test GetDrugsToExportDeletedOk en la etapa RED, que significa que este test no debería pasar



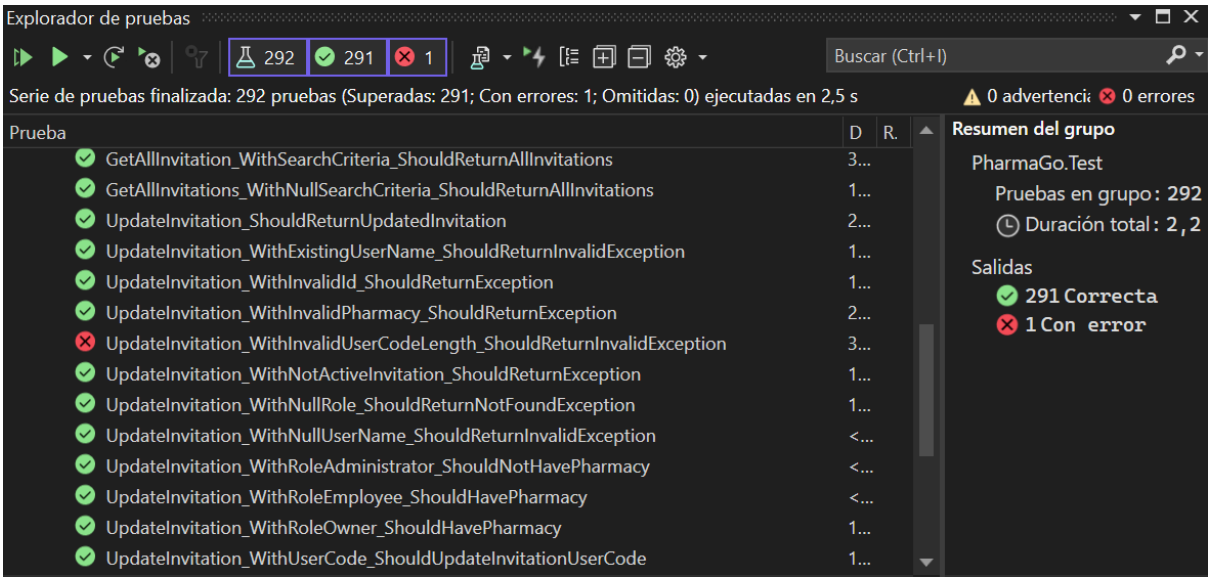
Luego, se crea el código necesario para que la prueba pase correctamente (GREEN) y que el bug quede resuelto. Este código consiste en filtrar aquellos medicamentos de la farmacia que fueron eliminados previo a realizar la exportacion asi solo se exportan aquellos vigentes



Bug 6

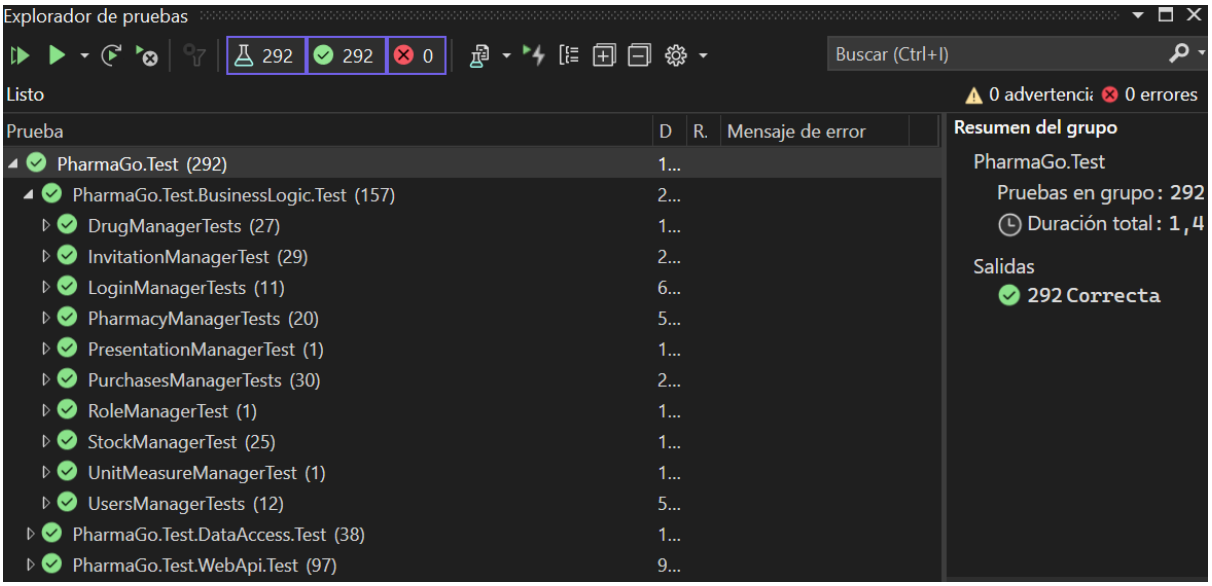
Se agregó un test en el backend para corroborar que funcione correctamente. Se hizo en PharnaGo.Test/BusinessLogic.TestInvitationManagerTest con el nombre de UpdateInvitation_WithInvalidUserCodeLength_ShouldReturnInvalidException.

Antes de agregar los cambios para solucionar el bug 6 (RED)



Se modifico el archivo PharmaGo.BusinessLogic/InvitationManager.cs del backend, agregando una condición en la función UpdateInvitation.

Después de agregar los cambios para solucionar el bug 6 (GREEN)

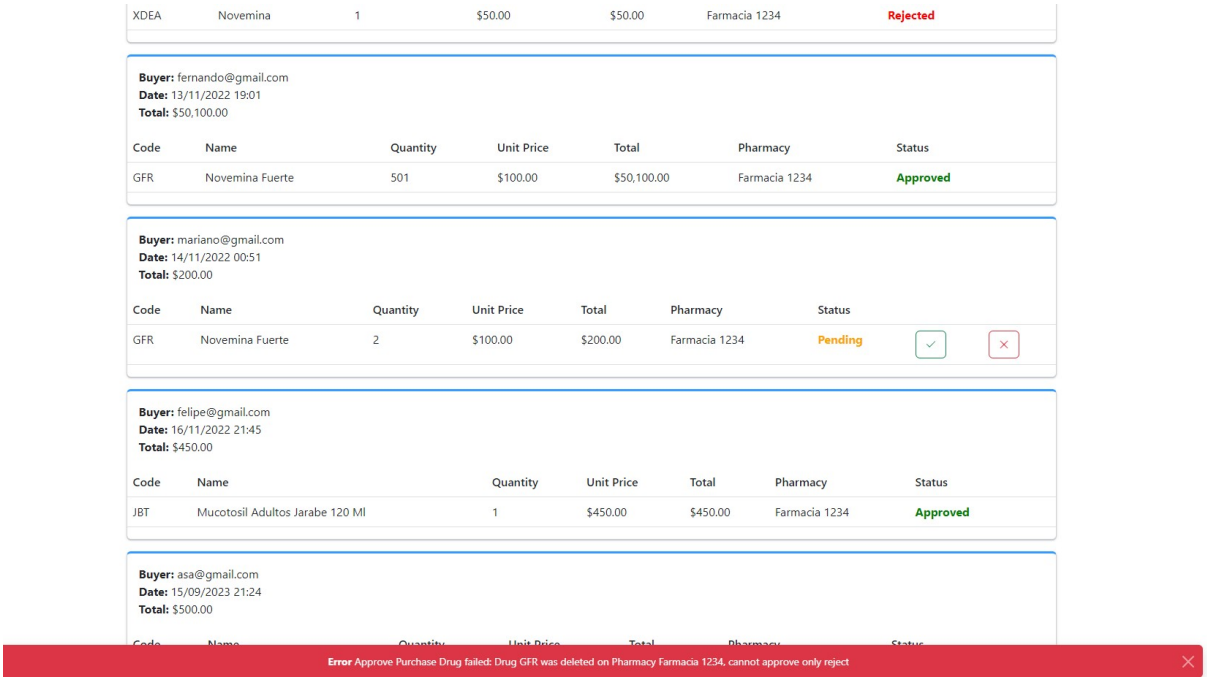


Casos de Prueba y review con product owner

Para realizar la review con el product owner decidimos realizar una llamada entre los tres y turnar el rol de Product Owner para que los tres podamos pasar por el rol. Durante la review el desarrollador muestra un caso de prueba de la solución del respectivo bug y luego el Product Owner aporta feedback sobre la solución realizada.

Bug 3

Para resolver el bug de que se permita rechazar la compra de un medicamento borrado lo que se hizo fueron 2 cambios. Primero se cambió el mensaje que no permitía el aceptar la compra de un medicamento borrado el cual antes era “Drug {drugCode} not found in Pharmacy {pharmacy.Name}” por uno más entendible para el usuario “Drug {drugCode} was deleted on Pharmacy {pharmacy.Name}, cannot approve only reject”. Para esto se realiza un test y a su vez se cambió el código de rechazar una compra para que permita rechazar compras de remedios borrados. De esta manera el empleado de la farmacia podría rechazar los pedidos de medicamento que la farmacia ya borro y no posee más y no podría ni siquiera por error aprobar dichas compras.



Bug 4

Para resolver el Bug 4 se debe solucionar los medicamentos exportados ya que en un principio al realizar una exportación, los medicamentos que aparecen en la lista son todos aquellos que pertenecen a la farmacia aunque hayan sido eliminados, en este caso el objetivo era que solo sean exportados aquellos que no fueron eliminados, por lo tanto, también decidimos eliminar el atributo de Deleted del modelo final.

En un principio, con la base de datos inicial del proyecto la lista resultante de exportaciones realizando el login con employee1 era la siguiente:

[{"Code":"XDEA","Name":"Novemina","Symptom":"Dolor de cabeza","Deleted":true},{"Code":"ABCD","Name":"Perifar Flex","Symptom":"Fiebre","Deleted":true},{"Code":"ZASW","Name":"Aspirina","Symptom":"Dolor en general","Deleted":true},{"Code":"BIO","Name":"Bio Grip Plus 8 Comprimidos","Symptom":"Gripe","Deleted":true},{"Code":"ASCX","Name":"aaaaa","Symptom":"eeeee

eee","Deleted":true},{"Code":"ZXZ","Name":"qwewr","Symptom":"asdda","Deleted":true},{"Code":"WWW","Name":"asdasasfasf","Symptom":"sadsadasd","Deleted":true},{"Code":"GFR","Name":"Novemina Fuerte","Symptom":"Dolor general","Deleted":false},{"Code":"VGT","Name":"Paracetamol 1 Gr 10 Comprimidos ","Symptom":"Gripe, Fiebre, Dolor de cabeza","Deleted":false},{"Code":"ZAQ","Name":"Teofilina Efa 250 Mg 30 Comprimidos","Symptom":"Metilxantina broncodilatadora indicada en las crisis asmáticas y otras situaciones que cursen con broncoespasmo.","Deleted":false},{"Code":"FRE","Name":"Alerfedine 120 Mg 10","Symptom":"Antialérgico.","Deleted":false},{"Code":"JBT","Name":"Mucotosil Adultos Jarabe 120 Ml","Symptom":"Mucolítico","Deleted":false}]

Luego de solucionarlo la lista resultante es:

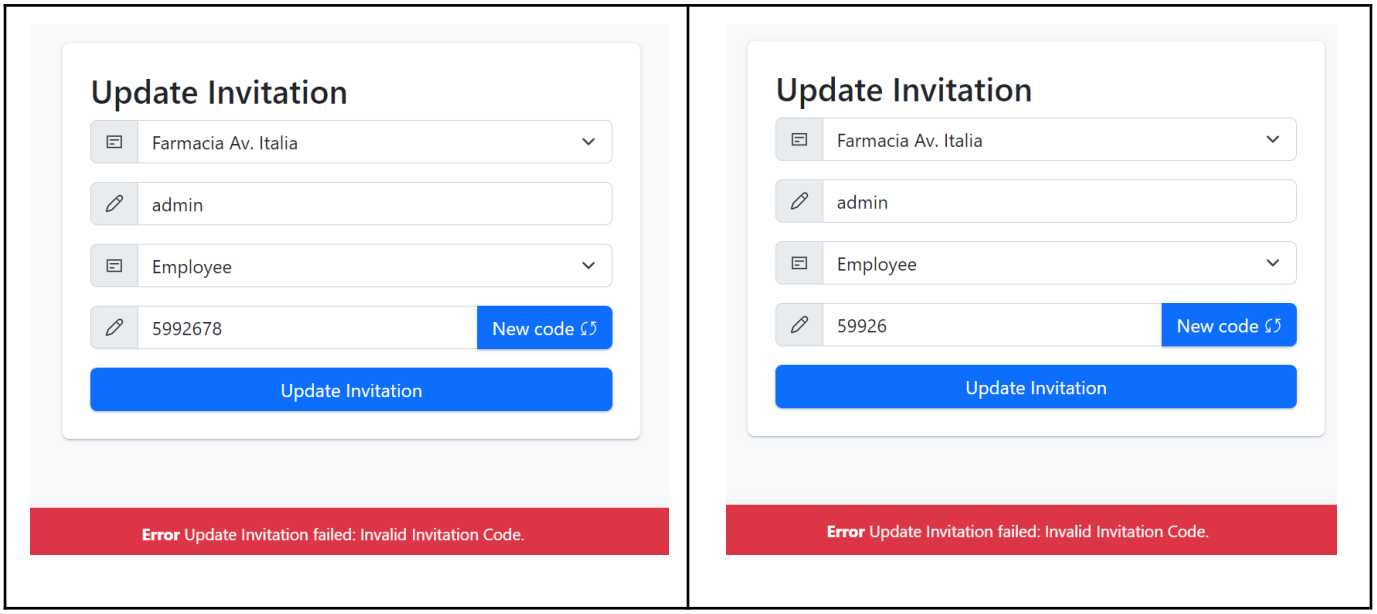
[{"Code":"GFR","Name":"Novemina Fuerte","Symptom":"Dolor general"},{"Code":"VGT","Name":"Paracetamol 1 Gr 10 Comprimidos ","Symptom":"Gripe, Fiebre, Dolor de cabeza"},{"Code":"ZAQ","Name":"Teofilina Efa 250 Mg 30 Comprimidos","Symptom":"Metilxantina broncodilatadora indicada en las crisis asmáticas y otras situaciones que cursen con broncoespasmo."},{"Code":"FRE","Name":"Alerfedine 120 Mg 10","Symptom":"Antialérgico."},{"Code":"JBT","Name":"Mucotosil Adultos Jarabe 120 Ml","Symptom":"Mucolítico"}]

Durante la review con el product owner se muestra la lista de medicamentos resultante, se elimina un medicamento y se vuelve a realizar la exportación mostrando que el medicamento eliminado ya no esta en la lista resultante de exportaciones.

Bug 6

Modificando la invitación original para el empleado con nombre admin de la farmacia de av.italia. Se cambia el código a uno con menos o más de 6 dígitos y se observa que no permite modificar la base de datos. A su vez, se puede observar que el código sigue funcionando correctamente cuando el código es de 6 digitos.

Con código 7 dígitos:	**Con código 5 dígitos:**



Resumen de Issues

Prioridad/Severidad	*Crítico*	*Mayor*	*Menor*	*Leve*
Inmediata	0	0	0	0
Alta	0	0	0	0
Media	0	0	2	0
Baja	0	0	7	2

Análisis del trabajo

Registro de las daily's

19/09/2023: mensaje de texto por whatsapp

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

- No

P2: Como equipo , ¿qué estamos buscando finalizar?

- En esta conversación se dialogó sobre la letra del obligatorio. Planteando preguntas e iniciando una estructuración de lo que hacer.

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

- No

20/09/2023: llamada en teams

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

- No

P2: Como equipo , ¿qué estamos buscando finalizar?

- En esta reunión se buscó concretar las tareas y user stories necesarias para la segunda entrega, en este momento nuestro foco está en la resolución de bugs del sistema

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

- No

23/09/2023: llamada en teams

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

- No

P2: Como equipo , ¿qué estamos buscando finalizar?

- Queremos finalizar las tareas necesarias para concretar la segunda entrega, en especial trabajar en la documentación.

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

- Los cuellos de botella que existían fueron solucionados, en este momento no hay impedimentos al flujo de trabajo

27/09/2023: llamada en teams

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

- No

P2: Como equipo , ¿qué estamos buscando finalizar?

- Las tareas quedaron finalizadas

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

- Los cuellos de botella que existían fueron solucionados, en este momento no hay impedimentos al flujo de trabajo

27/09/2023: llamada en teams, retrospectiva

Ver siguiente apartado.

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:

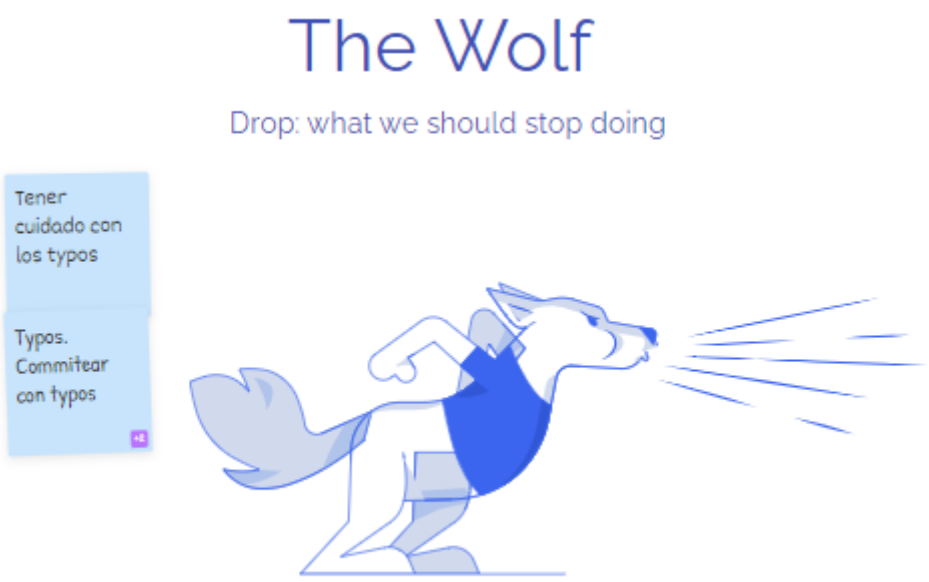
- 5 minutos iniciales para responder todas las secciones. Esto se hace de forma anónima.
- Abrir las notas para que todos puedan ver lo que se escribió.
- Dialogar sobre cada sección y discutir para llegar a una respuesta conjunta.
- Luego, para finalizar, se eligiran las ideas más importantes de las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

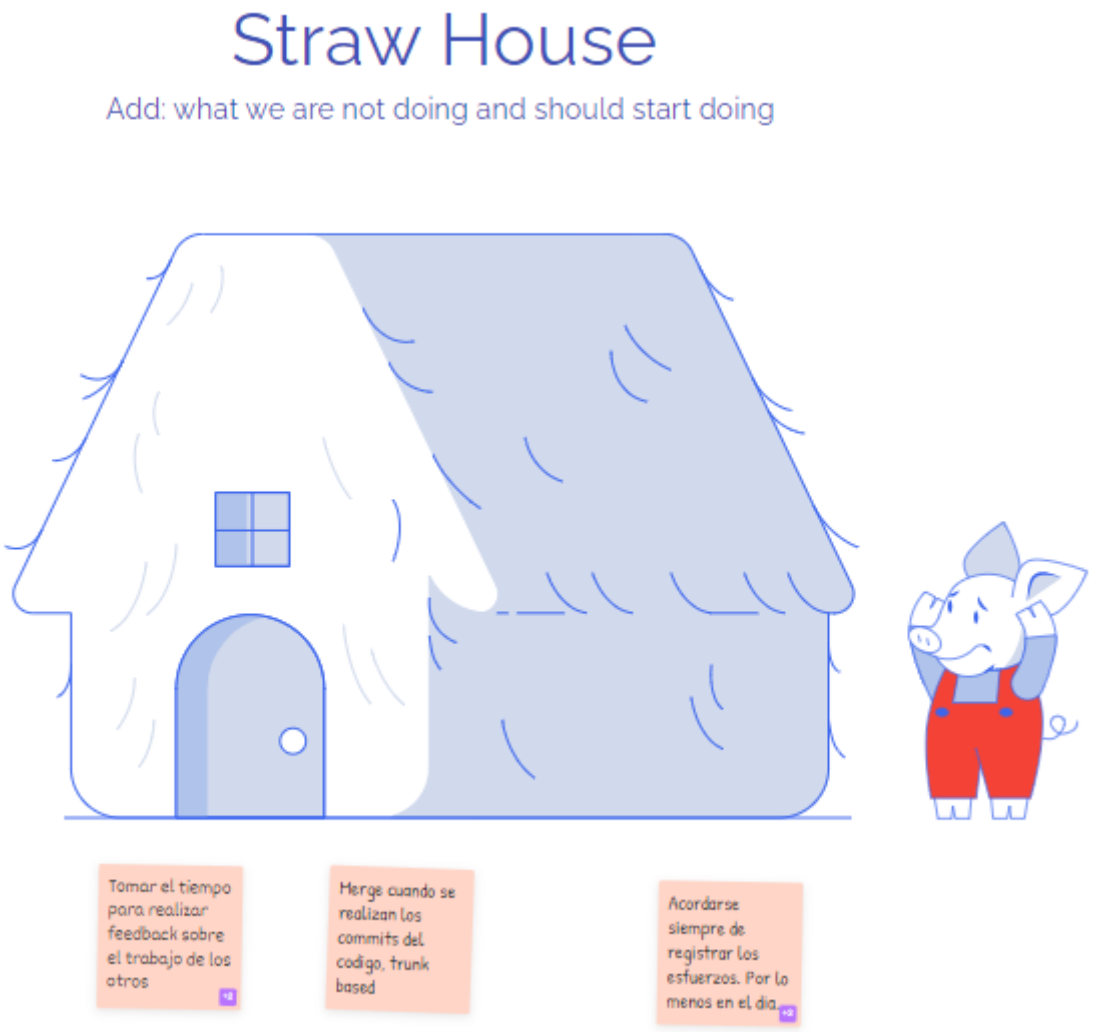


La idea más importante destacada fue: “Tener un poco más de cuidado con la ortografía a la hora de realizar commits y en la documentación ”

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?

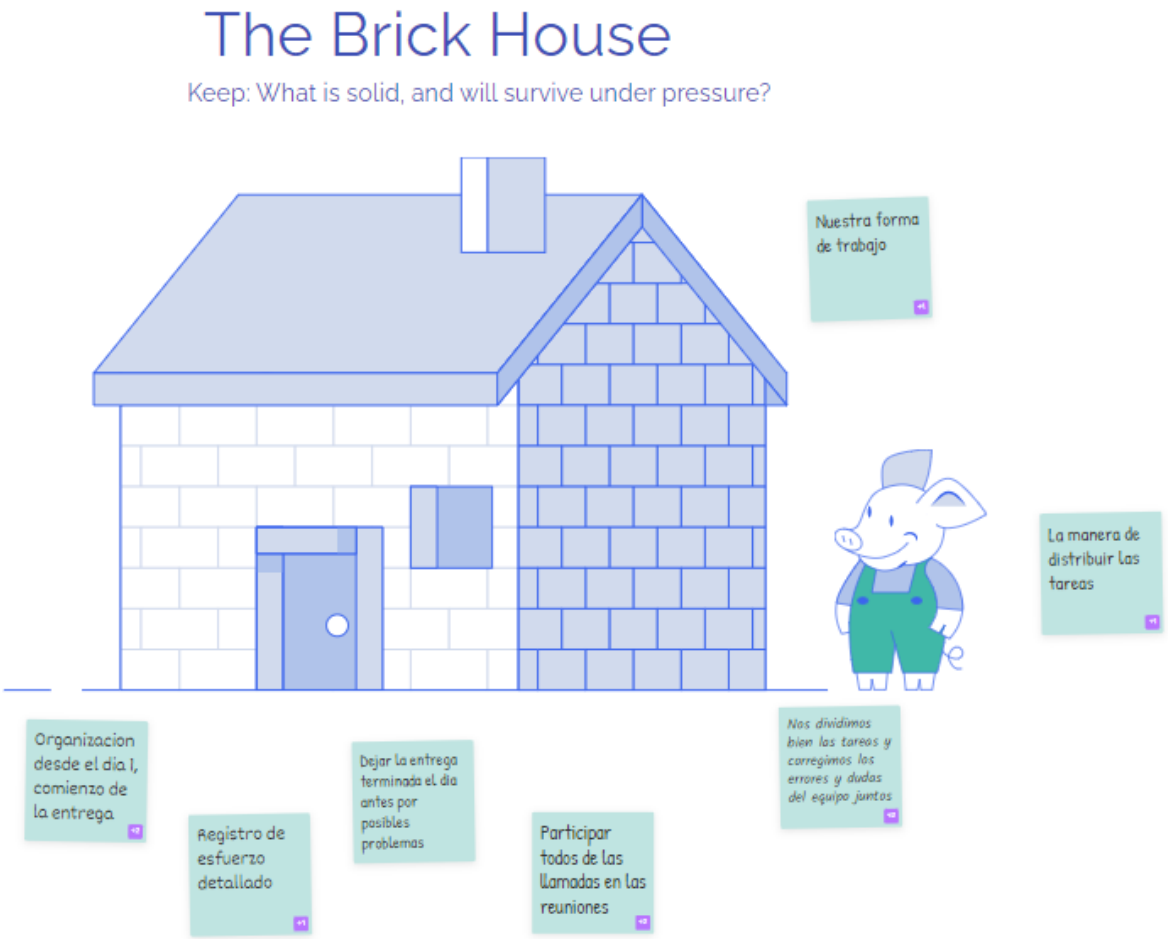


La idea más importante destacada fue: “Intentar acordarnos de registrar los esfuerzos cuando se termina una tarea y tomar el tiempo para realizar feedback”

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?

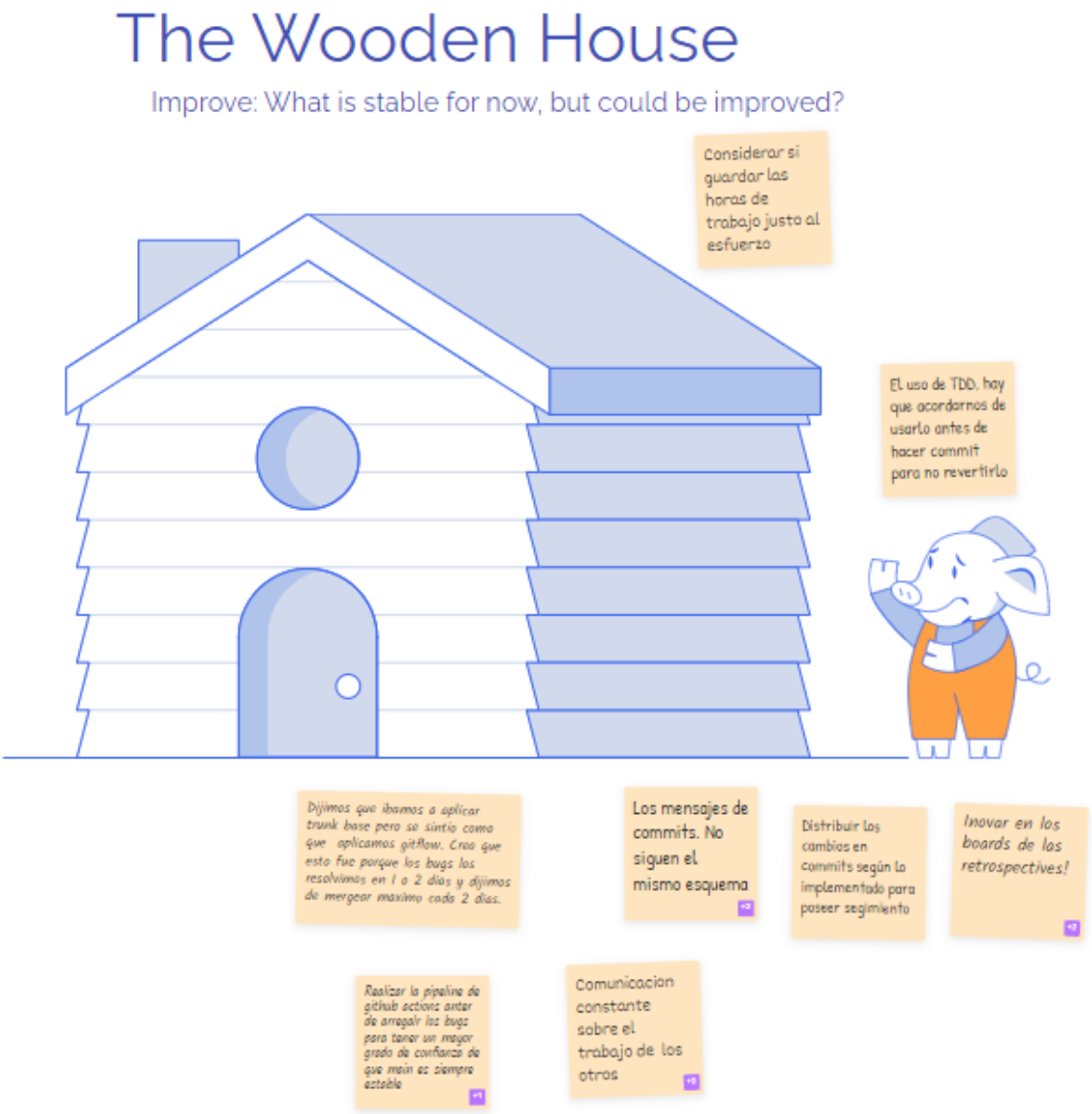


La idea más importante destacada fue: “Buena organización”

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: “Mejorar el esquema de commits y un correcto uso de TDD”

Registro de la Review

Se realizaron 3 videos cada uno para un bug. Aqui estan los links:

- Bug 3
- Bug 4
- Bug 6

Registro de avance

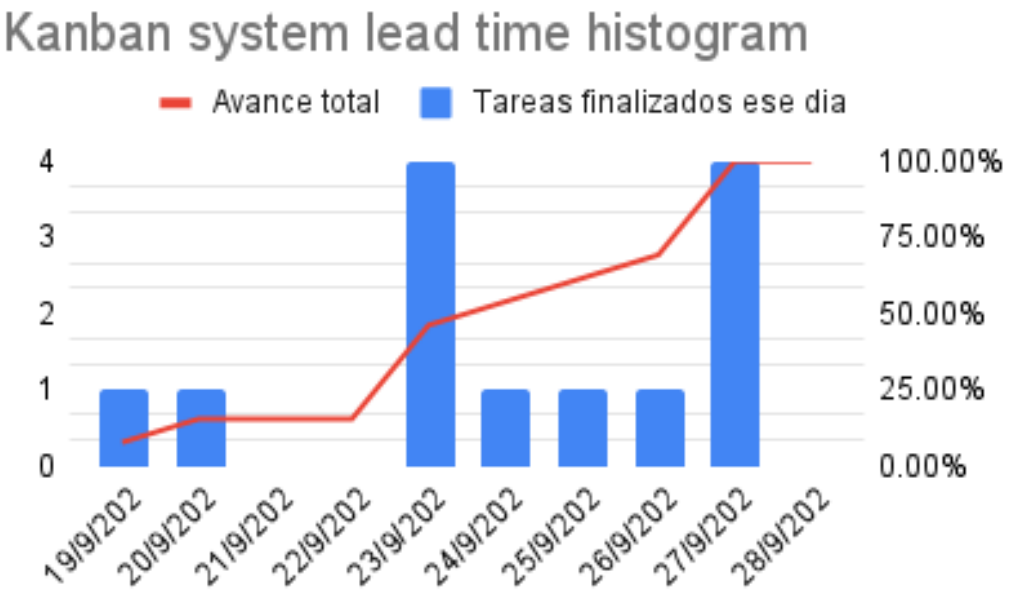
Registro del trabajo y Esfuerzo

Nro sub tarea	Tipo	Descripción	Horas/Esfuerzo persona	Responsable	Fecha In Todo	Fecha In Diseño	Fecha In Programing	Fecha In Done	Cycle Time (dias)	Lead Time (dias)
1	Tarea	Creación de segunda versión del tablero	0:30:00	Romina	19/9/2023	19/9/2023	19/9/2023	19/9/2023	0.00	0.00
2	Tarea	Ingreso de tareas al tablero de KANBAN	1:00:00	Todos	19/9/2023	19/9/2023	19/9/2023	27/9/2023	8.00	8.00
3	Tarea	Actualización del proceso de ingeniería y KANBAN	1:30:00	Sofia	19/9/2023	23/9/2023	23/9/2023	25/9/2023	2.00	6.00
4	Tarea	Configuración del pipeline	1:00:00	Andres	19/9/2023	23/9/2023	23/9/2023	24/9/2023	1.00	5.00
5	Tarea	Elección de tres bugs registrados como issues en GitHub	0:30:00	Todos	19/9/2023	20/9/2023	20/9/2023	20/9/2023	0.00	1.00
6	User Story	Fix Bug 3	1:30:00	Andres	20/9/2023	23/9/2023	23/9/2023	23/9/2023	0.00	3.00
7	User Story	Fix Bug 4	1:30:00	Sofia	20/9/2023	22/9/2023	22/9/2023	23/9/2023	1.00	3.00
8	User Story	Fix Bug 6	1:00:00	Romina	20/9/2023	23/9/2023	23/9/2023	23/9/2023	0.00	3.00
9	Tarea	Actualización de explicación de tablero	0:30:00	Romina	20/9/2023	23/9/2023	23/9/2023	23/9/2023	0.00	3.00
10	Tarea	Gia de configuración del pipeline	1:00:00	Andres	20/9/2023	26/9/2023	26/9/2023	26/9/2023	0.00	6.00
11	Tarea	Informe de bugs	1:00:00	Sofia	20/9/2023	25/9/2023	25/9/2023	27/9/2023	2.00	7.00
12	Tarea	Registro de esfuerzo y metricas	1:00:00	Todos	20/9/2023	20/9/2023	20/9/2023	27/9/2023	7.00	7.00
13	Tarea	Evidencia de casos de prueba	1:30:00	Todos	20/9/2023	24/9/2023	24/9/2023	27/9/2023	3.00	7.00

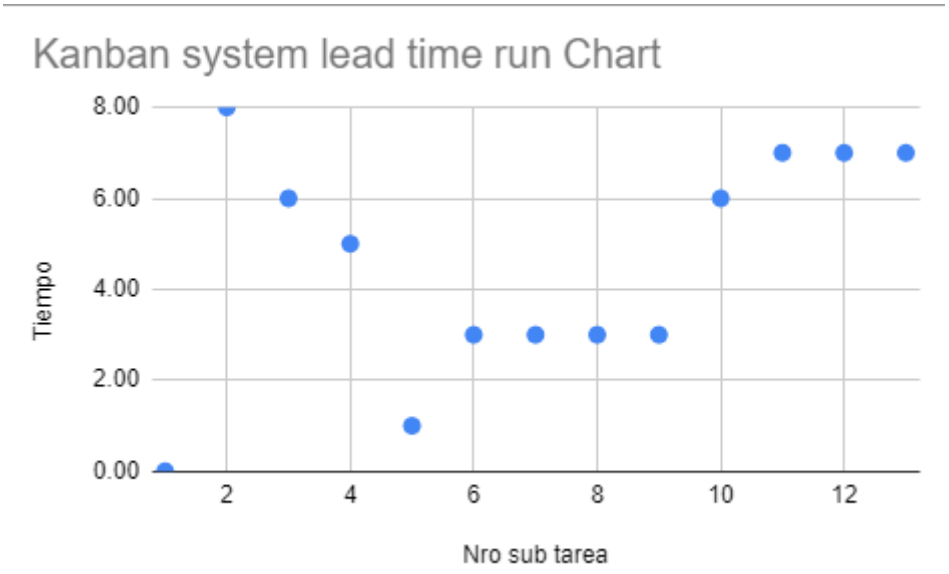
Para esta entrega contamos con 13 tareas y el total del registro de esfuerzo fue de 13 horas y media.

Podrá ver el seguimiento realizado aquí.

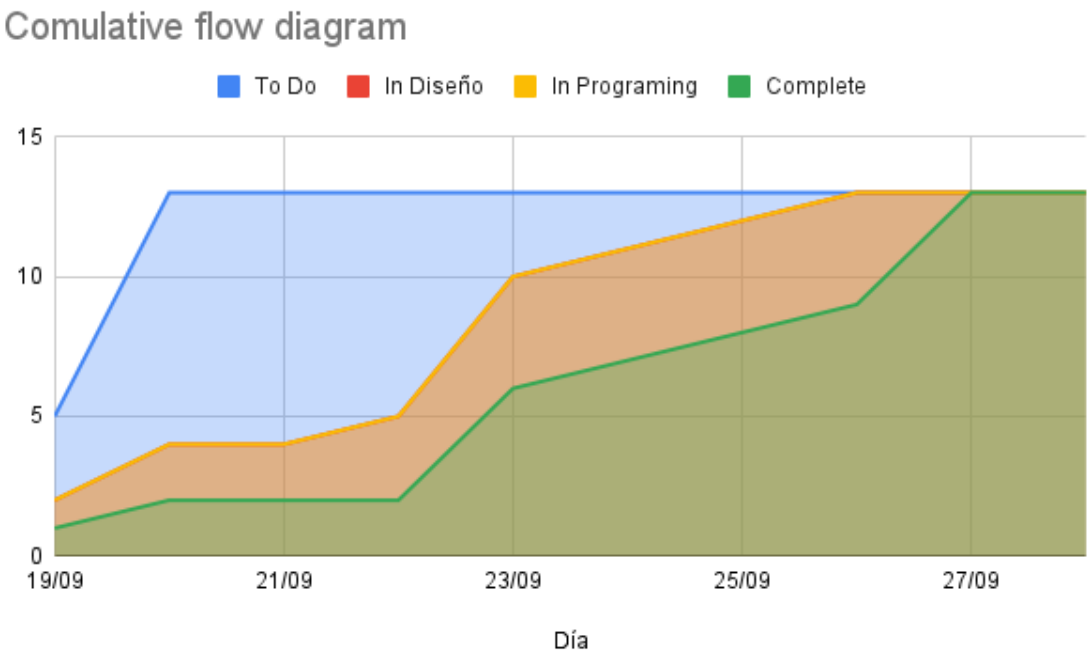
Métricas



Esta gráfica mide cuántas tareas se terminaron por día (azul) y el porcentaje del proyecto que teníamos terminado hasta ese momento (rojo). Se puede ver que logramos completar la totalidad del trabajo que nos propusimos a tiempo. También podemos ver que los días que más trabajamos, en realidad qué más tareas terminamos fueron los días sábado 23 y miércoles 27.

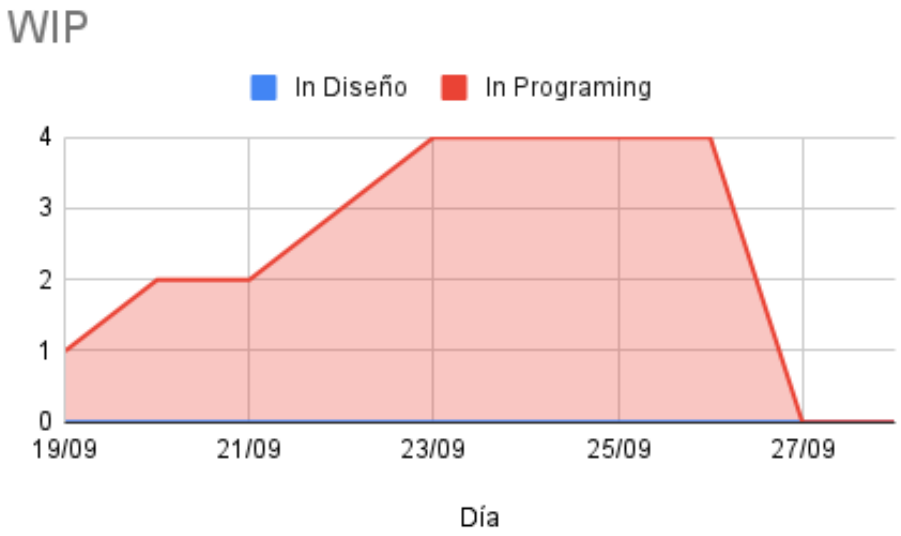


En esta gráfica se puede observar el tiempo que llevó cada tarea/user story desde que se definio hasta estar completa (pasar de Todo a Done). Se puede ver cómo en promedio las tareas demoran 4-5 días estar prontas. También se pueden ver algunos outliers por ejemplo la tarea 2 que nos tomó 8 días tenerla pronta. Esto no quiere decir que dicha tarea nos tomo 8 días realizarla sino que puede ser que dicha tarea estuvo 4 días en el ToDo sin que nadie la toque y luego demoro otros 4 días en realizarse. Para saber cuanto nos tomo realizar dicha tarea uno debe ver el cycle time (tiempo in progress, o sea desde design hasta Done) y no el leadtime.



Se ve la acumulación de cuantas tareas llegaron a cada columna. Si se desea saber cuántas tareas estaban en tal columna tal día, esto se puede hacer calculando la diferencia vertical en ese día entre una

gráfica y su sucesora. Ejemplo si deseamos saber cuántas cards (user stories o tareas) estaban en la columna “Programming” el día 23/09 simplemente le restamos al valor de la curva amarilla (Programming), 10, el valor de la curva verde (Complete), 6. Realizando la cuenta podemos ver que el día 23 había 4 cards en la columna Programming, igual a lo que nos dice la gráfica siguiente (WIP) para ese día. También se puede observar que no hay línea roja (Diseño) esto se debe a que está debajo de la línea amarilla y por esta razón no es visible. Esto sucede porque ninguna tarea quedó en diseño por un día o más. Esto se puede ver mucho más claramente en la gráfica a continuación.



En esta gráfica se podrá observar las tareas y/o user stories que se encontraban en cada columna en cada día. Se puede observar que en la columna In diseño al finalizar el día las tareas que ahí se encontraban se habían movido a In Programing. Es por esta razon que ningun día se termino con tareas en In diseño por lo cual se puede ver que la curva azul esta en 0 siempre.

Resultados obtenidos en el período

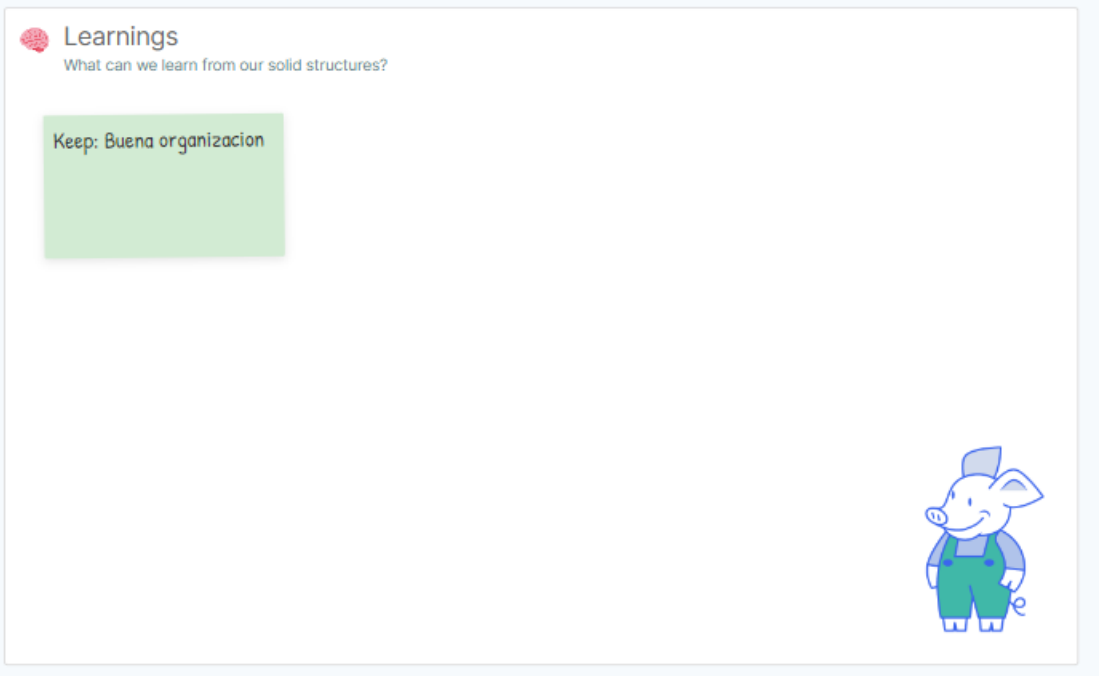
El objetivo de esta entrega, y sus resultados, fue resolver tres issues de los encontrados en la entrega anterior documentando y actualizando las distintas guías relacionadas. A su vez, se implementaron las métricas las cuales nos dan una representación de lo realizado.

Dificultades encontradas y formas de solución

Las dificultades encontradas en esta fue la creación del pipeline, ya que se encontraron errores los cuales costaron un poco resolverlos.

Lecciones aprendidas y mejoras en el proceso

Aprendimos a hacer pull requests y a trabajar con el método de Trunk Base. A su vez se utilizó git actions por primera vez y también se aprendió a hacer reviews.



Con respecto a la entrega anterior

La diferencia más grande fue que se trabajó con código y con el pipeline, lo cual le da dificultad al proyecto. También nos enfrentamos por primera vez a la resolución de issues sobre un código que no

creamos nosotros, lo cual implica entender la lógica de razonamiento de una persona ajena a nosotros y su forma de trabajo.