Entrega 2

Ingeniería de software ágil 2

Integrantes:

Rodrigo Haiache - N° 197344
Gabriel Lutz - N° 173507
Marcos Novelli - N° 201663
Martina Praderio - N° 242494

N7A - 2022

Universidad ORT del Uruguay

Repositorio de GitHub
https://github.com/ORT-ISA2-2022S2/obligatorio-equipo-6

Índice

Desarrollo/mantenimiento del nuevo tablero usando BDD

Para esta entrega al tener nuevas funcionalidades que realizar utilizando BDD, adaptamos nuestro tablero para un tablero sustentable. Para poder realizar el flujo de las tareas no asociadas a ninguna user story, a los bugs y a las nuevas funcionalidades juntas, se decidió fusionar el tablero anterior con un nuevo tablero sustentable.

El nuevo tablero se puede ver aquí

Flujo de Issues

Las nuevas columnas agregadas para este flujo fueron:

Requirements Definition (CCC): Aquí el desarrollador encargado, teniendo el nombre y la narrativa de la user story define los criterios de aceptación de la misma con sus distintos escenarios en lenguaje Gherkin.

Test Cases Implementation (Automation): Se tiene la user story con lo hecho en Requirements Definition y se implementan los casos de prueba con herramientas de automatización, en donde nosotros utilizamos SpecFlow.

App Implementation (Code): En este paso se codifica la funcionalidad de la user story en lenguaje C# cumpliendo todos los criterios de aceptación

Testing (Automation Tools): Teniendo ya la funcionalidad codificada, en este paso se corren los casos de prueba implementados verificando que todos pasen, sino se corrige el código hasta que todos los casos pasen.

Refactoring: Al tener nuestra funcionalidad hecha ya con todos los casos de prueba pasados correctamente se modifica el código para mejorarlo.

Review: En este paso un integrante del equipo el cual no tuvo participación en la codificación de la funcionalidad realiza el code review de la misma.

Por lo tanto el flujo en este caso va a ser:

Backlog → Requirements Definition (CCC) → Application Implementation → Test Cases Implementation (Automation) → Testing (Automation Tools) → Refactoring → Review → Done

Flujo de Bugs

En este caso también vamos a usar un subconjunto del tablero sustentable pero diferente al flujo de las nuevas funcionalidades ya que los requerimientos ya están definidos como también los casos de prueba. Por lo tanto vamos a aplicar los pasos Test Cases Implementation, Application Implementation y Refactoring de TDD.

El flujo en este caso va a ser:
Backlog → Test Cases Implementation → Application Implementation → Refactoring → Review → Done

Flujo de Tareas

Las tareas que no tengan que ver con nuevas user stories o bugs como por ejemplo las de documentación seguirán el flujo del tablero de la versión anterior, es decir:
Backlog → To Do → In Progress → Done

Configuración/mantenimiento del pipeline y su vínculo con el tablero

Como ya mencionamos antes para esta entrega utilizamos un tablero sustentable con BDD. Creamos todas las issues necesarias y las fuimos asociando al tablero con sus etiquetas correspondientes. La etiqueta que agregamos en esta entrega fue “user story”.

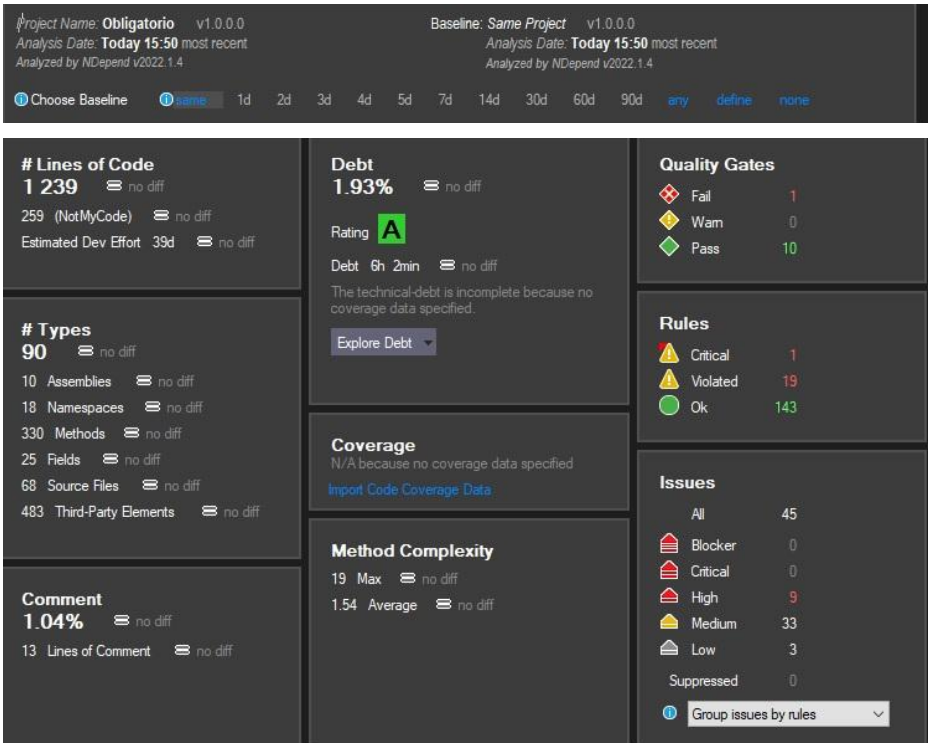
Con respecto al pipeline, con utilización de “Github Actions” creamos un pipeline para el backend para 3 versiones diferentes. Primero se instalan o restauran las dependencias, luego se realiza el build de la solución chequeando que se haga correctamente para después continuar con la corrida de tests.

También se intentó realizar para el frontend para tres versiones como en el backend pero por causa de varios problemas no logramos hacerlo funcionar, lo cual quedará para las próximas entregas.

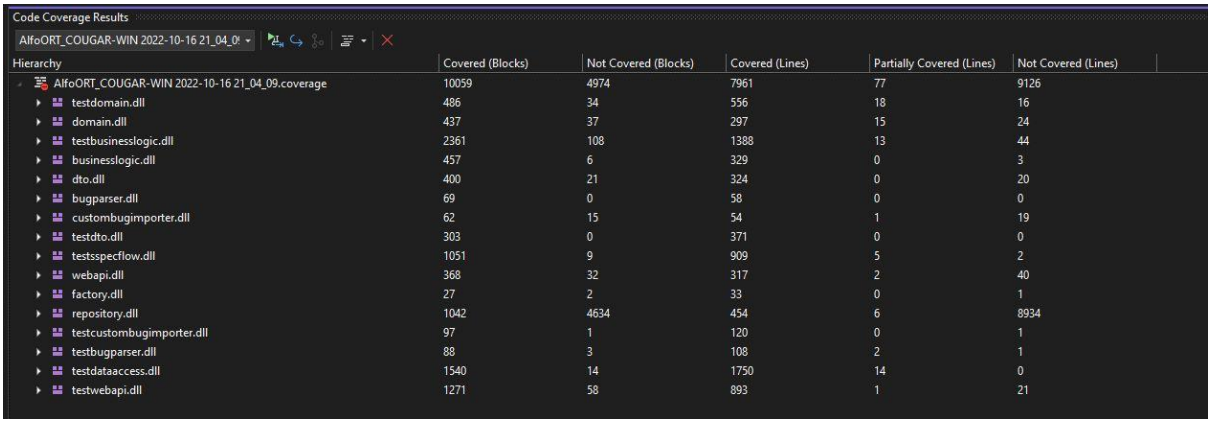
El mismo se ejecuta cada vez que se hace una pull request a develop o a main.
Se puede visualizar aquí

Analizadores de código

Inicialmente se utilizó NDepend para el análisis de código y cobertura. tuvimos diversos inconvenientes con su utilización que serán abordados en la sección de Dificultades Encontradas.



La cobertura de test por falta de tiempo no se pudo configurar la misma, decidimos verificar manualmente que al agregar las nuevas funcionalidades y el arreglo de los bugs, la cobertura no haya disminuido. Esto no lo logramos al 100% por lo que se intentará mejorar para las siguientes entregas.



Hierarchy	Covered (Blocks)	Not Covered (Blocks)	Covered (Lines)	Partially Covered (Lines)	Not Covered (Lines)
AlfoORT_COUGAR-WIN 2022-10-16 21_04_09.coverage	10059	4974	7961	77	9126
testdomain.dll	486	34	556	18	16
domain.dll	437	37	297	15	24
testbusinesslogic.dll	2361	108	1388	13	44
businesslogic.dll	457	6	329	0	3
dto.dll	400	21	324	0	20
bugparser.dll	69	0	58	0	0
custombugimporter.dll	62	15	54	1	19
testdto.dll	303	0	371	0	0
testsspecflow.dll	1051	9	909	5	2
webapi.dll	368	32	317	2	40
factory.dll	27	2	33	0	1
repository.dll	1042	4634	454	6	8934
testcustombugimporter.dll	97	1	120	0	1
testbugparser.dll	88	3	108	2	1
testdataaccess.dll	1540	14	1750	14	0
testwebapi.dll	1271	58	893	1	21

Justificación de la elección de bugs

Creación de usuarios repetidos

Al momento de analizar los Bugs, nos pareció que cada usuario del sistema tiene que tener un registro único de ingreso, ya que de esta manera se gana trazabilidad en la plantilla de funcionarios. Si bien el sistema le asignaba un id a cada usuario nuevo, este no era visible, por lo que se debería tener un procedimiento adicional para obtener dicho id y conocer la autenticidad de cada empleado. Esta regla de negocio es fundamental para la buena organización y trazabilidad de las tareas.

Borrado de los bugs fix de un proyecto al cambio de su nombre

Cuando se modifica un proyecto los bugs relacionados a este se borraban de la BD, por lo que al momento de obtener los procesos que se hicieron relacionados a dicho proyecto, se perdían, Este bug es importante solucionarlo a la brevedad ya que para el negocio, perder datos relacionados a los proyectos que se llevan a cabo es crítico, más aún cuando los empleados reciben remuneración por horas trabajadas. De perderse dicha información traería serios inconvenientes de gestión de la empresa.

Registro de esfuerzo

Para el registro de esfuerzo seguimos utilizando una tabla de Excel pero se mejoró para esta entrega y se realizó por tarea, integrante y por día registrado en minutos.

Se puede en detalle aquí, en la pestaña “Entrega 2”.

Reviews

Las reviews se hicieron con el integrante Martina Praderio como Product Owner y Rodrigo Haiache y Gabriel Lutz como los desarrolladores. En los videos se puede ver como se muestran los 2 bugs reparados y la nueva funcionalidad realizada, con el feedback recibido del Product Owner

Los videos de las reviews se pueden ver en los siguientes enlaces:
review de funcionalidad
review de bug1
review de bug2

Retrospectiva basada en DAKI

Se realizó una reunión de 12 minutos en donde se siguió la misma metodología que la retrospectiva anterior, analizando al comienzo de la misma una reflexión acerca de la mostrada en la entrega 1.

Destacamos que gracias a la primera retrospectiva, mantuvimos las tareas que se hicieron bien y en esta segunda etapa, se incorporaron a esta las correcciones de la primera, escalando así en seguridad y planeamiento.

Así mismo, observamos que debemos mejorar en la distribución de tareas y analizar el código con herramientas conocidas por nosotros, ya que se pierde tiempo valioso intentando aprender para aplicarla. de esta forma debemos agregar para la siguiente entrega la utilización de herramientas que dominamos como también prever el tiempo que nos puede tomar cada tarea para una mejor planificación y distribución de las mismas (como el uso de spikes en el tablero).

El video puede verse aquí

Nuevas funcionalidades

Las nuevas funcionalidades que se desarrollaron fueron:

- Clasificación de Bugs considerando la severidad
- Clasificación de Bugs considerando la prioridad

La funcionalidad quedó realizada solo en el backend (ya que no era un requisito hacerlo en el front).Con una pequeña deuda, por el momento un bug solo puede tener una severidad o una prioridad, pero no las 2 simultáneas (como debería ser).
Al momento de crear un bug desde un cliente Http , como por ejemplo Postman, se le puede asociar un bugType que es una entidad a la que se le asocia la severidad o prioridad.
La funcionalidad no fue hecha utilizando BDD, por lo que se abordara en la sección “Dificultades encontradas y forma de solución”

Resultados obtenidos en el periodo

Culminamos esta entrega con los bugs de las issues 23 y 19, además de la funcionalidad descrita anteriormente, la cual quedó realizada solamente en el backend. También se utilizó github actions para la implementación del pipeline, la cual no está completa por falta de distintos factores para su correcta utilización.

Dificultades encontradas y formas de solución

En la planificación inicial, se delimitaron las tareas teniendo en cuenta la experiencia y recursos de los integrantes. Se buscó tener un balance de conocimiento y software de manera de realizar la técnica de Pair- Programming separando en 2 grupos de 2 integrantes cada uno. Esta distribución no estuvo acorde con los tiempos de un equipo, ya que se demoró en el comienzo de la implementación y quedamos “dependientes” de un solo actor (que tuvo problemas particulares y personales), por lo tanto, como no se contempló los problemas que se pudieran suscitar (y que sí ocurrieron), ocasionó que no podamos terminar con las tareas de nuevas funcionalidades y se tuvo que dedicar tiempo reservado a otras tareas para finalizar las que tuvieron inconvenientes. En el futuro se tomarán los recaudos necesarios para tener tiempo para diversos problemas que puedan ocurrir.
Dedicamos tiempo en utilizar Ndepend sin previo conocimiento, por lo que nos quitó tiempo valioso para dedicarlo a otras tareas. En el futuro utilizaremos las ya conocidas.

También se encontró con el problema del uso de Specflow, el cual es una novedad en el curso y que también fue una dificultad para el integrante que realizó la nueva funcionalidad. Por su complejidad, se terminó aplicando TDD y quedó como una deuda para la siguiente entrega, donde intentaremos aplicar BDD correctamente.

Lecciones aprendidas y mejoras en el proceso

- Debemos separar las tareas de forma que las mismas sean más pequeñas y rápidas de terminar, de tal forma que si alguien tiene poco tiempo o algún inconveniente sea más fácil de ayudar o terminarla, así como también estimar el tiempo que nos insumen las mismas.
- Necesitamos ser ágiles con las herramientas de software y utilizar las que conocemos a fin de no sumar más tiempo aprendizaje del ya estipulado en la currícula de la materia.
- Tendremos que dedicarle más tiempo a las tecnologías mostradas en el curso a fin de poder aplicarlas correctamente y aprender de forma óptima.
- De haber aplicado correctamente el ciclo BDD, hubiéramos podido detectar a tiempo los problemas que ocurrieron en la implementación de las nuevas funcionalidades, además de la prolijidad y seguridad que brinda la técnica BDD en su aplicación.
- Gracias a la Retrospectiva, continuaremos mejorando en el proceso, a fin de seguir aplicando las tareas que se realizan correctamente a la fecha.