Universidad ORT Uruguay

Obligatorio - Reporte Sprint 3

Ingeniería de Software Ágil 2

Docentes:

Martin Solari

Carina Fontán

Sofía Decuadra 233397

Agustín Ferrari 240503

Joaquín Meerhoff 247096

https://github.com/ORT-ISA2-2022S1/obligatorio-decuadra_ferrari_meerhoff.git

2022

Indice

Introducción

El siguiente documento es un reporte del sprint 3 para la materia Ingeniería de Software Ágil 2, que resume el contenido encontrado dentro del repositorio[1], por lo cual aquí se da una vista general de lo trabajado que puede ser visto con más profundidad en los documentos markdown encontrados en github.

A lo largo del documento se pueden encontrar hipervínculos a los documentos completos almacenados en github en caso de que se quiera ver algún tema en mayor detalle.

Por el feedback recibido sobre el sprint 2, en este sprint se realizó un análisis de métricas en más detalle, que será detallado más adelante en este documento, además de realizar Testing Exploratorio para buscar errores en la aplicación. Por último se realizó análisis de código por el feedback, como clean code por ejemplo.

El readme del github se vió actualizado con información acerca de los nuevos archivos y donde se encuentra cada uno.

Métricas

Por el feedback recibido en clase se decidió crear un documento de métricas, ya que previamente no contábamos con un lugar centralizado en donde se analizaron las mismas.

Creemos que la adición de este documento nos ayudó a encontrar puntos de mejora en nuestro proceso de ingeniería. Por ejemplo, encontramos que deberíamos darle más importancia al sprint planning, para no tener que agregar nuevas tareas a mitad de sprint.

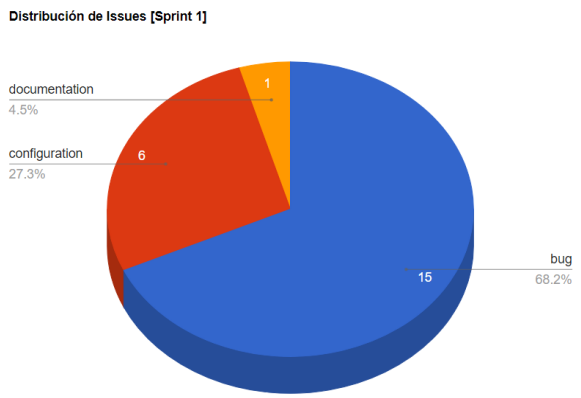
Además evaluamos métricas como Lead Time y Cycle Time pero nos dimos cuenta que no nos aportaban tanta información como esperábamos ya que una vez se termina de codificar una feature, la misma queda en espera hasta la instancia de Review que se da al final de cada sprint. Por este detalle tanto el Lead Time como el Cycle Time se veían extendidos hasta después de la Review, por lo cual las tareas que necesitaran Reviews terminarían al mismo tiempo.

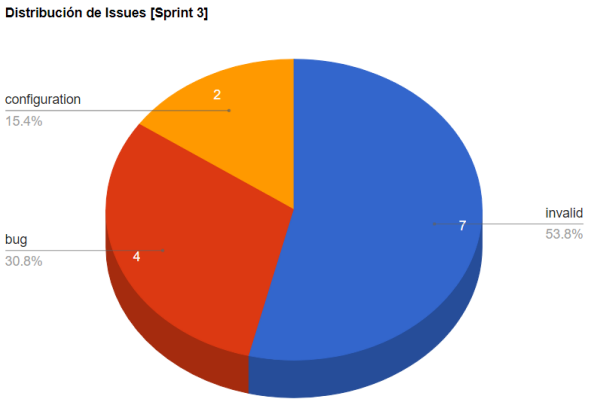
Es por esto que decidimos utilizar Lead Time to Review y Cycle Time to Review, que siguen la misma lógica que Lead Time y Cycle Time pero que terminan una vez la tarea está lista para Review. Concluimos que esta métrica es más representativa de nuestro trabajo siguiendo Trunk Based Development, ya que muestra cuando llega a main una tarea, eliminando los tiempos de espera por la review que era realizada al final o casi al final del sprint.

[1] https://github.com/ORT-ISA2-2022S1/obligatorio-decuadra_ferrari_meerhoff.git

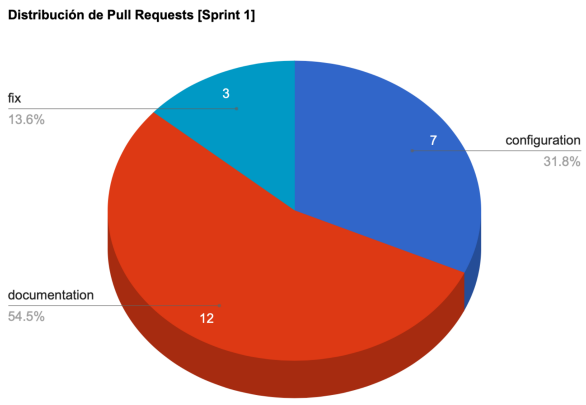
Otra de las métricas que evaluamos refieren a los issues del proyecto, para cada sprint evaluamos tres métricas de issues, la cantidad de issues junto con su estado (Abierto/Cerrado), la distribución de los issues dependiendo su categoría y la distribución de los issues dependiendo de su prioridad.

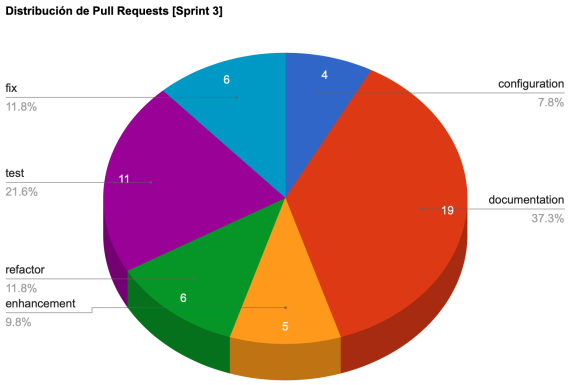
La métrica de distribución por categoría fue de gran valor para el análisis ya que evidencia los objetivos de un sprint dado con respecto a la búsqueda de errores. A simple vista nos podemos dar cuenta que en el sprint 1 nos tuvimos un mayor foco en el testing exploratorio (issues de tipo bug) que en los demás (aunque también haya sido realizado en los siguientes sprints), o que en el sprint 3[2] nos focalizamos en el análisis de código (issues de tipo invalid).





También evaluamos la cantidad de pull requests abiertos y cerrados por sprint, en donde nos encontramos con un resultado esperado, estos eran la misma cantidad. Esto se debe a que estuvimos siguiendo un desarrollo trunk-based donde cada rama que abrimos la cerrábamos en menos de un día. Además, analizamos la distribución de los pull requests, que refiere a la categoría/s que le asignamos a cada uno para ver los tipos de cambios que más se vieron reflejados en cada iteración. Particularmente, en este sprint nos enfocamos más en hacer testing y agregar nuevas funcionalidades, y con esta métrica fue fácilmente identificable.





[2] Anexo con issues del sprint 3 y link de issue filtrados de github

Esfuerzo[3]

En esta iteración se volvió a tener estimaciones muy alejadas de la realidad, al igual que en el sprint 1. Esto fue causado por subestimar el tiempo que nos llevaría el combinar las herramientas utilizadas en el sprint anterior, como SpecFlow, las herramientas de automatización de testing funcional utilizadas en la presentación de Selenium. Otro factor que contribuyó a esta extensión fue tareas que no habían sido planificadas en un principio, ya se el obtener puntos de carga que no habíamos tomado en cuenta, o que por feedback debía realizarse nuevas tareas. Todos estos factores llevaron a que se tuviera 57.55 horas de esfuerzo en ejecución comparadas a las 15 estimadas, llevando a un aumento de 280% de esfuerzo extra. En planificación notamos que subestimamos el tiempo real, manteniéndose este siempre mayor de 12 horas a lo largo del proyecto. La planificación se vió afectada por alrededor de un 40% extra de horas de esfuerzo, afectado en particular por el mayor enfoque sobre el análisis de métricas.

Todo esto implica que aún se debería mejorar las estimaciones, aunque factores externos de la materia como entregas o pruebas también pueden dificultar el predecir el esfuerzo.

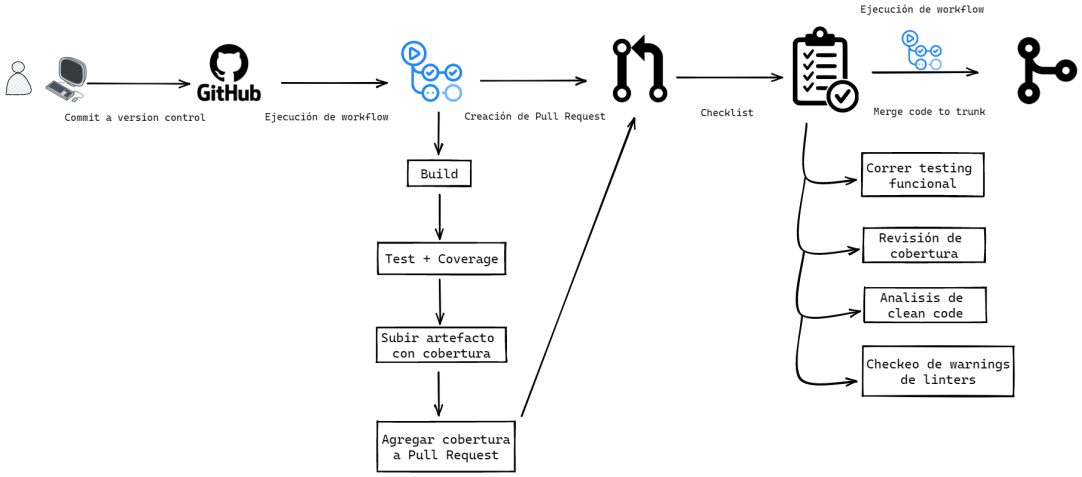
Etapa	Estimación	Real
Planifación	9 horas persona	12 horas persona
Ejecución	15 horas persona	57.55 horas persona
Control	16.5 horas persona	26.5 horas persona

[3] Tabla de esfuerzo en anexo

Resultados Obtenidos

Esta iteración mantuvo el tablero de Kanban de la iteración anterior, que incluye el Sprint Backlog, Doing (para aquellas tareas que no sigan BDD), Test Cases Implementation, Application Implementation, Refactor, Waiting for Review y Done. Se mantuvo la regla de poder agregar nuevas tareas o realizar modificaciones al tablero ya que aún estamos aprendiendo el uso correcto de estas herramientas con implementación de software.

Guia de pipeline



Por el feedback recibido en clase, se actualizó la guía del pipeline para que esta incluya aspectos por fuera de las github actions que no estaban previamente documentados. Esto implicó incluir las checklists de que se espera que se haga previó a un merge de un Pull requests: El correr el testing funcional de la solución de tests de integración, el tomar en cuenta la cobertura de código (incluyendo la reportada por la action), asegurarse de haber implementado siguiendo clean code y no agregar warnings detectadas por linters.

Desarrollo usando BDD

Se completaron las dos nuevas funcionalidades (agregar y dar de baja puntos de carga para autos eléctricos con su frontend) siguiendo la práctica de BDD al hacer automatización de testing funcional con SpecFlow y Selenium. Para esto se actualizó la guía de BDD del proyecto, haciendo una guía general y no específica al sprint como antes. Se actualizaron los escenarios de las funcionalidades para ser más generales y no tan específicos al backend. Se tuvo que agregar la funcionalidad de obtener puntos de carga para ver los cambios que uno realizaba, y al igual que las otras funcionalidades, esto fue evaluado en la Sprint Review.

Dificultades encontradas y formas de solución

En esta sección describimos problemas que encontramos y las soluciones a las que llegamos, algunas de estas están mencionadas en la retrospectiva de la iteración:

- A la hora de calcular las métricas se una dificultad fue poder diferenciar issues y pull requests por iteración y categoría a la que pertenecían. En un principio comenzamos contándolos a mano pero rápidamente nos dimos cuenta de que iba a tomar mucho tiempo. La solución que encontramos fue utilizar la herramienta de filtros de github, basándonos en la documentación de búsqueda en issues y pull requests pudimos encontrar como crear filtros incluyendo fechas de sprints, categorías y prioridades por ejemplo.
- El registrar el esfuerzo a mano se nos dificultó ya que tuvimos que estar muy pendientes de no olvidarnos y consideramos que sería más sencillo y preciso si utilizáramos alguna herramienta que nos permitiera guardar esto automáticamente. Una de las soluciones consideradas para futuras iteraciones sería Toggl.
- El asegurar la independencia entre las pruebas de integración fue un problema, ya que aspectos como la cantidad de puntos de carga presentes podían llevar a resultados diferentes. Se decidió que se eliminarían los puntos de carga previó a toda prueba que requiera algo específico de los puntos de carga. Para esto se sugirió en clase hacer uso de requests HTTP para eliminar y agregar como parte del setup de la prueba, pero ya lo habíamos resuelto eliminando los puntos de carga haciendo click en eliminar cada uno de manera automática al principio de la prueba con el Selenium WebDriver. Otra solución podría haber sido correr la base de datos en memoria para esas pruebas pero no se logró hacer.
- En esta iteración nos dimos cuenta de que el uso de hack.md para documentos grandes y trabajo colaborativo puede resultar en problemas de performance para escribir en los documentos, además también fue una dificultad mantener los documentos sincronizados con las versiones que ya estaban subidas en github.

  Encontramos dos posibles soluciones para resolver este problema, una sería activar la sincronización automática de github pero esto rompería nuestro estándar de commits.

  La solución que encontramos fue trabajar todos juntos en liveshare en el archivo md, evitando problemas de performance y sincronización ya que se trabaja sobre el repo local directamente.

Lecciones aprendidas

En esta sección describimos las lecciones que aprendimos y las mejoras en los procesos que encontramos en esta iteración:

- Aprendimos a combinar SpecFlow con Selenium, para realizar automatizaciones de testing funcional.
- Aprendimos a utilizar la herramienta de Github Copilot, que facilitó el uso de herramientas nuevas.
- Aprendimos la importancia del uso de identificadores para elementos del frontend no solo para accesibilidad sino también para el desarrollo con pruebas funcionales.
- Notamos la importancia de un estándar de codificación, para que el código sea más legible para todos los miembros del equipo.
- Aprendimos que las métricas tienen una importancia mayor a la esperada, al punto que se debería hacer uso de herramientas como Toggl para la gestión de tareas en vez de hacerlo manualmente.
- Notamos nuevamente la importancia de trabajar simultáneamente en tareas diferentes, para poder hacer uso del Andon Cord. En particular fue importante por seguir Trunk Based Development, para el cual decidimos mergear a main sin review (realizando al final del sprint) a no ser que fuese necesario.

Sprint Review

Las funcionalidades desarrolladas en frontend usando BDD fueron revisados en una instancia grabada en el video:
https://www.youtube.com/watch?v=e0VnvqfJmUA.
Este video tiene en la descripción la separación por secciones para ir directamente al minuto y segundo en el que se trata cada sección.
En general no se encontraron errores y se evaluó que las funcionalidades fueron agregadas correctamente, donde el único aspecto a mencionar es la adición no planificada en el sprint planning de la funcionalidad de obtener puntos de carga. Se decidió que al ser necesaria para el desarrollo de eliminar un punto de carga, el Product Owner de esta tarea sería Sofía ya que era PO de eliminar también. Para agregar y eliminar puntos de carga se tomó la decisión de mantener los product owners de la iteración anterior, ya que sería más acercado a la realidad que no cambien los expertos del negocio.

Anexo

Tabla de esfuerzo

Fecha	Horas persona	Integrantes	Tags	Actividad
Miércoles 25/05/22 14:00-16:00	6	Todos	Documentación, Gestión	Sprint Planning
Viernes 27/05/22 18:00-20:00	6	Todos	Documentación, Desarrollo	Charging Spot BDD Gherkin, Step Definition
Sábado 28/05/22 8:30-9:30	3	Todos	Desarrollo, Mantenimiento	Organización y diseño de frontend
Sábado 28/05/22 9:30-13:00	3.5	Agustín	Documentación, Desarrollo	Requerimientos BDD de get, backend del mismo, agregar datos de prueba a la base de datos
Sábado 28/05/22 9:30-12:00	5	Sofía y Joaquín	Desarrollo	Implementación de tests comunes de agregar y borrar punto de carga en selenium
Sábado 28/05/22 12:00-13:00	1	Sofía	Desarrollo	Implementación de tests para agregar punto de carga en selenium
Sábado 28/05/22 12:00-13:00	1	Joaquín	Desarrollo	Implementación de tests para obtener todos los puntos de carga en selenium
Sábado 28/05/22 18:45-20:00	1.2	Agustín	Desarrollo	Implementación de tests de BDD de delete charging spot
Sábado 28/05/22 18:45-20:00	1.2	Sofía	Desarrollo	Implementación de frontend para agregar punto de carga
Sábado 28/05/22 18:45-20:00	1.2	Joaquín	Desarrollo	Implementación de frontend para obtener todos los puntos de carga
Domingo 29/05/22 11:00-12:00	1	Sofía	Desarrollo	Implementación de frontend para agregar punto de carga
Domingo 29/05/22 14:30-15:40	1	Joaquín	Documentación, Desarrollo	Implementación de frontend para borrar puntos de carga, arreglos de gherkin de get y delete
Lunes 30/05/22 17:15-19:00	3.6	Agustín y Sofía	Desarrollo	Pruebas de integración de BDD con Selenium de agregar
Lunes 30/05/22 17:15-19:00	1.8	Joaquín	Desarrollo	Pruebas de integración de BDD con Selenium de eliminar y obtener
Lunes 30/05/22 19:00-20:00	2	Agustín y Sofía	Desarrollo	Pruebas de integración de BDD con Selenium de agregar
Martes 31/05/22 15:00-16:15	1.2	Sofía	Desarrollo	Pruebas de integración de BDD con Selenium de agregar

Martes 31/05/22 15:00-16:15	2.4	Joaquín y Agustín	Desarrollo	Pruebas de integración de BDD con Selenium de eliminar y obtener
Martes 31/05/22 17:30-19:30	6	Todos	Desarrollo	Pruebas de integración de BDD con Selenium
Martes 31/05/22 19:30-20:30	2	Agustín y Sofía	Desarrollo	Pruebas de integración de BDD con Selenium de agregar
Jueves 2/06/22 17:10-20:30	3.3	Joaquín	Desarrollo, Mantenimiento	Pruebas de integración de BDD con Selenium de eliminar, arreglar warnings de linters
Jueves 2/06/22 17:10-19:00	2	Sofía	Desarrollo	Pruebas de integración de BDD con Selenium
Jueves 2/06/22 17:10-20:30	3.3	Agustín	Documentación, Mantenimiento	Actualización de guía de pipeline, template de pull requests y checklist
Jueves 2/06/22 19:00-20:30	1.5	Sofía	Desarrollo	Arreglar pruebas de BDD sin Selenium
Viernes 3/06/22 13:15-14:00	2.25	Todos	Mantenimiento	Actualizar template de effort
Viernes 3/06/22 14:00-15:00	1	Agustín	Desarrollo	Analizar si se sigue Clean Code en el código
Viernes 3/06/22 14:00-15:30	1.5	Joaquín	Documentación	Documentar estándares de programación
Viernes 3/06/22 14:00-15:00	1	Sofía	Mantenimiento	Arreglar pruebas de BDD sin Selenium
Viernes 3/06/22 18:30-19:20	0.8	Joaquín	Documentación	Arreglar definición de requerimientos de Sprint 3
Viernes 3/06/22 19:20-20:00	1.3	Joaquín y Agustín	Mantenimiento	Reporte de issues a github y actualización de deuda tecnica
Viernes 3/06/22 20:00-21:00	2	Joaquín y Agustín	Documentación	Análisis de métricas
Viernes 3/06/22 18:30-21:00	1.5	Sofía	Mantenimiento	Arreglar pruebas de BDD sin Selenium
Sábado 4/06/22 10:50-12:50	6	Todos	Documentación	Análisis de métricas
Sábado 4/06/22 13:45-15:45	3	Todos	Documentación	Análisis de métricas
Sábado 4/06/22 17:00-18:10	3.5	Todos	Documentación, Gestión	Sprint Review
Sábado 4/06/22 19:00-21:00	6	Todos	Documentación, Gestión	Sprint Retrospective
Domingo 5/06/22 19:00-21:00	6	Todos	Documentación, Gestión	Sprint Report

Issues Sprint 3

[Link a búsqueda filtrada de issues creados en el sprint 3.]