Universidad ORT Uruguay

Facultad de Ingeniería

Ingeniería de software ágil 2

Documentación de la entrega 4

https://github.com/IngSoft-ISA2-2023-2/obligatorio-grupo11-bandera-caffarena-gonzalez

Sebastián Bandera-220417

Martín Caffarena-230351

Nicolas Gonzalez-231843

Entregado como requisito de la materia Ingeniería de software ágil 2

Designación de roles

De igual forma que se realizó para las etapas anteriores, volveremos a rotar los diferentes roles para esta etapa, en este caso contaremos con Sebastian como el SM, y a Martin como el PO. En esta etapa no hicimos distinción de desarrolladores front y back como la anterior, sino que de igual forma que se realizó en la etapa 1 y 2 todos los miembros del equipo son desarrolladores y testers.

Proceso de ingeniería

A lo largo de las tres etapas del proceso de ingeniería, se han implementado mejoras significativas para optimizar la planificación, desarrollo y entrega de software. A través de un enfoque iterativo y colaborativo. El equipo ha adaptado su metodología y herramientas para abordar los desafíos específicos de su proyecto. A continuación, hablaremos de los procesos de ingeniería de cada etapa del proceso de ingeniería, conectadas por los ajustes y evaluaciones realizados durante el proyecto.

Etapa 1:

Las etapas del proceso en esta iteración son planificación, análisis de código y test.

La etapa de planificación se realizará por el SM con apoyo del resto de integrantes y generará las tarjetas en el tablero Kanban, columna “backlog”, luego de revisar los requisitos para la iteración.

La etapa de análisis de código la realizará el rol de desarrollador que tiene entendimiento sobre el código y genera el registro de las issues en GitHub, las cuales a su vez generan una tarjeta en el Backlog. Este análisis se realiza con una herramienta para optimizar los tiempos de trabajo

La etapa de Test la realiza el tester ejecutando la aplicación y comparando el comportamiento contra lo especificado en la letra original. También produce issues y tarjetas asociadas en el Backlog. También se usa test exploratorio para buscar errores generales como permitir cantidades negativas, precios negativos.

Vemos como de la etapa 1 a la etapa 2 el equipo adoptó una nueva estrategia de gestión de ramas y commits, pasando de un enfoque tradicional a un método "Trunk based". Esto permitió un mayor control sobre los cambios realizados y una integración más fluida de las contribuciones individuales al proyecto.

Etapa 2:

Como únicas actualizaciones al proceso de ingeniería tenemos que en esta etapa se definió la metodología para el manejo de ramas/commits y trabajo del equipo y se reestructuró el tablero de Kanban.

Para esto el equipo decidió utilizar el método de Trunk based, donde todos trabajamos en main y se comitea sobre el mismo, para luego ser corrido el Pipeline automático. Debido a que no se trabajó en el proyecto todos los días, debido a el tamaño de la entrega se decidió el hacer commit luego de cada día de trabajo individual, por lo que no se commitea a main todos los días sino solo aquellos en los que el desarrollador trabajó.

Luego en la etapa 3 se introdujo la metodología de BDD, con un enfoque en la definición de historias de usuario y escenarios. Esto marcó un cambio significativo en cómo se abordaba el desarrollo de nuevas funcionalidades y pruebas. Además, se mantuvo la estrategia de gestión de repositorio "Trunk based", pero con una mayor atención a la estabilidad antes de realizar "push" a la rama principal.

Etapa 3:

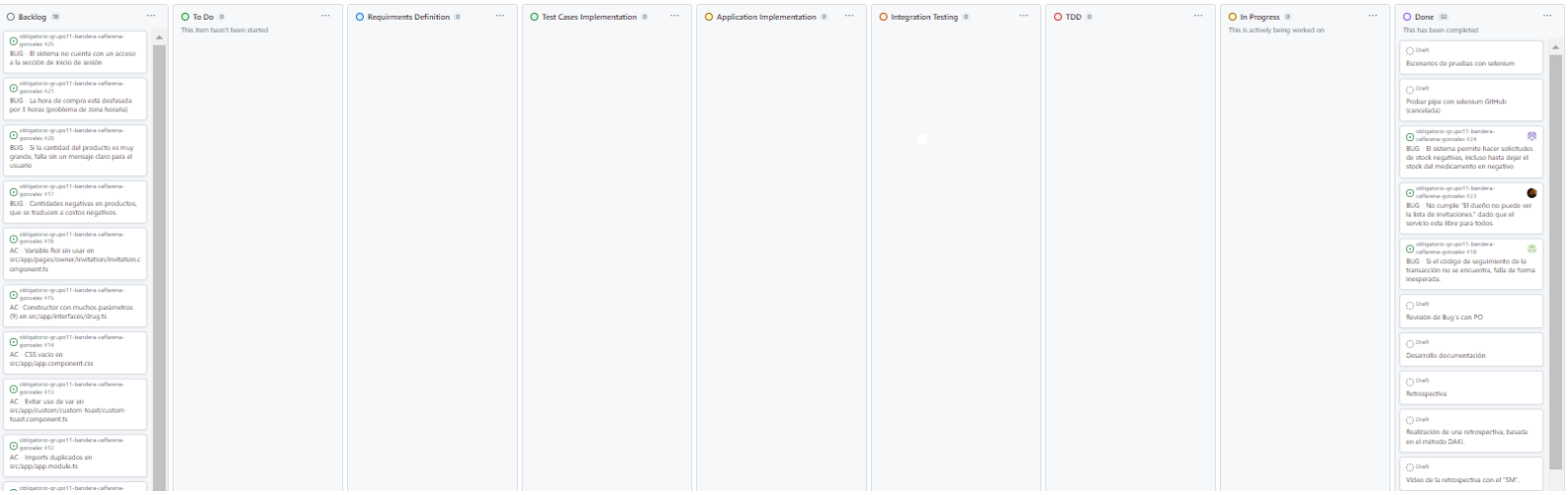
Para la implementación de las nuevas funcionalidades se utilizó la metodología de BDD, la cual el desarrollo de las mismas son guiados por los comportamientos y para hacer uso de la misma se utilizó la herramienta de SpecFlow, en la cual se definió la narrativa de las historias de usuario, en nuestro caso cada funcionalidad tenía su propia historia de usuario porque nos pareció que quedaba con mayor claridad si se manejaba de esta manera, y también los escenarios de las mismas. Dentro de BDD identificamos la fase RED, que al igual que en TDD corresponde a la creación de los casos de prueba, en este caso la narrativa con sus escenarios, pero sin implementar la misma, luego la fase GREEN donde se implementa el código para que pasen todos los escenarios dentro de las pruebas de la historia de usuario correspondiente y por último la fase de REFACTOR, la cual no siempre está presente pero corresponde a hacer cambios sobre el código y las pruebas para mejorar estos.

Para el manejo del repositorio utilizamos el método de trunk-based, definimos que al final de cada momento/día donde los desarrolladores hubiesen realizado cambios sobre el proyecto y además estos estén funcionales, se trata de no “pushear” a main cualquier cambio que no se encuentre estable para así no perjudicar al resto de desarrolladores, los mismos pushean todo lo realizado al repositorio y describen los cambios. A pesar de que siempre se trató de mantener este flujo de trabajo en algunas instancias hubo que “pushear” algunos cambios antes de terminar de trabajar, porque se estaba trabajando en paralelo entre los desarrolladores y en algunas circunstancias se precisaba tener un cambio en el cual estaba trabajando otro desarrollador.

La pipeline utilizada se mantuvo debido a que ya es capaz de ejecutar todas las pruebas incluidas las de SpecFlow. Además está optimizada para sólo ejecutar la pipeline de Front o Back según corresponda revisando los archivos modificados del commit.

A lo largo de estas etapas, el equipo mantuvo la práctica de utilizar un tablero Kanban para la gestión de tareas y la colaboración. El uso constante del tablero Kanban sirvió como una herramienta clave para la planificación y el seguimiento de tareas, facilitando la implementación de la metodología BDD en la Etapa 3.
En resumen,en estas etapas del proceso de ingeniería se ha adaptado y ha evolucionado de forma continua en conjunto con la metodología y herramientas utilizadas para abordar los desafíos específicos del proyecto. Desde la planificación hasta la implementación de nuevas funcionalidades, el equipo ha demostrado un enfoque flexible y colaborativo para lograr sus objetivos.

Tablero KANBAN



En esta entrega, el tablero KANBAN será una combinación de los tableros anteriores como si el proyecto comenzará de nuevo con esta entrega pero integrando el trabajo de las anteriores en conjunto. Las columnas del nuevo tablero combinado en orden son:

- **Backlog**: Utilizada como almacén inicial de las tarjetas. Puede tener tarjetas que no trabajarán en la entrega actual. Decidimos que todas las tarjetas deben encontrarse aquí para mayor transparencia del proyecto, pudiendo ver el trabajo que queda por delante.
- **To Do**: Contiene las tarjetas habilitadas para trabajarlas en la entrega actual.
- **Requirements Definition:** Columna para las historias de usuario asociadas a nuevos requerimientos, mientras estos se están definiendo. El próximo paso de una tarjeta ubicada aquí, es el Test cases implementation. Proviene de la entrega 3.
- **Test Cases Implementation:** Columna que refleja que se están escribiendo los escenarios para la historia de usuario de nuevo requerimiento. El próximo paso de una tarjeta ubicada aquí, es el Application implementation
- **Application Implementation:** Esta columna refleja que el requerimiento asociado a la tarjeta, está siendo implementado a nivel de código para pasar sus pruebas en RED a GREEN. El próximo paso de una tarjeta ubicada aquí, es el Integration Testing
- **Integration Testing:** Columna para realizar las pruebas de integración con el FrontEnd y revisión de la pipeline en GitHub. El próximo paso es el Done.
- **TDD:** La resolución de bugs se realizó con TDD, así que este paso se utiliza para estos. El siguiente paso es el Done.
- **In progress**: Columna dedicada a aquellas tarjetas asociadas a una tarea puntual. El próximo paso de una tarjeta ubicada aquí, es el Done. Proviene de la entrega 1.
- **Done:** Punto final de los flujos de tareas, bugs y funcionalidades.

En resumen los flujos de las tarjetas son los siguientes dependiendo que representan las mismas:
Bugs: To Do - TDD - Done
Tareas: To Do - In Progess - Done
Historias de usuario: To Do - Requirements Definition - Test Cases Implementation - Application Implementation - Integration Testing - Done

En la entrega 2 se utilizaron las columnas “Design” para reflejar el diseño e implementación de la prueba, y “To test” para reflejar la prueba por los testers y pipeline. En esta entrega entendemos que es mejor que el flujo para los bugs pase del “ToDo” a “TDD” para luego finalizar en el “Done” para reflejar mejor el uso de TDD.

Informe de avance de la etapa

Luego de la designación de roles rotados con respecto a la entrega anterior, se procedió a realizar las pruebas con la herramienta selenium sobre la mayoría de los escenarios especificados con BDD. Se encontraron dificultades al trabajar con selenium, entre ellas problemas con la grabación, falta de identificadores, errores por velocidad de ejecución y pipeline automatizada en GitHub. Estas dificultades se describen en los siguientes subcapítulos.

Luego se realizaron cálculos sobre los registros de esfuerzo de las entregas pasadas para elaborar varias métricas de Kanban, a nivel de actividad, de entrega y global.

Dificultades encontradas y formas de solución.

Selenium:

Una de las primeras dificultades con las que se encontró el equipo fue con el uso de la herramienta Selenium para la realización de las pruebas de front, que si bien se había visto la utilización de la misma en clase ninguno de los miembros del equipo la había utilizado personalmente, por lo que nos encontramos con diferentes dificultades a lo largo de su uso, especialmente por falta de experiencia con la misma, donde con un poco de investigación todo fue solucionado.

Problemas:

a) Diferencia de velocidad en la ejecución IDE vs consola: Esta diferencia provoca que al realizar la grabación de los pasos con el IDE y la prueba sobre el mismo, no quede claro en qué momentos se está dependiendo de un elemento que aparece milisegundos después de la carga de la página (ejemplo los mensajes de información). Entonces poder ejecutar con éxito todas las pruebas en el IDE, no asegura que se ejecuten correctamente por comando. Se necesitó colocar “wait” y “pause” en algunos casos para evitar estos problemas
b) Identificadores faltantes: En algunos puntos, para evitar una gran complejidad en la selección de los campos, fue necesario agregar identificadores al frontend para poder identificar inequívocamente ciertos campos.

Métricas:

Por otro lado otra gran parte de esta etapa era el análisis de las etapas, si bien el equipo contaba con una detallada bitácora con información registrada de las anteriores etapas y todas las métricas eran conocidas por el grupo, a la hora de hacer el análisis de las mismas se discutió sobre el significado y razón de los resultados obtenidos, ya que hasta el dia de hoy nunca habíamos tenido que realizar un análisis de un proyecto en el que hubiésemos participado y que fuese tan grande, sino que siempre habia sido a través de ejercicios en clase. Sin embargo como se puede apreciar en el apartado de análisis de métricas esto no fue una barrera que nos detuviese de sacar valiosa información de las mismas.

Pipeline con Selenium:

Finalmente otro problema con el cual se enfrentó el equipo pero no logro solucionar face el uso de Pipeline con Selenium, donde el equipo no logro hacer que el Pipeline corriese de forma exitosa las pruebas del Selenium, por razones de tiempo se decidió no invertir más tiempo de momento en la investigación y resolución del error y se tratará con este en futuras etapas.
El primer problema encontrado fue de compatibilidad con node 16 utilizado. Este problema se pudo resolver pasando a node 18. Sin embargo, aunque se colocó en el yml la instalación de chrome, dependencias y la ejecución de los test como se muestra a continuación

```
 - name: Setup Chrome for Selenium
        uses: browser-actions/setup-chrome@v1.2.3
      - name: Install Dependencies Selenium
        run: |
          cd Codigo/Frontend
          npm install -g selenium-side-runner
          npm install -g chromedriver
      - name: Test Selenium
        run: |
          cd Codigo/Frontend
          selenium-side-runner PharmaGo.side
```

cada test lanzaba lo siguiente

```
SessionNotCreatedError: session not created: Chrome failed to start: exited normally.
    (session not created: DevToolsActivePort file doesn't exist)
    (The process started from chrome location /usr/bin/google-chrome is no longer running,
so ChromeDriver is assuming that Chrome has crashed.)

    at Object.throwDecodedError (node_modules/selenium-webdriver/lib/error.js:524:15)
    at parseHttpResponse (node_modules/selenium-webdriver/lib/http.js:601:13)
    at Executor.execute (node_modules/selenium-webdriver/lib/http.js:529:28)
```

Se consumieron un par de horas para intentar resolverlo, pero se decidió no continuar ya que aunque se resolviera, ya no quedaba tiempo para levantar en la máquina virtual el backend, con su base de datos precargada de prueba.

Nuevos aprendizajes

Como ya mencionamos en el apartado anterior de dificultades encontradas, Selenium y el análisis de las métricas eran dos cosas que el equipo no había utilizado/realizado por lo que tuvo que aprender. Primero que nada a utilizar esta nueva herramienta y poder realizar pruebas que aporten valor y seguido a analizar de forma correcta las métricas de forma de obtener información valiosa de las mismas, también se debió aprender bastante más sobre el funcionamiento del pipeline en GitHub, a pesar de no haber podido resolver los problemas del mismo con Selenium el equipo considera que ahora se encuentra con un conocimiento mucho más capaz del mismo.

Registro de las actividades realizadas (fecha, horas, integrante)

***Se utilizó como período de tiempo mínimo, 30 minutos = 0,5 horas**

Fecha	Concepto	Horas Bandera	Horas Caffarena	Horas Gonzalez	Total esfuerzo	Total día
24/10/2023	Pruebas con selenium	1,5	0,0	0,0	1,5	1,5
28/10/2023	Pruebas con selenium - compras	2,0	0,0	0,0	2,0	2
29/10/2023	Reunión sobre estructura de documentación	2,5	2,5	1,5	6,5	
29/10/2023	Probar pipe con selenium GitHub	1,0	0,0	0,0	1,0	
29/10/2023	Más escenarios con selenium	5,5	0,0	0,0	5,5	
29/10/2023	Documentación	1,0	1,5	0,0	2,5	
29/10/2023	Análisis para métricas	0,0	2,0	0,0	2,0	17,5
30/10/2023	Reunión final	0,5	0,5	0,5	1,5	
30/10/2023	Grabación reto y reunion PO	1,0	1,0	1,0	3,0	4,5
	Total Entrega 4	0,0	0,0	0,0	0,0	25,5

Horas totales para entrega 4: 25,5 horas persona

Métricas tomadas

A continuación hablaremos sobre las métricas tomadas para este proyecto, en este caso nos basaremos en los tiempos registrados por cada miembro del equipo en la bitácora o “tabla de registro de actividades realizadas”, la cual cada integrante era responsable de ir registrando su trabajo. Indicando fecha, asunto tratado y tiempo trabajado. Si bien se podría haber optado por utilizar la información en los tableros, tras un breve análisis de parte el equipo se decidió por el uso de la bitácora ya que esta recaudaba mayor información de forma más precisa. Tal vez si el equipo hubiese llevado un registro más riguroso del lado del tablero y no solo concentrado en la bitácora se podría haber utilizado estos como fuente de información, por ejemplo nos hubiese gustado el identificar las tarjetas del tablero por tareas, bugs, user Stories, etc. Entendemos que hubiese sido lo más correcto, pero por falta de conocimiento sobre esta herramienta del github el equipo no supo hacerlo y considero que sería más sencillo llevar un registro más riguroso en la bitácora. Aclarar que para el

cálculo de cada métrica se utiliza como unidad los días y no las horas, sin embargo para el cálculo de cantidad de horas por tarea si se utilizó como medida las horas registradas en la bitácora. A su vez debemos remarcar que no todas las actividades se tuvieron en cuenta para este análisis, actividades como reuniones (retro, reunion de org, reuniones con PO) no se tuvieron en cuenta, además otras actividades como creación de documentación tampoco fueron tenidas en cuenta para el cálculo de las métricas, esto porque el equipo no las considero necesario y/o importante el considerarlas.

Las métricas que mediremos serán las siguientes:

- Lead Time
- Cycle Time
- Esfuerzo de una tarea
- Flow efficiency
- Throughput

Entrega 1	Actividad	Lead Time (En Dias)	Cycle Time (En Dias)	Esfuerzo de una tarea (En horas)	Flow efficiency
Entrega 1 (04/09/23 - 18/09/2023)	Análisis de deuda técnica	13	2	4,5	15,38%
Entrega 2 (18/09/2023 - 27/09/2023)	Creación de pipelines backend y frontend	5	1	2	20,00%
	BUG Nro 22	7	1	2	14,29%
	BUG Nro 18	7	1	1	14,29%
	BUG Nro 24	9	1	1	11,11%
Entrega 3 (27/09/2023 - 23/10/2023)	Creacion nuevo producto	23	1	5	4,35%
	Creacion y modificacion nuevo producto FrontEnd	23	1	3,5	4,35%
	Escenarios compra producto	24	1	0,5	4,17%
	BDD compra productos	24	1	3	4,17%
	Eliminacion producto	24	1	3	4,17%
	Cambios de dominio	24	1	1,5	4,17%
	Listados de producto	25	1	5,5	4,00%
	Eliminacion producto y compra con productos FrontEnd	26	1	5,5	3,85%
	Edición de producto	26	1	2	3,85%

Imagen 1 - Métricas por actividad

Entrega 1	Throughput	Lead Time Promedio (En Dias)	Cycle Time Promedio (Dias)	Esfuerzo promedio de una tarea (En horas)	Flow efficiency
Entrega 1 (04/09/23 - 18/09/2023)	0,07	13,00	2,00	4,50	15,38%
Entrega 2 (18/09/2023 - 27/09/2023)	0,44	7,00	1,00	1,50	14,29%
Entrega 3 (27/09/2023 - 23/10/2023)	0,35	24,33	1,00	3,28	4,11%

Imagen 2 - Métricas por entrega

Metricas de todas las entregas	
Lead Time Promedio (En Dias)	14,77
Cycle Time Promedio (Dias)	1,30
Esfuerzo promedio de una tarea (En horas)	3,00
Flow efficiency	11,27%
Throughput	0,29

Imagen 3 - Métricas en todas las entregas

Análisis de Métricas

Primero que nada algo que destaca a la vista en los resultados de nuestras métricas es el Lead Time que obtuvimos a lo largo de las 3 entregas, esto sin embargo es fácilmente entendible, de forma sencilla de explicar esta métrica lo que nos indica es el tiempo que las actividades demoran en pasar de el TO_DO a DONE, es decir, desde que comenzó cada etapa hasta que finalizamos cada tarea. Por lo que tan alto valor es esperable pues, por diferentes razones como compromisos con otras materias de los miembros del equipo, cada etapa no era iniciada de forma inmediata, sino que recién era comenzada sobre las fechas finales de la misma. El equipo entiende que esto no es lo mejor para el desarrollo de un proyecto, sin embargo, como mencionamos rara vez el equipo contaba con disponibilidad para comenzar previamente. Esto se nota más en la última entrega en donde al contar con más tiempo, el equipo se tardó más en comenzar y por ende obtuvimos un lead time muy grande.
Sin embargo hace falta aclarar que como se puede ver en la siguiente métrica, el Cycle time, es decir cuánto se demoraba en terminar una actividad una vez esta se iniciaba, vemos como prácticamente todas las actividades son terminadas de forma inmediata no llevando más de un dia, por lo que el equipo si estaba trabajando de una forma efectiva. Llevando cada tarea un promedio de 3 horas, lo cual consideramos en un buen valor. Tener en cuenta que si bien se intentó a la hora de definir tarea/actividades mantener un criterio de definición de las mismas de forma que todas sean del mismo tamaño, es decir ninguna tarea muy grande o corta en relación al resto, hay algunas tareas que eran más “costosas” / grandes que otras por lo que vemos como varían las horas de trabajo necesarias para cada una. Entendemos sin embargo que esto mejoraría con la realización de mas interacciones, de forma que el equipo se conozca más a sí mismo y sea capaz de determinar de forma más acertada la duración de cada una.
Si continuamos con el análisis de las métricas, llegamos a el Flow Efficiency, el cual en pocas palabras, nos indica que tan eficientemente circulan las tareas por nuestro tablero. Lo recomendado sería el haber obtenido valores por encima del 5% para todas las etapas o incluso llegando por encima de un 40% considerándose esto un valor muy bueno. Sin embargo si bien el equipo logró esto llegando a +14% en la entrega 1 y 2, podemos ver como el no se logró llegar a este 5% en la entrega 3. Pero nuevamente si consideramos que dicha métrica se calculó usando la siguiente fórmula, $cycle\ time\ /\ lead\ time$, vemos que el lead time juega un enorme papel en el resultado de la misma y nuevamente al haber demorado con el inicio del trabajo en cada etapa y haber obtenido valores tan grandes de Lead Time. Esto destaca como mencionamos en esta última entrega por su larga duración. Igualmente creemos fuertemente en el equipo que si se hubiese tomado una iniciativa de comenzar a trabajar de forma inmediata en cada etapa, considerando nuestro Cycle time, hubiésemos tenido resultados muy buenos de parte de flow efficiency.
Finalmente tenemos el throughput, el cual nuevamente vemos valores bajos para las 3 etapas, en especial la primera, pero nuevamente si analizamos las actividades que fueron tomadas en cuenta y el tiempo que se contaba para cada etapa, vemos que esto es fácilmente explicable. Por un lado no todas las actividades fueron tenidas en cuenta, como mencionamos, como por ejemplo las preparaciones de los ambientes de desarrollo o la creación de las documentaciones, cosas que formaron un gran rol en cada etapa, especialmente en la primera , donde solo se tuvo una sola actividad en cuenta. Ya que el resto de actividades constaban de reuniones de organización del equipo y otras, como el preparar el proyecto, levantar la base de datos, etc. Al tener en cuenta lo recién

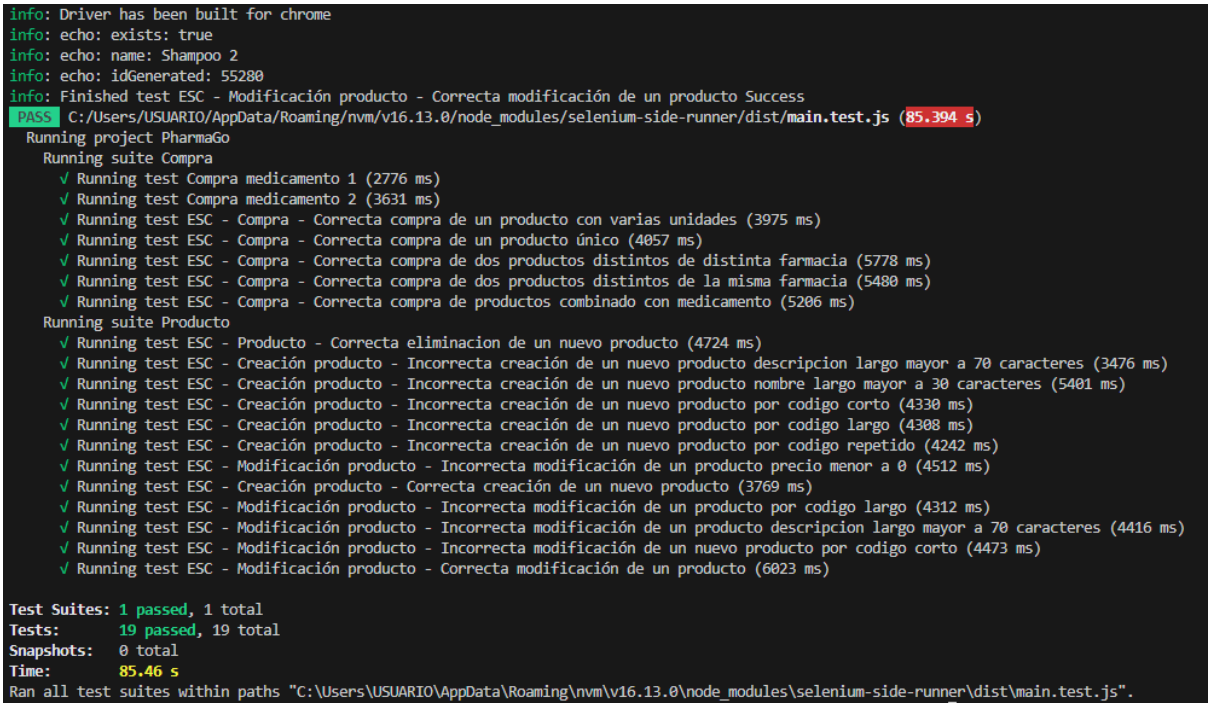
mencionado, la larga duración de las etapas y que calculamos throughput, como la cantidad de actividades realizadas sobre tiempo disponible llegamos a estos valores. Por lo tanto es totalmente esperable, aclarar sin embargo que no consideramos que la duración de las etapas fuese incorrecta ya que todos los miembro del equipo cuentan con otras obligaciones y esto es solo una manera de aprender y no un proyecto 100% real en donde se trabaja cada día.

Evidencia de ejecución de casos de prueba

Se ejecutó y grabó en un video el proceso de prueba automática con SeleniumIDE. Se encuentra en Documentación/Entrega 4 - ISA2/EvidenciaSeleniumIDE.mkv
El link directo es:
https://github.com/IngSoft-ISA2-2023-2/obligatorio-grupo11-bandera-caffarena-gonzalez/blob/main/Documentaci%C3%B3n/Entrega%204%20-%20ISA2/EvidenciaSeleniumIDE.mkv

Tener en cuenta que los test automáticos prueban no sólo casos correctos, sino que también prueban casos incorrectos para ver que despliegue el mensaje de error. Así que se podrán ver en algunos casos mensajes de error.

La captura de la ejecución por comando es la siguiente:



Se definieron dos suites de pruebas, una para compras y otra para producto. La suite de compra abarca un par de compras de medicamentos y varios escenarios definidos en la especificación BDD. La suite de producto tiene escenarios definidos en la especificación BDD para la baja, alta y modificación. Se excluyeron algunos escenarios dado que son imposibles de reproducir desde el frontend, porque este ya lo controla. Por ejemplo el escenario ‘Scenario: Incorrecta eliminación de un nuevo producto Given que se ingresa un id de un product 12345 que no existe’, como el frontend lista los productos eliminables, un

producto que no existe no estará listado, y por tanto no tendrá un botón de borrado asociado

Video de la revisión de las nuevas funcionalidades con el PO

El vídeo de la revisión con el PO, se puede encontrar en el repositorio en GitHub, rama main, directorio “Documentación/Entrega 4 - ISA2”. Contiene la revisión de las nuevas funcionalidades de la aplicación

El link directo es:
https://github.com/IngSoft-ISA2-2023-2/obligatorio-grupo11-bandera-caffarena-gonzalez/blob/main/Documentaci%C3%B3n/Entrega%204%20-%20ISA2/RevisionConPO.mkv

Video de retrospectiva

El vídeo de la retrospectiva se puede encontrar en el repositorio en GitHub, rama main, directorio “Documentación/Entrega 4 - ISA2”.

El link directo es:
https://github.com/IngSoft-ISA2-2023-2/obligatorio-grupo11-bandera-caffarena-gonzalez/blob/main/Documentaci%C3%B3n/Entrega%204%20-%20ISA2/Retro4.mkv

Tablero DAKI

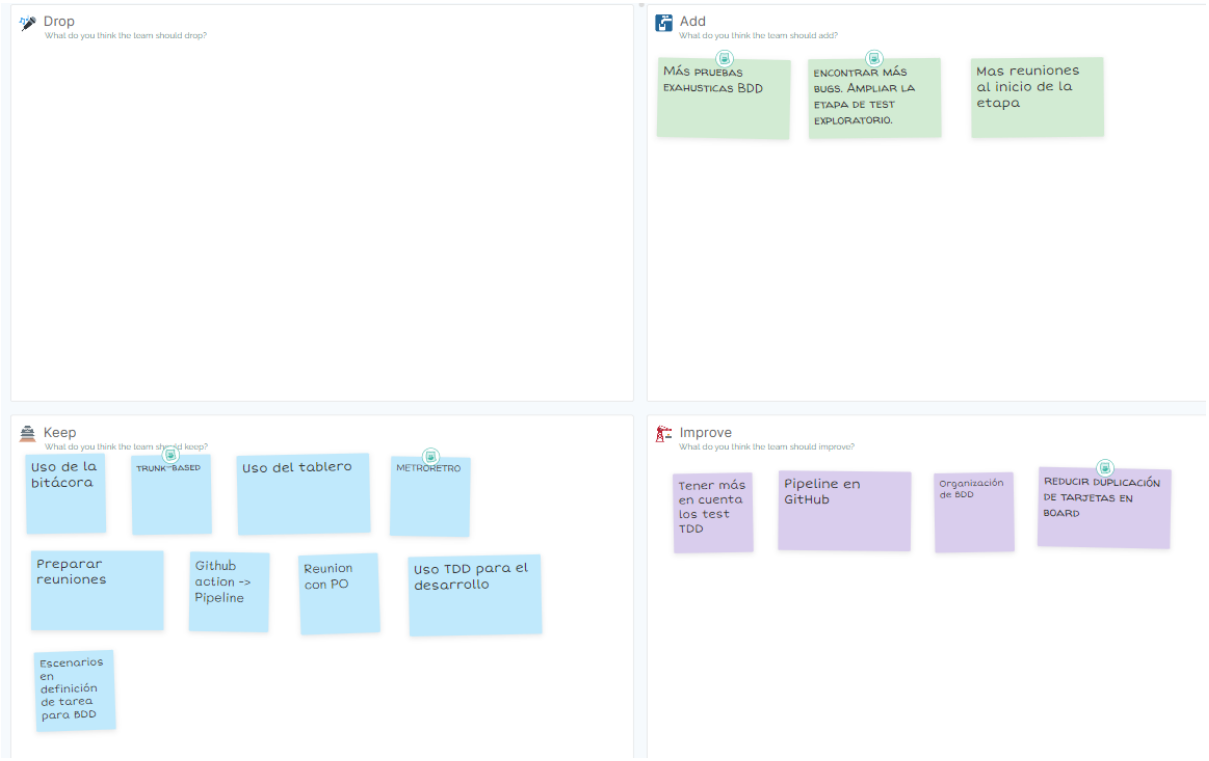


Imagen 1 - Tablero DAKI

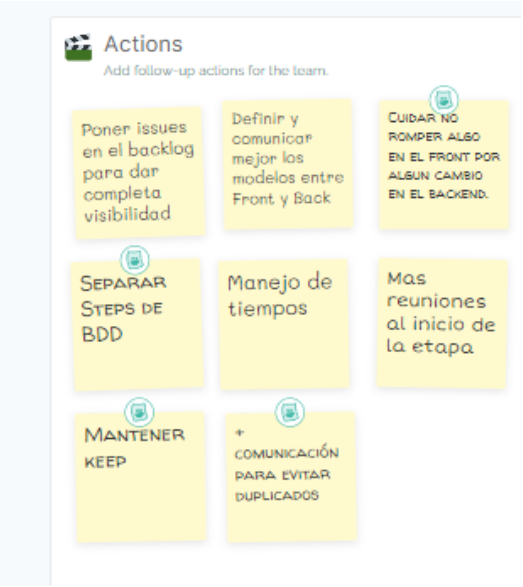


Imagen 2 - Tablero DAKI

Para esta etapa al entender que se trata de un conjunto de las tres etapas anteriores, se decidió el juntar todos los datos recolectados en retros anteriores y se juntó todo en un único tablero, donde podemos ver todo lo hablado en cada retor , como cosas a mantener, mejorar y agregar, así como los aprendizajes del equipo y las actions que fueron tomadas y si esto hubiera sido una única etapa, todo lo que se tendría en cuenta para un futuro.