Universidad ORT Uruguay
Facultad de Ingeniería

Ingeniería de Software Ágil II
Informe académico final

Martina Cantera - 256233

Facundo Diaz - 263783

Juan Toledo - 222371

Docentes: Alvaro Ortas
Carina Fontán

2023

Link a repositorio:
https://github.com/IngSoft-ISA2-2023-2/obligatorio-cantera-diaz-toledo

Resumen de gestión de proyecto

Entrega 1: Proceso con Kanban

Para esta entrega realizamos un análisis sobre la deuda técnica del proyecto que se nos entregó. El análisis sobre la calidad del código se logró subdividiendo esta tarea en el análisis del código backend, el análisis del código frontend, bugs del frontend y bugs del backend.

Al realizar el testing exploratorio encontramos diversas fallas en la aplicación que fueron reportadas como Issues. También se realizaron las siguientes ceremonias : Retrospectiva para reflexionar sobre cómo trabajamos en la entrega, Stand-up semanal para mantener al equipo informado y alineado con los objetivos del proyecto, y llevamos cabo las actividades propias de la entrega como generar el registro de horas, creación del tablero, definición del proceso de ingeniería y el informe de avance. El equipo obtuvo como resultado de lo anterior un análisis profundo respecto a la deuda técnica con un respectivo reporte, clasificación y descripción de los Issues encontrados.

Cantidad de issues reportados y cerradas:

En esta entrega se reportaron 14 Issues, pero no se cerró ninguno.

Link al informe de avance de la entrega:

https://github.com/IngSoft-ISA2-2023-2/obligatorio-cantera-diaz-toledo/blob/Develop/Entregas/Entrega%201/Informe%20de%20avance/InformeDeAvance.pdf

Entrega 2: Redefinición y Bugs

En la segunda entrega, redefinimos nuestro proceso debido a cambios en los requisitos. Para esta entrega, además de llevar a cabo las ceremonias, realizamos actividades de codificación y testing.

Nos enfrentamos a la reparación de tres bugs, utilizando el Desarrollo Dirigido por Pruebas (TDD) para bugs de backend y realizando cambios directos para los bugs de frontend. Destacamos la importancia de la adaptabilidad y la necesidad de ajustar ágilmente nuestro proceso.

También nos enfrentamos al desafío de crear el pipeline, esta tarea fue la que nos llevó más tiempo ya que nunca habíamos creado un pipeline. Pero al quedar funcionando nos permitió asegurarnos que todo el código que quede deployado pueda ser compilado y los test pasaran todos.

Cantidad de issues reportados y cerradas:

Habían reportados 14 Issues, logramos reparar dos por lo cual quedaron 12 Issues sin resolver.

Link al informe de avance de la entrega:

https://github.com/IngSoft-ISA2-2023-2/obligatorio-cantera-diaz-toledo/blob/Develop/Entregas/Entrega%202/Informe%20de%20Avance/InformeDeAvance.pdf

Entrega 3: Software con BDD

La tercera entrega se centró en la incorporación del Desarrollo Dirigido por Comportamiento (BDD). Iniciamos con una reunión de planificación para establecer objetivos y requisitos. Creamos historias de usuario y definimos escenarios BDD utilizando la sintaxis de Gherkin. Desarrollamos tanto el backend como el frontend, implementando pruebas automatizadas para garantizar el cumplimiento de los comportamientos definidos. Ejecutamos pruebas de integración y aceptación, seguidas de refactorización y mantenimiento según sea necesario.

También actualizamos el pipeline para que se ejecuten las nuevas pruebas que desarrollamos con BDD.

Link al informe de avance de la entrega:

https://github.com/IngSoft-ISA2-2023-2/obligatorio-cantera-diaz-toledo/blob/Develop/Entregas/Entrega%203/Informe%20de%20Avance/Informe%20de%20avance.pdf

Entrega 4: BDD y Test Exploratorio

En esta entrega nos encargamos de realizar la automatización de los tests exploratorios relacionados a los bugs que fueron resueltos en la Entrega II, además de los features agregados en la Entrega III. Esto se logró por medio de la herramienta Selenium. También confeccionamos y realizamos un análisis de las métricas del proyecto.

Link al informe de avance de la entrega:

https://github.com/IngSoft-ISA2-2023-2/obligatorio-cantera-diaz-toledo/blob/Develop/Entregas/Entrega%204/Informe%20de%20avance/InformeDeAvance.pdf

Métricas:

Métricas:
https://github.com/IngSoft-ISA2-2023-2/obligatorio-cantera-diaz-toledo/blob/Develop/Entregas/Entrega%204/Metricas/Metricas.md

Conclusiones de las métricas:
https://github.com/IngSoft-ISA2-2023-2/obligatorio-cantera-diaz-toledo/blob/Develop/Entregas/Entrega%204/Metricas/ConclusionesDeMetricas.md

1. Enfoque Metodológico:

Kanban: Utilizamos el método Kanban para gestionar proyectos. Este método visualiza el flujo de trabajo mediante un tablero, permitiendo al equipo ver claramente las tareas en curso, pendientes y completadas. Esto proporciona una visión transparente del progreso del proyecto.

2. Gestión de Tareas:

Definición de Tareas y WIP: La definición de tareas específicas facilitó la asignación y seguimiento de actividades no relacionadas con el sistema. Establecer un límite de trabajo en progreso (WIP) a 3 tareas evitó la acumulación excesiva de trabajo, asegurando un flujo constante y una atención adecuada a cada tarea.

3. Reuniones y Comunicación:

Stand-up Semanal: La reunión semanal proporcionó un espacio regular para actualizaciones, coordinación y sincronización del equipo, aun al no poder cumplir con esta reunión siempre, fue clave para la correcta realización del proyecto. Además la posibilidad de realizar reuniones adicionales en caso de imprevistos garantizó una comunicación efectiva y la resolución rápida de problemas.

4. Flexibilidad y Adaptabilidad:

Gestión Ágil: La capacidad de adaptarse a cambios de requisitos o prioridades de manera ágil fue necesaria, con la cual el equipo contó y aplicó a la perfección. La metodología utilizada permitió ajustes en el proceso sin comprometer la eficiencia del equipo.

5. Proceso de Ingeniería:

Desarrollo Dirigido por Comportamiento (BDD): Se adoptó un enfoque BDD para el desarrollo, que comienza con la planificación inicial, definición clara de historias de usuario y la creación de escenarios BDD con la sintaxis de Gherkin. Este proceso estructurado abarca desde la implementación hasta las pruebas y la refactorización.

6. Mejoras en el Proceso:

Integración de BDD y Pruebas Exploratorias: Se introdujo BDD y pruebas exploratorias para bugs y nuevas funcionalidades. Se ajustó el proceso para abordar reparaciones de bugs en backend y frontend, manteniendo la adaptabilidad a cambios y prioridades del proyecto.

7. Tareas No Relacionadas con el Sistema:

Proceso Eficiente: Se implementa un proceso de un solo paso para tareas no relacionadas con el sistema. Esto garantiza una gestión eficiente e integrada con el flujo principal de desarrollo, simplificando la ejecución de estas tareas independientes.

Reflexiones sobre el aprendizaje

Objetivo: Aplicar un marco de gestión ágil.
Mantuvimos la práctica de reuniones semanales, lo cual resultó efectivo para mantener a todo el equipo informado y alineado con los objetivos. Deberíamos incorporar las conclusiones de la retrospectiva para ajustar nuestros procesos ágiles. Otra cosa que podríamos mejorar sería la asignación de responsabilidades dentro del equipo, asegurándonos de que cada miembro entienda sus tareas.

Objetivo: Analizar la deuda técnica.
Subestimamos la importancia de abordar atentamente la deuda técnica, lo que llevó a complicaciones en la reparación de los bugs. Igualmente, mantuvimos un registro detallado de la deuda técnica acumulada, lo que nos permitió tener una visión clara de las áreas que requerían atención.

Objetivo: Implementar un repositorio y procedimientos de versionado.
Mantuvimos un repositorio centralizado que facilitó el seguimiento de los cambios y la colaboración en equipo. Aunque nos desviamos un poco de la planificación inicial ya que luego no implementamos los releases. Deberíamos generar una nueva documentación con la mantención del repositorio actualizada.

Objetivo: Crear un pipeline con eventos y acciones.
Mantendremos la práctica de dividir el pipeline en pasos claros y manejables para facilitar la identificación y solución de problemas.
Mantuvimos un enfoque incremental en la construcción de la pipeline, lo que facilitó su construcción, mantenimiento y entendimiento.

Objetivo: Integrar prácticas de QA en el pipeline y gestionar el feedback.
Mantuvimos un enfoque colaborativo entre desarrolladores lo que ayudó a que se identifiquen los problemas más rápidamente. Además, implementaremos un sistema de gestión de feedback más ágil para abordar problemas identificados durante las pruebas.

Objetivo: Generar escenarios de testing desde la perspectiva del usuario.
Mantuvimos una variedad de casos de prueba que abordaron diferentes aspectos del sistema. Podríamos mejorar la colaboración con usuarios finales para asegurarnos de que nuestros escenarios de prueba sean representativos de las experiencias del usuario real.

Objetivo: Automatizar el testing funcional o de caja negra.
Experimentamos ciertos desafíos al automatizar la ejecución de las pruebas unitarias, pero una vez que lo logramos fueron de gran ayuda para identificar problemas de manera temprana. Podríamos añadir la cobertura de pruebas automatizadas, para identificar fácilmente en donde el sistema no está siendo testeado.

Objetivo: Reflexionar sobre DevOps.
Trabajaremos en mejorar la documentación y la comunicación de las prácticas DevOps dentro del equipo, asegurándonos de que todos tengan una comprensión clara de los procesos y responsabilidades, y promoviendo una cultura de colaboración continua.

Lecciones aprendidas

Lección Aprendida: Si incorporamos al trabajo las conclusiones de las retrospectivas, entonces mejoramos la adaptabilidad del equipo.

Anécdota: Durante el proyecto, notamos que, a pesar de tener reuniones semanales, no estábamos utilizando completamente las conclusiones de las retrospectivas para ajustar nuestros procesos ágiles. En varias retrospectivas, identificamos como problema que muchas tareas quedaban para realizarse al final de la entrega, pero nunca logramos completar todas las tareas con tiempo de sobra. Incorporar las conclusiones de las retrospectivas mejora la adaptabilidad del equipo. No aplicar completamente estas conclusiones llevó a acumulación de tareas.

Lección Aprendida: Si subestimamos la importancia de abordar la deuda técnica, entonces enfrentamos complicaciones en la reparación de bugs.

Anécdota: A la hora de elegir los bugs a reparar, elegimos uno que luego nos dimos cuenta que no era un bug, sino que funcionaba correctamente pero de una forma diferente a como la habíamos interpretado. Deberíamos haberle dedicado un poco más de tiempo a identificar e investigar los bugs.

Lección Aprendida: Si dividimos el pipeline en pasos, entonces facilitamos la implementación del mismo.

Anécdota: Al mantener una división del pipeline en pasos manejables, pudimos implementarlos más fácilmente. Cuando enfrentamos un problema en uno de los pasos, no afectó a los siguientes pasos, y sirvió de aprendizaje para que no suceda el mismo error en pasos posteriores. La lección aprendida fue la importancia de la modularidad en el diseño de la pipeline.

Lección Aprendida: Si mejoramos la documentación y comunicación del equipo, entonces fortalecemos la comprensión del equipo y promovemos una cultura de colaboración continua.

Anécdota: Experimentamos algunos problemas de coordinación entre el equipo en la segunda parte del proyecto, ya que cambió la extensión de semanas de la entrega. Al mejorar la comunicación e implementar un régimen más frecuente de reuniones, logramos fortalecer la comprensión del equipo y promover una cultura de colaboración continua, lo que contribuyó a la eficiencia y la resolución rápida de problemas. Problemas de coordinación se resolvieron con comunicación frecuente y reuniones mejoradas.

Conclusiones Finales:

Reflexiones sobre el Aprendizaje:

Gestión Ágil:

- Las reuniones semanales fueron efectivas, pero se debe incorporar más las conclusiones de retrospectivas para ajustar procesos ágiles. La asignación de responsabilidades debe mejorar.

Deuda Técnica:

- La subestimación de la deuda técnica causó complicaciones en la reparación de bugs. Mantener un registro de la deuda técnica fue útil para identificar áreas críticas.

Repositorio y Versionado:

- Un repositorio centralizado facilitó la colaboración, pero hubo desviaciones en la implementación inicial. Se destaca la necesidad de mantener documentación actualizada.

Pipeline con Eventos y Acciones:

- Dividir el pipeline en pasos manejables fue clave para la construcción y mantenimiento. La implementación incremental resultó exitosa ya que nunca antes habíamos realizado una tarea del estilo.

Prácticas de QA y Feedback:

- La colaboración efectiva entre desarrolladores fue positiva. Implementar un sistema ágil de gestión de feedback es esencial.

Escenarios de Testing y Automatización:

- La variedad de casos de prueba abordó diversos aspectos del sistema. Hubo desafíos al automatizar pruebas, pero resultaron útiles para identificar problemas temprano.

Reflexión sobre DevOps:

- Mejorar la documentación y comunicación fortaleció la comprensión del equipo y promovió una cultura de colaboración continua.

Guía de instalación para desarrollo y despliegue en producción

Base de datos:

1. Restablecer backup de la base de datos:

En SQLExpress restaurar la base de datos que se encuentra en la siguiente dirección comenzando desde la carpeta raíz del repositorio:

- ./MaterialObligatorio/Implementacion/Base de Datos/backupObligatorioISA2.bak

Backend:

Tener el paso anterior hecho.

1. Clonar el Repositorio:

Clone el repositorio desde la URL proporcionada:

- git clone https://github.com/IngSoft-ISA2-2023-2/obligatorio-cantera-diaz-toledo.git

2. Navegar hasta la carpeta del backend:

Navegue al directorio del backend. Desde la carpeta raíz del repositorio seria:

- cd ./MaterialObligatorio/Implementacion/Codigo/Backend/

3. Instalar Dependencias Backend (.NET):

Ejecutar el siguiente comando para instalar las dependencias:

- dotnet restore

4. Compilar el backend:

Ejecutar el siguiente comando:

- dotnet build

5. Ejecutar test:

Para ejecutar los test debe ejecutar el comando:

- dotnet test

Nos sucedió que al ejecutar los test con este comando hay un test que no pasa, pero al ejecutarlos desde el IDE pasan todos. No lo logramos solucionar.

6. Ejecutar Backend:

Navegar a la siguiente dirección desde donde se encuentra en este momento:

- cd ./PharmaGo.WebApi/bin/Release/net6.0/

Ejecutar el comando:

- dotnet PharmaGo.WebApi.dll

El backend ya está en ejecución.

7. Configuración del Backend:

No debería haber problema, pero en caso de fallar la conexión con la base de datos modifique el archivo appsettings.json con la especificación de su servidor SQLExpress.

Frontend:

Tener los pasos anteriores corriendo.

1. Navegue al directorio del frontend:

Desde la carpeta raíz del repositorio seria:
- cd ./MaterialObligatorio/Implementacion/Codigo/Frontend/

2. Instalar Dependencias Frontend (Angular):

Ejecute:
- npm install

3. Ejecutar Frontend:

Inicie el servidor de desarrollo de Angular:
- npm start

Acceda a la aplicación desde http://localhost:4200/home

Pruebas funcionales:

Tener los pasos anteriores corriendo.

1. Navegue al directorio de las pruebas de selenium:

Desde la carpeta raíz del repositorio seria:
- cd ./Entregas/Entrega 4/Pruebas Selenium

2. Instalar selenium side runner:

Ejecute el comando:
- npm install -g selenium-side-runner

3. Correr las pruebas:

Ejecute el comando:
- selenium-side-runner ObligatorioISA2.side