Entrega Final

Ingeniería de Software Ágil 2

N7A

Franco Ligerini - 196335

Federico Perez - 167726

Rodrigo Trelles - 165832

Índice

Desarrollo del proyecto

En el siguiente informe se detallan y analizan las actividades realizadas en cada una de las cuatro entregas que tuvo el proyecto. Para el desarrollo de este proyecto, debíamos utilizar un marco de trabajo ágil. Se optó por utilizar Kanban ya que este marco de trabajo se enfoca en procesos iterativos e incrementales que proporcionen una visión óptima del flujo de trabajo. Kanban nos permitió también adaptarnos rápidamente a los cambios en el flujo de trabajo que fueron surgiendo en cada entrega.

Comenzamos con un tablero KANBAN estándar, con las típicas columnas To Do, In Progress y Done. Acorde fue avanzado el proyecto, fue necesario ir modificando el tablero según las necesidades, entre ellas, el adaptarlo al desarrollo BDD.

La versión final de nuestro tablero KANBAN terminó conteniendo las columnas de To Do, In Progress, Code Review, Testing, Review y Done.

Por otra parte, realizamos ceremonias de Retrospectiva y de Review al final de cada iteración, con el objetivo de inspeccionar tanto el proceso como el producto respectivamente.

Ambos eventos nos parecieron fundamentales en cada entrega, ya que nos permitieron como equipo reflexionar sobre el trabajo realizado y hacer los ajustes necesarios para mejorar tanto el producto, como el proceso de desarrollo.

Todas estas instancias fueron grabadas y almacenadas dentro del repositorio del proyecto.

Tanto los proyectos de frontend y backend, la documentación, las métricas, los tableros, como las grabaciones de todas las ceremonias se encuentran alojadas dentro de un repositorio único.

Sobre la primera entrega

En esta primera etapa se realizó la definición del marco del proyecto basado en Kanban. Para esto, se creó la primera versión del tablero y se definieron los roles en el equipo. Nos pareció conveniente trabajar sobre un tablero KANBAN tradicional (To Do/In Progress/Done), dada la naturaleza más administrativa de las tareas de esta etapa.
Se seleccionó GitHub como la herramienta para alojar y gestionar nuestro repositorio y tablero. Esto nos permitió tener acceso a toda la información centralizada en una sola herramienta.

Para preparar la siguiente etapa se llevó a cabo un análisis de la deuda técnica del proyecto y se detectaron distintos tipos de issues a ser solucionados tanto en el frontend como el backend, priorizándolos por severidad.

Finalmente, como cierre a esta etapa, se realizó la retrospectiva correspondiente con todos los miembros del equipo siguiendo la metodología DAKI. Obtuvimos como resultado puntos de acción a realizar para la siguiente etapa:

- Comenzar a hacer standups por whatsapp a diario
- Definir mejor las tareas a realizar al comienzo de cada entrega
- Incrementar el esfuerzo por tener un avance diario
- Hacer una reunión para compartir el conocimiento obtenido de la solución de problemas encontrados en la base de datos

En entregas posteriores se lograron todos los puntos de acción menos el primero. Los avances obtenidos fueron de gran importancia para el proyecto, principalmente la definición de tareas y la reunión para compartir el conocimiento de los problemas de la base de datos.

Sobre la segunda entrega

En esta segunda entrega, se seleccionaron dos de los issues reportados en base al grado de severidad de los mismos según la priorización previamente realizada. Estos issues debían ser reparados siguiendo TDD y como prueba de ello, fue necesario demostrar evidencia de la ejecución de los tests.

Decidimos que era conveniente para esta entrega, crear otro tablero KANBAN diferente para separar las tareas de gestión de las de desarrollo.

Durante esta etapa se requirió crear el primer pipeline, tarea que se cumplió satisfactoriamente. El pipeline fue creado con la capacidad de ejecutar pruebas y linters para el backend y linters para frontend. Se explicó con gran detalle cómo se integra dicho pipeline con el tablero y se adaptó el mismo, para contemplar un proceso de desarrollo de software, incluyendo pull requests y una definición de lo que implica cada columna del mismo.

En el aspecto técnico, se desarrolló la solución de ambos bugs seleccionados satisfactoriamente, se organizó y ejecutó una revisión con el Product Owner designado para controlar la calidad y valor del software entregado.

Para terminar esta etapa, se llevó a cabo una segunda Retrospectiva, donde se detectaron los siguientes puntos de acción:

- Crear ticket en el backlog para modificar el workflow de .Net y agregar en el, el reporte de la clase de hoy (17/4/23)
- Meeting para organización de tercera entrega (Repartir tareas y fijar fechas en que vamos a trabajar)
- Comenzar a hacer standups por whatsapp a diario (accion arrastrada de la Retro 1)

En entregas posteriores se cumplieron con todas las acciones y creemos que fue notorio en la entrega 3 el efecto de estas acciones. La tercera entrega fue la que más esfuerzo requería y su ejecución fue mucho más llevadera y sostenible.

Sobre la tercera entrega

Para la tercera entrega se nos solicitó desarrollar dos nuevas features. El desarrollo constó tanto del front-end y del back-end en ambas funcionalidades.

Fue requerimiento la creación de los escenarios de prueba siguiendo BDD en ambas features. Se mostró la evidencia de la ejecución de los mismos, la cual se encuentra disponible en el repositorio.
En cuanto al proceso de ingeniería, referente a los tableros en esta entrega decidimos eliminar uno de los tableros (Documentación) ya que nos pareció que un solo tablero era suficiente. La diferenciación de las tareas decidimos hacerla en base a issues diferentes.
El tablero para esta entrega pasó a tener las siguientes columnas (el motivo y la función de cada columna se detalla en GitHub):

- Backlog
- In progress
- Code Review
- Testing
- Review
- Done

Referente a la configuración del pipeline debimos modificarlo para excluir las nuevas clases agregadas por BDD, ya que era necesario tener el Backend levantado, sino los test fallaban y las modificaciones estaban fuera del scope del trabajo.

Se organizó y ejecutó una revisión con el Product Owner designado para controlar el funcionamiento de los requerimientos solicitados.

Para terminar esta etapa, se llevó a cabo una retrospectiva donde se detectaron los siguientes puntos de acción:

- Hacer un chequeo diario para recordar que se hayan subido las horas en la planilla de registro de horas de las tareas correspondientes
- Analizar lo relacionado con las métricas para el siguiente sprint donde va a ser necesario calcularlas

Sobre la cuarta entrega

La cuarta entrega tuvo como foco la automatización de los test exploratorios de las dos features que fueron solicitados en la entrega anterior. Para esto, se utilizó la herramienta Selenium que nos permite grabar pruebas exploratorias para su posterior ejecución. Dichas pruebas fueron basadas en los escenarios de BDD.
Se creó un informe con la evolución de las métricas de las entregas dos y tres. Como conclusiones del mismo obtuvimos que:

- Se observaron disminuciones significativas en los tiempos de lead time entre ambas entregas
- La implementación de medidas para mejorar la comunicación y el enfoque de equipo tuvieron un impacto positivo.

Se organizó y ejecutó una revisión con el miembro del equipo designado como product owner para validar la ejecución de los test exploratorios utilizando Selenium.

Para terminar esta etapa, se llevó a cabo una retrospectiva donde se detectaron los siguientes puntos de acción:

- Registrar fecha de ingreso en cada columna de cada issue.
- Crear definition of done de cada tipo de issue y agregarlo al repositorio del proyecto.

Lecciones aprendidas

Lección 1

Sintaxis: Si se desea trabajar en un proyecto con C# y .Net sugerimos utilizar el sistema operativo de Windows.

Anécdota: Uno de los miembros del equipo intentó utilizar una Mac para ejecutar el proyecto. Con el Frontend no hubo problema pero con el backend y levantar la base de datos localmente fue un dolor de cabeza, incluso lo conversamos en la primera retrospectiva y decidimos instalar todo de nuevo en una máquina Windows.

Lección 2

Sintaxis: Si se desea entender en dónde están los cuellos de botella del proceso, es necesario registrar el tiempo de vida de cada issue en cada una de las columnas de nuestro tablero de trabajo.

Anécdota: En cada entrega fuimos registrando tres valores para cada issue: La fecha que se agregó a la columna To Do, la fecha que pasó a In Progress y la fecha que pasó a Done. Pero en nuestro tablero, el proceso de In Progress tenía diferentes etapas y son los tiempos registrados de ingreso en cada una de las columnas no pudimos visualizar dentro del proceso In Progress donde estaba habiendo cuellos de botella.

Lección 3

Sintaxis: Si se desea hacer una review completa, sugerimos que se lea uno a uno los requerimientos durante la Review y se demuestre la funcionalidad al terminar de leer cada requerimiento.

Anécdota: En una de las Review, nos salteamos algunos de los requerimientos por no leerlos. Nos enfocamos en el camino feliz y había requerimientos complejos que no fueron mostrados y contemplados en la grabación.

Lección 4

Sintaxis: Si se desea gestionar diferentes tipos de issues, sugerimos identificar diferentes issues en un mismo tablero y no crear distintos tableros para cada tipo de tarea,

Anécdota: En cierto punto del proyecto optamos por tener dos tableros diferentes para diferentes tareas. Uno para las tareas de documentación y otro para los issues tales como bugs o improvements. Esto nos obligaba a llevar actualizado dos tableros diferentes cuando en realidad, crear diferentes labels para cada tipo de issue nos permitió gestionar todo dentro de un único tablero y si algún issue no correspondía en alguna columna, simplemente la salteaba. De esta manera, pasamos a tener todo el trabajo del equipo en un solo lugar.

Lección 5

Sintaxis: Si se desea continuar mejorando y obtener buenos resultados a lo largo de un proyecto, entonces sugerimos evitar tableros estáticos y permitir adaptarlos en base a la necesidad de cada etapa del proyecto.

Anécdota: Todo el equipo tenía la impresión de la experiencia previa de que el tablero Kanban era rígido y no debía ser prácticamente modificado en su estructura. Con el proceso de entregas y que cada una de ellas implica cambios en el proceso, aprendimos y entendimos que el tablero es dinámico y que se puede ajustar siempre que el proceso lo requiera.

Lección 6

Sintaxis: Si se desea definir un proceso de ingeniería de software, entonces sugerimos que todo el equipo sea partícipe.

Anécdota: Antes del proyecto asumimos (y creemos que es un error muy común) que el proceso y el marco de gestión son únicamente responsabilidades del PM. Durante el

desafío observamos que el pipeline está muy vinculado a tareas que son responsabilidad del equipo entero, por lo que todos tienen mucho valor que aportar.

Lección 7

Sintaxis: Si se desea tener claro cuando una tarea esta completada, sugerimos tener definidos desde el inicio el Definition of Done de cada tipo de issue.

Anécdota: En alguna ocasión, nos ocurrió de que algunos realizabamos algunas acciones y otros otras para los issues del mismo ticket, lo que generó un poco de incertidumbre de que es lo mínimo indispensable para cerrarlo. Fue un tema que también surgió en una Retro que hubiese sido buena contar con estas definiciones desde un comienzo para que todos sepamos que debemos hacer y controlar para que un issue pase a Done.

Conclusiones

Durante todo el ciclo de vida del trabajo fuimos aplicando conceptos que eran nuevos para varios integrantes del equipo. Esto implicó dedicar tiempo de investigación como también de ensayo y prueba ya que había cosas que no fueron sencillas de comprender y ninguno de nosotros tenía experiencia previa (por ejemplo: Selenium). El trabajo en equipo fue fundamental para enfrentar los desafíos planteados dada la escasez de tiempo al balancear trabajo y estudio. Esto salió a relucir en múltiples oportunidades en las retrospectivas.

Entendemos que el fin de este proyecto fue experimentar la importancia y valor aportados por un buen marco de ingeniería de procesos. Profundizamos en el concepto de la unión de Desarrollo con el área de Operaciones y nos llevó a reflexionar sobre los procesos que vivimos en nuestros respectivos trabajos.

En cuanto a aprendizaje técnico, destacamos principalmente el valor del pipeline de Github Actions. Creemos que el mismo es uno de los menos costosos a nivel de implementación y mantenimiento, pero al mismo tiempo es uno de los que mayor valor aporta. El pipeline es el corazón del proceso -de hecho, se podría decir que es el proceso en sí- y sin él no se podría automatizar la ejecución de todo el resto de herramientas técnicas con las que se experimentó (como por ejemplo, las pruebas unitarias). Poder correr en momentos clave de nuestro proceso de ingeniería las pruebas o análisis pertinentes aporta muchísimo valor y especialmente calidad al software.

TDD y BDD pueden resultar tediosos al inicio, sobre todo para quienes no tienen experiencia en el desarrollo basado en pruebas. Además, es cierto que implican un costo elevado en esfuerzo para el proyecto, pero los tres estamos de acuerdo en que al superar el primer umbral de incertidumbre encontramos mucho valor. Esto fue mucho más claro al momento de agregar la nueva feature de snacks. La misma estaba suficientemente vinculada al código que ya estaba implementado para que los cambios implicasen un riesgo. Tener las pruebas nos dió mucha seguridad en esta etapa.

La implementación de Selenium fue uno de los terrenos más desconocidos para el equipo. Entendemos que tiene un valor increíble en cuanto a la seguridad mencionada en el párrafo anterior. Por otra parte, creemos que este tipo de prueba requiere una muy buena base de las anteriores; principalmente pruebas unitarias. Esto se debe a que la ejecución y automatización de las pruebas en Selenium es mucho más costosa y compleja en comparación.

Como conclusión final, nos parece importante destacar la actitud de estar abierto a los cambios y adaptarse rápido al entorno y las necesidades que el proyecto requiera. Esta actitud y capacidad es fundamental en esta carrera, y creemos que hicimos un buen trabajo en este aspecto, poniendo por delante de todo siempre la transparencia, la comunicación y la búsqueda continúa de la mejora de los procesos.

Guía de instalación para desarrollo y despliegue en producción

Frontend

Requisitos

- Node.js (https://nodejs.org/es/download)
- Angular y Angular Cli (https://angular.io/guide/setup-local)
- npm (https://angular.io/guide/setup-local)
- Git (https://git-scm.com/downloads)
- Visual Studio Code (https://code.visualstudio.com/download)

Instrucciones

1. Descargar Repositorio del proyecto en tu máquina local
    a. Se aconseja dirigirse a la carpeta en donde deseamos alojar el proyecto.
    b. Abrir una terminal, dirigirse a la ruta de la nueva carpeta y ejecutar el comando *git clone https://github.com/rodrigotrelles/isa2-obligatorio.git*
    c. Por defecto vamos a estar parados en la branch de "main" al clonar el repo.
2. Acceso al proyecto
    a. Ir a Visual Studio Code y abrir la carpeta donde se encuentra el Proyecto de Frontend (/isa-obligatorio/Obligatorio/codigo/ArenaGestorFront).
    b. Al abrir dicha carpeta, veremos desplegados todos los archivos del proyecto Angular de Frontend en sección izquierda del VSC.
    c. Abrir una terminal en VSCode y ejecutar el comando *npm install.* Este comando instalará todas las dependencias del proyecto.
    d. Por último, ejecutamos *ng serve -o* para levantar el proyecto y correrlo local. (*-o* es una flag que abre automáticamente el navegador por defecto de tu máquina)
3. Build de producción

a. Para desplegar este proyecto en producción, es necesario hacer un build del proyecto. Esto se realiza con el comando *ng build –prod* (la flag *–prod* indica que debe considerar las variables de ambiente de producción para el build)

Backend

Requisitos

- Git (https://git-scm.com/downloads)
- Visual Studio 2019 (https://visualstudio.microsoft.com/es/downloads/)
- .NET (https://dotnet.microsoft.com/en-us/download)
- SpecFlow (https://docs.specflow.org/projects/getting-started/en/latest/GettingStarted/Step1.html)

Instrucciones

1. Descargar Repositorio del proyecto en tu máquina local
    a. (Mismo escenario que el caso anterior para el Frontend)
2. Acceso al proyecto
3. Ir a Visual Studio y abrir la carpeta donde se encuentra el Proyecto de Backend (/isa-obligatorio/Obligatorio/codigo/ArenaGestor/ArenaGestor.sln)
4. Al abrir dicho archivo, veremos desplegados toda la lista de proyectos de la solución dentro de Visual Studio.
5. Con el proyecto abierto, hacemos un Build de la solución
6. Click en la pestaña de Compilar
7. Compilar solución

Ejecución de pruebas unitarias:
Para ejecutar las pruebas unitarias, es necesario abrir el IDE de Visual studio 2019 y hacer click en la pestaña Pruebas, seguido de click en Ejecutar todas las pruebas.

Ejecución de pruebas BDD:
El proceso es análogo al de las pruebas unitarias, pero en este caso es necesario el paso previo de ejecutar la aplicación. Para ello, click en el boton IIS Express.

Ejecución pruebas Selenium:

Para ejecutar los casos de prueba en Selenium es necesario instalar el plugin Selenium IDE en el explorador. A continuación se detalla la instalación en Google Chrome:
Ingresar al siguiente link:
https://chrome.google.com/webstore/detail/selenium-ide/mooikfkahbdckldjjndioackbalphokd?hl=es-419
Instalar el plugin Selenium IDE

Luego de instalar el plugin y ejecutar el mismo, abrir el archivo ArenaGestor.side.
Las pruebas deben ejecutarse en el orden establecido con números al inicio del nombre de cada prueba. Selenium ordena las pruebas automáticamente.

Una vez ejecutadas, para volver a ejecutar las mismas es necesario eliminar el usuario creado por la prueba de registro de usuarios. Para esto:
Ingresar a la base de datos en Microsoft SQL Server Management
Click derecho en la tabla dbo.User de la base de datos correspondiente al proyecto
Seleccionar la opción "Select top 1000 Rows"
Borrar el template generado
Ingresar el siguiente comando: DELETE from [ArenaGestorDB].[dbo].[User] where [Email] = 'tester@arenagestor.com'
Luego de eliminar el usuario, la prueba debería ejecutarse correctamente.

Despliegue en producción

Para el despliegue en producción se deben seguir los pasos detallados a continuación:

- Instalar IIS
- Instalar .NET Core hosting (https://dotnet.microsoft.com/en-us/download/dotnet/thank-you/runtime-aspnetcore-5.0.17-windows-hosting-bundle-installer)
- Crear la base de datos ArenaGestorDB (Seguir la documentación incluída en isa2-obligatorio\Obligatorio\Documentacion\Documentacion.pdf)
- Crear un usuario en la base de datos con el nombre “IIS APPPOOL\webApiAgil”
    - En la pestaña Server Roles otorgar los permisos “public” y “sysadmin”
    - En la pestaña User Mapping, habilitar ArenaGestorDB y en la sección de abajo, habilitar todos los roles para esta base de datos
- Copiar el contenido de la carpeta isa2-obligatorio\Obligatorio\Entrega en la carpeta C:\inetpub\wwwroot
- Abrir IIS
- Click derecho en sitios, Agregar sitio web
- Crear el sitio “webApiAgil”
    - Con la ruta de acceso física C:\inetpub\wwwroot\backend
    - Con el puerto 80
    - Click en Aceptar
- Click derecho en sitios, Agregar sitio web
- Crear el sitio “angularSiteAgil”
    - Con la ruta de acceso física C:\inetpub\wwwroot\arena-gestor-front
    - Con el puerto 8080
    - Click en Aceptar
- Asegurarse que para ambos sitios, en Grupos de aplicaciones, al hacer click derecho sobre el sitio y seleccionar configuración avanzada, bajo Modelo de proceso en Identidad figura “ApplicationPoolIdentity”
- Asegurarse que ambos sitios están corriendo y que se ejecutó primero “webApiAgil”
- Se puede acceder al frontend desde http://localhost:8080