Universidad ORT Uruguay

Facultad de Ingeniería

Informe de avance de etapa: Entrega 1

Ingeniería de Software Ágil 2

Entregado como requisito para la obtención del título de Ingeniero en Sistemas

Sofia Montero Crespo - 250618

Romina Pardo Rebella - 253445

Andres Wilchinski - 193694

Tutor: Carina Fontan y Alvaro Ortas

2023

https://github.com/IngSoft-ISA2-2023-2/250618_193694_253445

Índice

Información sobre la entrega

Objetivos del Proyecto

- Aplicar un marco de gestión ágil.
- Analizar la deuda técnica.
- Implementar un repositorio y procedimientos de versionado.
- Crear un pipeline con eventos y acciones.
- Integrar prácticas de QA en el pipeline y gestionar el feedback.
- Generar escenarios de testing desde la perspectiva del usuario.
- Automatizar el testing funcional o de caja negra.
- Reflexionar sobre DevOps

Entregas y Evaluación

El proyecto se divide en cinco entregas. Cada una se evaluará en función de su cumplimiento y calidad. Además, se llevarán a cabo retrospectivas después de cada entrega para reflexionar sobre el proceso y realizar mejoras continuas.

Informe Final

Al final del proyecto, se presentará un informe académico que documentará todas las actividades realizadas, lecciones aprendidas y conclusiones obtenidas a lo largo del proyecto.

Registro de actividades

En cada entrega se aplicarán ceremonias de gestión ágil que el equipo deberá incorporar y adaptar.

Se espera que cada integrante dedique un mínimo de 5 horas semanales a las actividades de ingeniería del proyecto (sin incluir gestión y retrospectivas). Antes de cada entrega, el equipo realizará una retrospectiva y confeccionará un informe de avance, que debe incluir:

- Registro de las actividades realizadas (fecha, horas, integrante).
- Resultados obtenidos en el período.
- Dificultades encontradas y formas de solución.
- Lecciones aprendidas y mejoras en el proceso.

Se debe realizar un informe académico que detalle las actividades realizadas y justifique las decisiones tomadas para cumplir con estos objetivos. Es esencial fundamentar todas las decisiones tomadas durante el proyecto.

Definiciones Iniciales

Definición de Kanban

Para este proyecto se decidió utilizar la metodología Kanban. Kanban en japonés significa letrero o cartel de publicidad y esto es mayoritariamente por su artefacto kanbas board. El mismo está formado por 2 ceremonias obligatorias, el stanup diario y la retrospectiva. En esta metodología no hay roles de product owner o scrum master como si lo hay en scrum pero como también la metodología se tiene que adaptar a las necesidades del equipo y no al revés decidimos agregar dichos roles ya que nos favorecía. El stanup diario como lo sugiere su nombre, se realiza todos los días y dura aproximadamente 15 minutos. En esta asisten los desarrolladores del equipo y cada uno da un breve resumen de en qué trabajaron el día anterior, en que van a trabajar hoy y también dicen si tienen alguna dificultad o no actualmente. La otra ceremonia es la retrospectiva, esta se realiza con el objetivo de que el equipo mejore. Aquí se aplica un sistema DAKI (drop, add, keep, improve), en el cual se define qué cosas:

- Drop: que hizo el equipo o la persona y debe dejar de hacerse
- Add: que no se hizo y estaria bueno que se haga
- Keep: que se hizo y se debe continuar haciendo
- Improve: que se hizo y se debe mejorar

A esta ceremonia además de ir el equipo de desarrollador, se agrega el scrum master, el cual es quien conduce dicha ceremonia. Además de estas 2 ceremonias decidimos agregar la review, en esta le mostraremos al product owner como resolvimos los bugs o cuales son las nuevas funcionalidades agregadas al proyecto. Es por esto que en esta primera entrega no realizaremos review. Kanban también posee una fundamental el cual es el kanban board. En este se puede visualizar fácilmente qué tareas o user stories fueron ya realizadas, se están haciendo o todavía no se han comenzado a hacer. Hay varios tipos de tableros, dependiendo de las necesidades y especificaciones que necesitemos de en qué estado se encuentra cada tarea el tablero que usaremos.

Kanban se basa en 6 principios bien claros.

- Visualizar el flujo de trabajo: esto es posible gracias al kanban board
- Limitar el WIP (work in progress): esto permite evitar los cuellos de botella y defectos rápidamente.
- Gestionar y medir el flujo: medir para ver si se mejora y en base a eso tomar medidas acordes.
- Implementar ciclos de feedback: para este principio las retrospectivas son clave
- Explicar políticas y procedimientos: esto se hace por ejemplo al definir la definition of done o definition of ready
- Mejora continua mediante la colaboración.

Roles asignados

Roles Posibles: PO, SM, DESA, TES

Todos los integrantes del equipo deben poseer los roles de DESA y TES. Por otro lado, en esta primera instancia, se ve pertinente que sean asignados los siguientes roles: Desarrollador, Tester, Product Owner y Scrum Master. Esta decisión fue tomada teniendo en cuenta la letra de la entrega y los artefactos recomendados para la misma. En el caso de Scrum Master, lo consideramos necesario debido a que una de las ceremonias, la retrospectiva, es liderada por el. En cambio, la necesidad del Product Owner se debe a que una de las clasificaciones de los issues debe ser realizada por este rol.

- **Sofia Montero**: Desarrollador, Tester, Product Owner
- **Romina Pardo**: Desarrollador, Tester, Scrum Master
- **Andres Wilshiski**: Desarrollador, Tester, Product Owner

Explicación del tablero y su vínculo con el proceso de ingeniería

Para la creación y primera versión del tablero consideramos las siguientes columnas: ToDo, In Progress y Done. La elección de las columnas se debe al hecho de que, para esta entrega, solo se tiene tareas y no User Stories. Todas las tareas al ser creadas son ingresadas a la columna To Do, cuando cada tarea comienza a realizarse pasa a la columna Doing y una vez terminada pasa a la columna Done. A su vez, las tareas tienen la siguiente estructura:

- **Título:**
  - Tarea X: Nombre_Tarea
- **Descripción:**
  - Responsable: Nombre_del_Responsable
  - Fecha de Inicio: Fecha en la que la tarea fue creada
  - Fecha In Progress: Fecha en la que la tarea se arranca a realizar
  - Fecha Fin: Fecha en la que la tarea fue finalizada

Uso de tableros por entrega

Se creará un tablero por entrega utilizando la herramienta git project. La división de tablero por entrega nos permitirá poseer un control registrado del estado del proyecto en cada entrega.

Estructura y Mantenimiento del repositorio

Creación del repositorio

El repositorio fue creado en la plataforma github. Esto nos permite tener un control de versiones sobre el proyecto, lo cual significa poseer control de las modificaciones realizadas sobre el proyecto.

Para hacer esto, primero se creó un equipo en la plataforma, donde se encontraran presentes cada uno de los mails de los integrantes. Después se agregó a la organización de la materia un nuevo repositorio agregándolo como perteneciente al equipo creado con anterioridad.

Después de eso, se agregó el contenido del proyecto a analizar y modificar al repositorio. Este fue encontrado en la página de aulas de la materia. Después de haberlo descargado, se usó git bash para incorporarlo a la rama “main” del repositorio.

Cuando el proyecto DevOps estuvo incorporado, se decidió agregar una descripción resumida de los objetivos de este obligatorio a su vez del mecanismo de evaluación.

Mantenimiento del repositorio

Para realizar el mantenimiento del repositorio se decidió utilizar las herramientas de Google Hojas de Cálculos y Documentos. A su vez, todos los archivos creados con estas plataformas, serán guardados en una carpeta compartida de Drive. Podrá acceder a esta carpeta aquí.

Uso de Documentos de Google

Esta plataforma nos permitirá crear documentos de informes en formato pdf. Estos informes serán los que poseerán toda la información de análisis de cada entrega. Se creará un documento para cada entrega y luego se resumen todos para la entrega final.

Uso de Hojas de Cálculos de Google

Esta plataforma nos permitirá crear libros de trabajo donde registramos el esfuerzo con tareas y subtareas por entrega. Esto significa que se creará un libro de trabajo para cada entrega.

Registro de Issues

Para registrar los issues vamos a utilizar la plataforma de registro de issues de github. Cada una estará asociada a distintas labels, donde se registra que tipo de issue, el tipo de severidad y de prioridad. A su vez, nos permitirá poder ver los issues ya resueltos y realizar hilos de conversación dentro de el apartado de esa issue.

Puede ver más aquí.

Explicación de forma de testeo

Para realizar el testeo exploratorio decidimos listar los requerimientos con detalle e ir marcando aquellos requerimientos que fueron testeados para no probar todos los mismos. Para cada requerimiento se testea lo que nosotros consideramos casos bordes como, por ejemplo, un nombre de exacto 50 caracteres, una fecha límite, entre otros. Al encontrar un bug o mejora se agrega al reporte de issues de

github junto con una descripción, como este fue encontrado, su responsable y el esfuerzo que llevó encontrarlo. A su vez, cada issue es clasificado por su responsable según su severidad y si es un bug o mejora/issue. Por último, los issues son clasificados por el Product Owner según su prioridad. Si alguno de los desarrolladores tenía dudas sobre si algo que encontró puede ser clasificado como issue se le preguntaba a los otros testers para recibir feedback.

Análisis de deuda técnica

Análisis de código estático

Para realizar el análisis de código estático utilizamos el linter EsLint para que analice el código del front end y así detecte malas prácticas o cosas que se podrían mejorar en el código. Este reportó unos errores menores como por ejemplo el uso de var en lugares donde se podría haber utilizado let o const. O también el uso de let en variables que no son reasignadas por lo cual hubiese sido una mejor práctica definirlas como const. Además esLint encontró variables que se definen en el código y luego nunca se usan, por ejemplo “TitleStrategy” que se importa en el archivo list-invitation.component.ts y nunca se usa en el mismo. Otro error menor que fue encontrado gracias a esta herramienta es el uso de any para definir el tipo de ciertas clases en typescript, aunque entendemos que esto es permitido por el lenguaje, no es una práctica muy recomendada ya que el que estemos usando typescript y no javascript es porque queremos tener la ventaja de conocer los tipos de nuestras variables.

Issues

Tipos de Severidad y Prioridades

Estos tipos fueron identificados en la consigna para ayudar a identificar cuáles problemas son más urgentes o menos y cuales son los errores con mayor gravedad. A partir de la siguiente lista, se crearon los labels para categorizar los issues en github.

Prioridad

Inmediata → Plazo máximo 24 hs. (generalmente los defectos de severidad crítica se asocian a prioridad inmediata).

Alta → Plazo máximo 48 hs.

Media → Plazo máximo un par de semanas.

Baja → Sin plazo.

Severidad

Crítico → Un defecto que obstaculice o bloquee completamente la prueba o el uso de un producto o función.

Mayor → Una función principal que no cumpla con los requisitos y se comporte de manera diferente a lo esperado. Cuando funciona muy lejos de las expectativas o no está haciendo lo que debería estar haciendo.

Menor → Función que no cumpla con sus requisitos y se comporte de manera diferente a lo esperado, pero su impacto es insignificante hasta cierto punto o no tiene un impacto importante en la aplicación.

Leve → Defecto cosmético.

Registro de issues

Para el registro de los issues se utilizó el siguiente esquema:

Tipo Id - Descripción corta	
Descripción	Pequeña descripción del issue registrado
Cómo reproducirlo	Cómo replicar para que ocurra el problema o como este issue fue encontrado
Tipo	Bug (problema del sistema) o Improvement (mejora al sistema)
Severidad	Clasificación dada por el tester
Prioridad	Clasificación dada por el product owner
Responsable	Persona que encontró dicho issue
Esfuerzo (hora/persona)	Cantidad de tiempo/esfuerzo que el responsable tuvo en encontrarlo

Resumen de Issues

Prioridad / Severidad	*Crítico*	*Mayor*	*Menor*	*Leve*
Inmediata	0	0	0	0
Alta	0	1	0	0
Media	0	2	2	0

Baja	0	0	7	2

Análisis del trabajo

Registro de las daily's

13/09/2023: llamada en teams

P1: ¿Alguien está trabajando en algo que no está en el tablero?

- Al no tener trabajo en el tablero sin terminar creamos las tareas que consideramos necesarias para finalizar la entrega 1

P2: Como equipo , ¿qué estamos buscando finalizar?

- Queremos finalizar las tareas necesarias para concretar la entrega 1

P3: ¿Existen cuellos de botella u otros impedimentos al flujo de trabajo?

- Problemas con la instalación de la aplicación, sin la instalacion no podemos comenzar con el testing exploratorio

14/09/2023: mensaje de texto por whatsapp

P1: ¿Alguien está trabajando en algo que no está en el tablero?

- No

P2: Como equipo , ¿qué estamos buscando finalizar?

- Queremos finalizar las tareas necesarias para concretar la entrega 1, y hacer funcionar el proyecto a analizar para todos los integrantes.

P3: ¿Existen cuellos de botella u otros impedimentos al flujo de trabajo?

- Problemas con la instalación de la aplicación, se soluciono cambiando la versión del visual studio.

15/09/2023: mensaje de texto por whatsapp

P1: ¿Alguien está trabajando en algo que no está en el tablero?

- No

P2: Como equipo , ¿qué estamos buscando finalizar?

- El testeo exploratorio

P3: ¿Existen cuellos de botella u otros impedimentos al flujo de trabajo?

- Problemas con la conexión con la base de datos en uno de los integrantes, se soluciono cambiando los puertos para el frontend.

16/09/2023: llamada en teams

P1: ¿Alguien está trabajando en algo que no está en el tablero?

- No

P2: Como equipo , ¿qué estamos buscando finalizar?

- Queremos finalizar las tareas necesarias para concretar la entrega 1, en especial el testeo exploratorio

P3: ¿Existen cuellos de botella u otros impedimentos al flujo de trabajo?

- En este momento no consideramos que haya impedimentos al flujo de trabajo

17/09/2023: llamada en teams, retrospectiva

Ver siguiente apartado.

Registro de retrospectiva

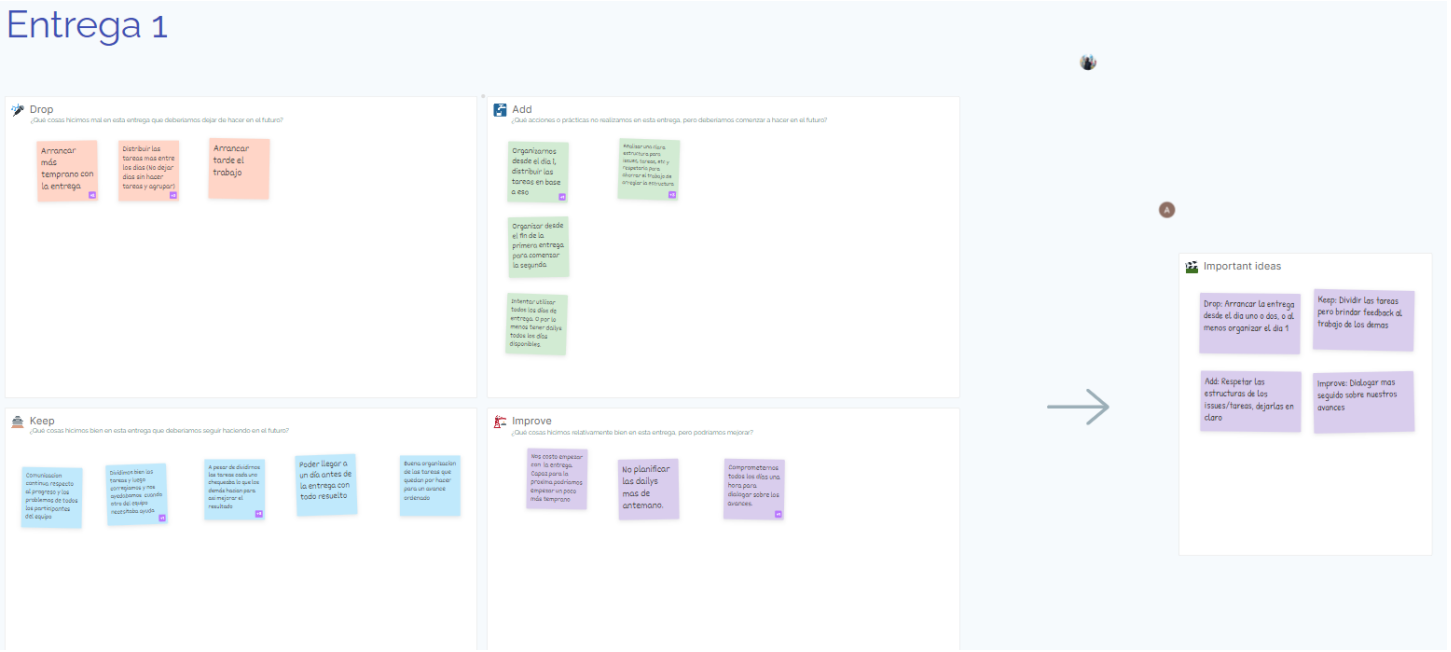
Este registro de retrospectiva se basa en el método DAKI y se utiliza para documentar los aprendizajes y resultados de la retrospectiva.

También se creó un tablero en retrometro para acompañar la retrospectiva.

Formato de la retrospectiva:

- 5 minutos iniciales para responder todas las secciones. Esto se hace de forma anónima.
- Abrir las notas para que todos puedan ver lo que se escribió.
- Dialogar sobre cada sección y discutir para llegar a una respuesta conjunta.
- Luego, para finalizar, se eligiran las ideas más importantes de las planteadas para cumplir en la próxima entrega.

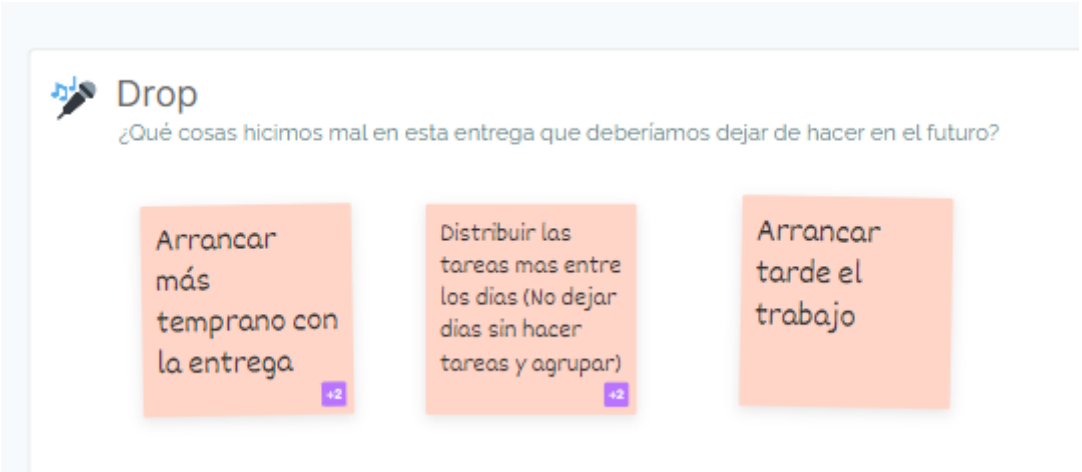
Podrá ver el tablero aquí y el video de la retrospectiva aquí.



Drop

En esta sección, identificamos las cosas que hicimos mal en esta entrega y que deberíamos dejar de hacer en el futuro.

Pregunta: ¿Qué cosas hicimos mal en esta entrega que deberíamos dejar de hacer en el futuro?

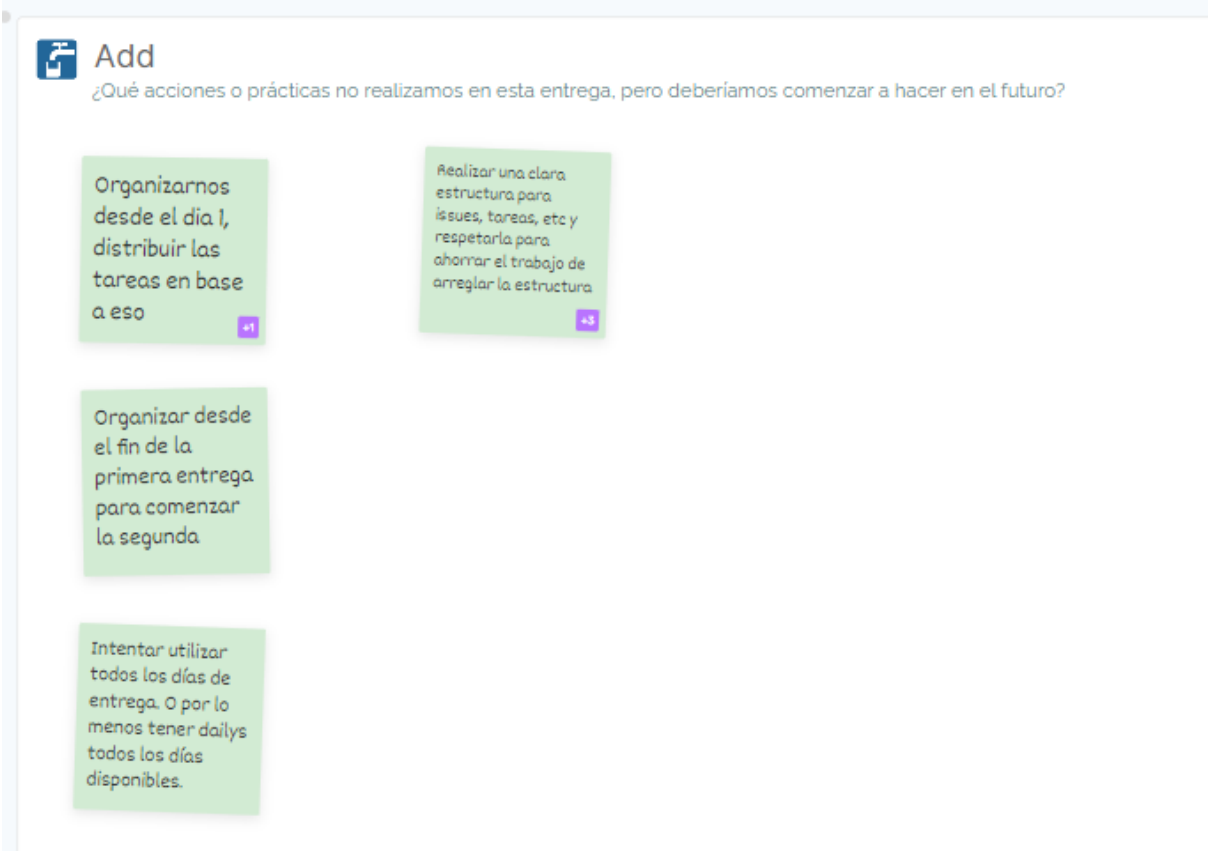


La idea más importante destacada fue: “Arrancar la entrega desde el día uno o dos, o al menos organizar el dia 1”

Add

Aquí, enumeramos las acciones o prácticas que no realizamos en esta entrega, pero que deberíamos comenzar a hacer en el futuro.

Pregunta: ¿Qué acciones o prácticas no realizamos en esta entrega, pero deberíamos comenzar a hacer en el futuro?

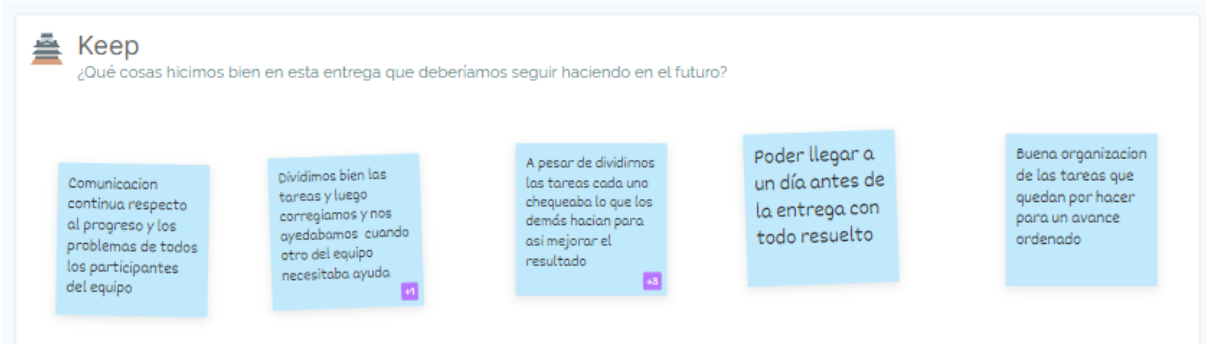


La idea más importante destacada fue: “Respetar las estructuras de los issues/tareas, dejarlas en claro”

Keep

En esta sección, destacamos las cosas que hicimos bien en esta entrega y que deberíamos seguir haciendo en el futuro.

Pregunta: ¿Qué cosas hicimos bien en esta entrega que deberíamos seguir haciendo en el futuro?



La idea más importante destacada fue: “Dividir las tareas pero brindar feedback al trabajo de los demas”

Improve

Finalmente, identificamos las cosas que hicimos relativamente bien en esta entrega, pero que podríamos mejorar.

Pregunta: ¿Qué cosas hicimos relativamente bien en esta entrega, pero podríamos mejorar?

La idea más importante destacada fue: “Dialogar más seguido sobre nuestros avances”

Registro de avance

Registro del trabajo y Esfuerzo

Nro sub tarea	Tareas	Horas/Esfuerzo persona	Responsable	Fecha de Inicio	Fecha In Progress	Fecha de Fin	Cycle Time (dias)	Lead Time (dias)
1	Creacion y organizacion del repo	1:00:00	Romina	11/9	11/9	13/9	3.00	3.00
2	Creacion y mantenimiento del tablero	1:00:00	Sofia	13/9	13/9	17/9	5.00	5.00
3	Explicacion del tablero	0:30:00	Sofia	13/9	13/9	14/9	2.00	2.00
4	Explicacion de la organizacion y versionado	0:30:00	Romina	13/9	13/9	13/9	1.00	1.00
5	Testing exploratorio e ingreso a github	7:00:00	Todos	13/9	14/9	17/9	4.00	5.00
6	Resumen de issues y explicacion	0:30:00	Sofia	13/9	17/9	17/9	1.00	5.00
7	Instalacion de la aplicacion	3:00:00	Todos	13/9	13/9	14/9	2.00	2.00
8	Definicion del proceso de kanban	1:00:00	Andres	13/9	16/9	16/9	1.00	4.00
9	Asignacion de Roles	0:30:00	Sofia	13/9	13/9	13/9	1.00	1.00
10	Testing exploratorio - division de requerimientos	0:30:00	Andres	13/9	13/9	13/9	1.00	1.00
11	Estructura del funcionamiento de la retro	1:00:00	Romina	13/9	16/9	16/9	1.00	4.00
12	Registro de esfuerzo	0:30:00	Sofia	13/9	13/9	17/9	5.00	5.00
13	Explicacion de testeo	0:30:00	Sofia	14/9	14/9	16/9	3.00	3.00
14	Analisis estatico de codigo	1:00:00	Andres	16/9	17/9	17/9	1.00	2.00
		18:30:00						

Para esta entrega contamos con 14 tareas y el total del registro de esfuerzo fue de 18 horas y media.

Podrá ver el seguimiento realizado aquí.

Resultados obtenidos en el período

El objetivo de la primera entrega, y sus resultados, fue hallar los issues del proyecto para poder modificarlos en la próxima entrega. Aquí identificamos mejoras a producir y los bugs hallados. A su vez los clasificamos por prioridad y severidad para poder trabajar ordenadamente y eficientemente en la próxima etapa. También entendimos que hacía la aplicación y cómo debería ser su correcto funcionamiento.

Dificultades encontradas y formas de solución

Las dificultades encontradas en esta entrega fueron el levantamiento de la plataforma. La solución vino de la mano con cambiar la versión de visual studio a la del 2022 y modificar el archivo environment.ts del frontend. El cambio fue modificar el puerto del localhost según el que nos aparezca al correr el proyecto de backend.

Lecciones aprendidas y mejoras en el proceso

Aprendimos a como reportar bugs, a utilizar un linter, analizador de código estático para typescript. Aprendimos cómo organizarnos y cómo dividir las tareas, a su vez de registrar nuestro trabajo exhaustivamente. Esto nos da un sentido más profundo a lo trabajado y/o al esfuerzo hecho.

Al hacer los stand ups diarios y la retrospectiva nos dimos cuenta de cosas que deberíamos mejorar para las siguientes entregas, a su vez de mejoras más pequeñas a lo largo del trabajo para esta entrega.

Con respecto a la entrega anterior

Como esta es la primera entrega no tenemos base para comparar.

Anexo

Anexo 1: Requerimientos para el testeo exploratorio

Administradores

- (*) **Alta de farmacia.** Los datos de una farmacia son: nombre (único en el sistema y máximo de 50 caracteres) y dirección. ✅
    - Notas: si es mayor de 50 caracteres no se permite, si es exactamente 50 no se permite, si es menos de 50 se permite. Sin direccion no se permite y si el nombre ya esta en el sistema no se permite, se permiten nombres con espacios
- (*) **Crear invitación.** El administrador selecciona una farmacia existente, ingresa un nombre de usuario (único en el sistema), un rol (administrador, dueño o empleado) y el sistema genera un código aleatorio numérico de 6 dígitos. Luego el usuario "invitado" utiliza dicho nombre de usuario y el código generado para auto registrarse en el sistema. Ingresa la combinación nombre de usuario / código, el sistema valida y solicita los siguientes datos para el registro: email (único y con formato de email), contraseña (mínimo 8 caracteres y al menos un carácter especial) y dirección. Cuando se da de alta se le asigna el rol correspondiente automáticamente y se almacena la fecha de registro. ✅❌
    - Nota:
        - Ademas, no da el codigo aleatorio. No es un issue pero es medio incomodo ir la lista para ver el codigo. Mejor que aparezca en popup

El administrador puede ver una lista de todas las invitaciones enviadas pudiendo:

- **filtrar por farmacia y/o nombre de usuario y/o rol.** ✅
- **Ver toda su información:** farmacia, nombre de usuario, rol, código aleatorio. ✅
- **Si la invitación fue utilizada o no.** ✅
- **Si aún no fue utilizada poder editar cualquiera de sus datos: farmacia, nombre de usuario, rol y generarle un nuevo código aleatorio.** ❌
    - Nota:
        - Me deja updatear el código a uno de 5 digitos

Dueños

- (*) Ver **solicitudes de reposición de stock** de medicamentos realizadas por los empleados de la farmacia pudiendo aceptar o rechazar dicha solicitud. En caso de que su solicitud sea aceptada, se pasa la solicitud a estado aceptada y se suma la cantidad solicitada al stock del medicamento. ✅
    - Nota:

        - Solo deja aceptar el conjunto, no deja aceptar medicamento por medicamento, igual no importa supongo.
- (*) **Ver en detalle las compras realizadas en cualquier periodo de fechas.** Por defecto le aparece el periodo del mes actual, por ejemplo, a hoy 6 de octubre, le aparece fecha desde 01/oct/2022 y fecha hasta 06/oct/2022, pero puede cambiar cualquiera de dichas fechas para obtener el detalle de medicamento/cantidad/monto vendido en el periodo seleccionado. ❌
    - Notas: no incluye las fechas de los extremos
- **Puede crear invitaciones al igual que un administrador con la restricción que son solo para rol empleado y su farmacia, por lo que solo ingresa el nombre de usuario.** El dueño no puede ver la lista de invitaciones. ✅ ❌

    - Nota:

        - No accese al codigo de la invitación sin el contacto con un admin

Empleados

- **Alta y baja de un medicamento** con los siguientes datos: código (único dentro de la farmacia), nombre, síntomas que trata, presentación (cápsulas, comprimidos, solución soluble, stick pack, líquido, etc.), cantidad por presentación, unidad de medida, precio, stock (el empleado no puede modificarlo solo visualizarlo y solicitar, por lo que un medicamento dado de alta comienza sin stock), y si se necesita receta para su retiro o no. ✅ ❌

    - Nota:
        - Cada vez que se crea un medicamente y falta ingresar un dato o se ingresa mal, alta error y se borran todos los datos. No es un error pero es engorroso

    - **Solicitar reposición de stock de un medicamento**. Los empleados pueden crear una solicitud para reponer un medicamento. La solicitud puede contener varias peticiones, por ejemplo: 1) XF324 - 50, 2) RS546 - 25. Los datos que se deben proveer para cualquier petición dentro de una solicitud son: código del medicamento + cantidad. ✅

Un empleado de farmacia puede:

- **Ver las compras realizadas por usuarios anónimos.** ✅
- **Ver sus solicitudes realizadas**, pudiendo filtrarlas por fecha de solicitud (fecha desde y fecha hasta), código medicamento y/o estado (pendiente, aceptada o rechazada). ✅

Exportación de medicamentos

Exportación de medicamentos. Se ha identificado la necesidad que los empleados puedan exportar medicamentos para poder enviar a diferentes laboratorios, se definió que el sistema

pueda tener la opción de agregar nuevos exportadores desarrollados por terceros de forma dinámica y desde cualquier tipo de fuente pudiendo ser tan diversa como se requiera y en varios formatos.

- Notas: muestra los medicamentos eliminados y el path comienza con la direccion de la carpeta PharmaGo.WebApi dentro de Backend

Usuario anónimo

- **Listar todos los medicamentos que están registrados en el sistema**, pudiendo filtrarlos por nombre y farmacias con stock. ❌
  - Nota:
    - no está por farmacia por stock, pero puede ser que sea por la segunda entrega
    - cuando se filtra por nombre se debe buscar el nombre exacto y completo (issue?)
- (*) **Realizar la compra de varios medicamento**s, para esto, debe indicar cada medicamento que desea comprar junto con su cantidad. En una misma compra pueden haber medicamentos de diferentes farmacias. ✅
- **La compra queda en un estado 'pendiente' hasta que algún empleado de cada farmacia 'confirma'** la misma, restando al stock de la farmacia recién en la confirmación. ✅
- **Los empleados pueden ver todas las compras realizadas por usuarios anónimos a sus farmacias.** Solo ven los medicamentos de la compra correspondientes a su farmacia y si se encuentra en estado pendiente o confirmada. En dicho listado puede confirmar o rechazar cada medicamento de la compra por separado. ✅
- **Al realizar una compra, el sistema debe genera un código aleatorio de seguimiento de la compra que se le muestra al usuario anónimo**. Con dicho código el usuario anónimo puede ingresarlo en una página para ver el estado de su compra, se le muestra si el medicamento está pendiente, confirmado o rechazado. ✅

Anexo 2: Análisis de código estático:

```
$ npx eslint .

C:\Users\andre\Documents\ORT\agil2\oblig\Implementacion\Codigo\Frontend\karma.conf.js
   4:1   error  'module' is not defined                          no-undef
   9:7   error  'require' is not defined                         no-undef
  10:7   error  'require' is not defined                         no-undef
  11:7   error  'require' is not defined                         no-undef
  12:7   error  'require' is not defined                         no-undef
  13:7   error  'require' is not defined                         no-undef
  28:12  error  Require statement not part of import statement  @typescript-eslint/no-var-requires
  28:12  error  'require' is not defined                         no-undef
  28:33  error  '__dirname' is not defined                       no-undef

C:\Users\andre\Documents\ORT\agil2\oblig\Implementacion\Codigo\Frontend\src\app\custom\custom-header\custom-header.component.ts
  27:5   error  Unexpected var, use let or const instead         no-var
  27:9   error  Unexpected aliasing of 'this' to local variable  @typescript-eslint/no-this-alias
  28:60  error  Unexpected any. Specify a different type         @typescript-eslint/no-explicit-any
  43:22  error  Unexpected any. Specify a different type         @typescript-eslint/no-explicit-any

C:\Users\andre\Documents\ORT\agil2\oblig\Implementacion\Codigo\Frontend\src\app\custom\custom-toast\custom-toast.component.ts
  18:5   error  Unexpected var, use let or const instead         no-var
  18:9   error  Unexpected aliasing of 'this' to local variable  @typescript-eslint/no-this-alias
  19:59  error  Unexpected any. Specify a different type         @typescript-eslint/no-explicit-any

C:\Users\andre\Documents\ORT\agil2\oblig\Implementacion\Codigo\Frontend\src\app\guards\authentication.guard.ts
  15:9   error  'state' is defined but never used                @typescript-eslint/no-unused-vars
  16:13  error  'roles' is never reassigned. Use 'const' instead  prefer-const
  17:13  error  'login' is never reassigned. Use 'const' instead  prefer-const
  19:21  error  'role' is never reassigned. Use 'const' instead   prefer-const

C:\Users\andre\Documents\ORT\agil2\oblig\Implementacion\Codigo\Frontend\src\app\interfaces\stock-request.ts
  27:11  error  Unexpected any. Specify a different type  @typescript-eslint/no-explicit-any

C:\Users\andre\Documents\ORT\agil2\oblig\Implementacion\Codigo\Frontend\src\app\pages\admin\create-invitation\create-invitation.component.ts
  19:23  error  Unexpected any. Specify a different type                  @typescript-eslint/no-explicit-any
  36:6   error  Unnecessary semicolon                                     no-extra-semi
  41:6   error  Unnecessary semicolon                                     no-extra-semi
  53:13  error  'name' is never reassigned. Use 'const' instead           prefer-const
  67:13  error  'name' is never reassigned. Use 'const' instead           prefer-const
  84:13  error  'pharmacyName' is never reassigned. Use 'const' instead   prefer-const
  85:13  error  'userName' is never reassigned. Use 'const' instead       prefer-const
  86:13  error  'roleName' is never reassigned. Use 'const' instead       prefer-const
  88:13  error  'invitationRequest' is never reassigned. Use 'const' instead  prefer-const

C:\Users\andre\Documents\ORT\agil2\oblig\Implementacion\Codigo\Frontend\src\app\pages\admin\create-pharmacy\create-pharmacy.component.ts
   4:10  error  'IconSetService' is defined but never used               @typescript-eslint/no-unused-vars
  15:21  error  Unexpected any. Specify a different type                 @typescript-eslint/no-explicit-any
  28:4   error  Unnecessary semicolon                                    no-extra-semi
  42:9   error  'pharmacyName' is never reassigned. Use 'const' instead  prefer-const
  43:9   error  'pharmacyAddress' is never reassigned. Use 'const' instead  prefer-const
  45:9   error  'pharmacyRequest' is never reassigned. Use 'const' instead  prefer-const
```

```
C:\Users\andre\Documents\ORT\agil2\oblig\Implementacion\Codigo\Frontend\src\app\pages\admin\list-invitation\list-invitation.component.ts
  11:10  error  'TitleStrategy' is defined but never used  @typescript-eslint/no-unused-vars
  43:6   error  Unnecessary semicolon                      no-extra-semi
  78:28  error  Unexpected any. Specify a different type   @typescript-eslint/no-explicit-any
  86:24  error  Unexpected any. Specify a different type   @typescript-eslint/no-explicit-any

C:\Users\andre\Documents\ORT\agil2\oblig\Implementacion\Codigo\Frontend\src\app\pages\admin\update-invitation\update-invitation.component.ts
   11:26  error  'Router' is defined but never used                        @typescript-eslint/no-unused-vars
   21:23  error  Unexpected any. Specify a different type                  @typescript-eslint/no-explicit-any
   24:30  error  Unexpected any. Specify a different type                  @typescript-eslint/no-explicit-any
   42:10  error  Unnecessary semicolon                                     no-extra-semi
   48:13  error  'id' is never reassigned. Use 'const' instead             prefer-const
   52:27  error  Unexpected any. Specify a different type                  @typescript-eslint/no-explicit-any
   61:14  error  Unnecessary semicolon                                     no-extra-semi
   67:14  error  Unnecessary semicolon                                     no-extra-semi
  120:63  error  Unnecessary semicolon                                     no-extra-semi
  132:13  error  'pharmacyName' is never reassigned. Use 'const' instead   prefer-const
  133:13  error  'userName' is never reassigned. Use 'const' instead       prefer-const
  134:13  error  'roleName' is never reassigned. Use 'const' instead       prefer-const
  135:13  error  'userCode' is never reassigned. Use 'const' instead       prefer-const
  136:13  error  'id' is never reassigned. Use 'const' instead             prefer-const
  138:13  error  'invitationRequest' is never reassigned. Use 'const' instead  prefer-const

C:\Users\andre\Documents\ORT\agil2\oblig\Implementacion\Codigo\Frontend\src\app\pages\employee\create-drug\create-drug.component.ts
   19:21  error  Unexpected any. Specify a different type                @typescript-eslint/no-explicit-any
   63:9   error  'unit' is never reassigned. Use 'const' instead         prefer-const
   77:9   error  'p' is never reassigned. Use 'const' instead            prefer-const
  115:9   error  'name' is never reassigned. Use 'const' instead         prefer-const
  116:9   error  'code' is never reassigned. Use 'const' instead         prefer-const
  117:9   error  'symptom' is never reassigned. Use 'const' instead      prefer-const
  118:9   error  'quantity' is never reassigned. Use 'const' instead     prefer-const
  119:9   error  'price' is never reassigned. Use 'const' instead        prefer-const
  120:9   error  'unitMeasureId' is never reassigned. Use 'const' instead  prefer-const
  121:9   error  'presentationId' is never reassigned. Use 'const' instead  prefer-const
  122:9   error  'prescription' is never reassigned. Use 'const' instead   prefer-const
  124:9   error  'drugRequest' is never reassigned. Use 'const' instead    prefer-const

C:\Users\andre\Documents\ORT\agil2\oblig\Implementacion\Codigo\Frontend\src\app\pages\employee\create-request\create-request.component.ts
   6:10  error  'RequestDetail' is defined but never used       @typescript-eslint/no-unused-vars
  33:15  error  Unexpected any. Specify a different type        @typescript-eslint/no-explicit-any
  48:13  error  'item' is never reassigned. Use 'const' instead  prefer-const
  56:11  error  'req' is never reassigned. Use 'const' instead   prefer-const
  67:20  error  Unexpected any. Specify a different type        @typescript-eslint/no-explicit-any

C:\Users\andre\Documents\ORT\agil2\oblig\Implementacion\Codigo\Frontend\src\app\pages\employee\delete-drug\delete-drug.component.ts
  15:15  error  Unexpected any. Specify a different type        @typescript-eslint/no-explicit-any
  30:53  error  Unexpected any. Specify a different type        @typescript-eslint/no-explicit-any
  34:14  error  'drug' is never reassigned. Use 'const' instead  prefer-const
  51:20  error  Unexpected any. Specify a different type        @typescript-eslint/no-explicit-any
  53:69  error  Unexpected any. Specify a different type        @typescript-eslint/no-explicit-any

C:\Users\andre\Documents\ORT\agil2\oblig\Implementacion\Codigo\Frontend\src\app\pages\employee\export-drugs\export-drugs.component.ts
  17:21  error  Unexpected any. Specify a different type  @typescript-eslint/no-explicit-any
```