

Universidad ORT Uruguay

Ingeniería De Software Ágil 2

Obligatorio

Entrega 1



Estudiantes:

- Emiliano Ruiz - 263136
- Facundo Jaume - 239695
- Marcel Bejerez - 242484

Docentes:

- Álvaro Ortas
- Carina Fontan

Link al repositorio:

<https://github.com/IngSoft-ISA2-2023-2/obligatorio-bejerez-jaume-ruiz>

Indice

Informe de avance de la etapa.....	3
Actividades realizadas.....	3
Justificación de decisiones.....	5
Resultados obtenidos en el período.....	5
Dificultades encontradas y formas de solución.....	6
Lecciones aprendidas y mejoras en el proceso.....	6
Anexo.....	7
Definición de marco general de KANBAN.....	7
Creación y posterior mantenimiento del repositorio: elementos que contiene y cómo se versionan.....	7
Creación de primera versión del tablero (en GitHub) en función del proceso de ingeniería enmarcado en KANBAN.....	8
Los IDs no se asignan automáticamente.....	8
Primera versión del proceso de ingeniería.....	8
Definición de roles del equipo (PO, SM, DESA, TES, etc.).....	9
Análisis de deuda técnica.....	9
Documento resumen de los issues.....	9
Buenas prácticas que asumimos.....	10
Registro de esfuerzo por tipo de tarea.....	10
Registro de esfuerzo total.....	11
Análisis de la retrospectiva.....	11
Enlace de video.....	12
Drop.....	12
Add.....	12
Keep.....	12
Improve.....	13
Actions.....	13

Informe de avance de la etapa

Actividades realizadas.

Para este obligatorio se busca la aplicación práctica de los conceptos de DevOps. Se nos entregó el producto de software (ya construido) que debemos mantener, ya sea por reparación de bugs o extensión de funcionalidades. Este proyecto que se usará como base utiliza lenguajes y tecnologías del semestre anterior de la carrera.

En esta entrega en particular, se busca descargar el proyecto e interiorizarse con la gestión de tareas a medida que se analiza el código, buscando bugs o mejor dicho, issues en general que representan deuda técnica.

Comenzamos creando un tablero en GitHub. Para esto, primero fue necesario definir cómo íbamos a adaptar el marco de Kanban a nuestro proyecto según nuestras prioridades o más bien necesidades que consideramos para la entrega. La definición del marco general se puede encontrar más detallada en: *Definición de marco general de KANBAN.*

Relacionado con lo anterior, necesitamos crear el repositorio donde versionamos el proyecto. Si bien en esta primera entrega no es tan fundamental y complementamos con otras herramientas como Google Drive, para las próximas aprovecharemos de mejor manera los beneficios que nos brinda GitHub. En el Anexo, desarrollamos en mayor profundidad sobre la *Creación y posterior mantenimiento del repositorio: elementos que contiene y cómo se versionan.*

Sobre el contenido del tablero, desarrollamos con mejor detalle en la sección de *Creación de primera versión del tablero (en GitHub) en función del proceso de ingeniería enmarcado en KANBAN.* Enlistando los atributos que contiene cada tarjeta, junto con una explicación de ciertos puntos que creímos necesarios registrar.

Por otra parte, para llevar a cabo el análisis de deuda técnica, comenzamos todos tomando ciertas actividades como el seteo de proyecto y la familiarización de la solución, ya visualizando posibles issues a detectar. Sin embargo, luego decidimos dividirnos para ser más efectivos en la detección y reporte. Desarrollamos otros aspectos en la sección de *Análisis de deuda técnica.*

Por un lado, Emiliano y Facundo se encargaron de realizar pruebas de Testing Exploratorio en la aplicación. Trabajaron en colaboración, explorando diversas secciones de la solución y la documentación proporcionada. Por ejemplo, Emiliano probó la aplicación en roles de empleado y propietario, evaluando exhaustivamente todas las funcionalidades mencionadas en la documentación y la letra e incluso sometiendo la aplicación a casos borde para verificar la robustez del código.

Por otro lado, Facundo evaluó la aplicación en los roles de administrador y usuario anónimo, siguiendo un enfoque similar. Al igual que Emiliano, leyó tanto la letra del obligatorio como la documentación, y probando todas las funcionalidades y considerando casos borde para identificar áreas que requieran mejoras o adiciones. Su objetivo era abarcar la mayor cantidad de features posibles en busca de posibles mejoras.

Mientras que por el otro, Marcel se enfocó en realizar análisis de código, priorizando los issues que puedan resultar de incumplimiento de buenas prácticas o Clean Code. Se detectaron ciertos “code smells”, aunque es importante aclarar que muchos de los issues detectados no indican que se haya desarrollado mal, sino más bien que se pueden mejorar. Además, sobre este punto, consideramos que lo más importante es siempre lograr una consistencia ya sea siguiendo lo que acuerde el equipo u organización. Por eso, en este caso, se podría decir que los issues vinculados a análisis de código podrían ser subjetivos, pero en base a las reglas que nosotros consideramos que hacen un código limpio (con base en libro de Clean Code) son aspectos a mejorar. Fuera de todo lo mencionado anteriormente, hay que aclarar que no se puede menospreciar el valor de lograr un código limpio, legible que facilite la mantenibilidad a lo largo del tiempo. Se puede encontrar más información sobre lo que tomamos en cuenta en la sección de Buenas prácticas que asumimos.

De todas formas, en ningún momento los integrantes se desentendieron de las otras tareas del proyecto. Es decir, todos estaban pendientes de las otras tareas que se estaban realizando gracias al tablero y a las reuniones que armábamos regularmente. Esto creo que ayudó a lograr un trabajo de gran calidad, complementando y asegurándonos de cubrir muchos puntos y detalles. Esto lo aplicamos para todo: el análisis de código, detección de bugs, registro de horas, gestión del tablero, etc. Esto probablemente disminuya en ciertas áreas para la próxima entrega, dado que buscaremos separar ciertos roles, sin perder esa ayuda mutua que nos benefició.

En paralelo a todas estas actividades, el equipo se encargó de registrar las horas trabajadas, para poder convertirlas en esfuerzo y así, luego, poder extraer métricas como Lead time o Cycle time de cada tarjeta. Esto está evidenciado con mayor detalle en Registro de esfuerzo por tipo de tarea y Registro de esfuerzo total.

El proceso para este registro fue armar una hoja de cálculo con las tareas que se iban realizando y junto a ellas la siguiente información:

- Cant Personas: Indica la cantidad de personas involucradas para la actividad a realizar
- Integrantes: Muestra los integrantes del equipo involucrados
- Fecha: Indica la fecha en la que se realizó la actividad
- Hora Inicio
- Hora Fin
- Duración: Cantidad de tiempo invertido en la tarea
- Horas Persona: Cantidad de tiempo invertido en la tarea por persona
- Total Horas Persona: Es el total de las horas trabajada en el proyecto por persona

Para calcular la Duración, Horas Persona y Total Horas Persona, en vez de calcularlas a mano, optamos por asignar una función para cada columna, y de esta manera obtenerlas automáticamente a partir de los otros campos. Consideramos que llevar este registro y en particular esta solución nombrada anteriormente para no calcular campos a mano va contribuir en gran manera a la hora de calcular las métricas en futuras entregas.

Justificación de decisiones.

En todas las secciones anteriormente mencionadas, siempre justificamos todas las decisiones. En particular, desarrollamos sobre aquellas que tuvimos que adaptar en mayor medida a nuestras necesidades para lograr los mejores resultados. Otro ejemplo de justificación de decisiones, es cuando abordamos el tema de cómo elegimos nuestra estrategia de branching en Primera versión del proceso de ingeniería.

Como se nos ha mencionado desde que empezamos con este tema, adoptamos Kanban como marco de trabajo ágil que no busca ser un contrato, sino una herramienta. Es por esto, que la prioridad siempre va a ser trabajar acorde a lo que el equipo (u organización) considere más beneficioso.

En base a esto, también es necesario hablar sobre la Definición de roles del equipo que tuvimos que llevar a cabo como parte del proyecto para poder trabajar sobre una base organizada con roles y responsabilidades claras. Como indica la letra, todos cumplimos los roles de desarrolladores y testers. Además, definimos un Scrum Master, mientras que un Product Owner no quedó definido con total claridad. En el Anexo justificamos dicha decisión y aclaramos que para futuras entregas vamos a mejorar dicho aspecto y asegurarnos de rotar los roles para que el aprendizaje sea mejor.

Resultados obtenidos en el período.

En cuanto a los resultados obtenidos, los consideramos satisfactorios en base al objetivo planteado para la entrega. Usamos el tablero de gran forma para identificar las tareas y agregarlas al “To do”, así como también para llevarlas hasta “Done”. en esta primer entrega aparecieron más bien tareas, pero siempre estuvieron muy bien organizadas: con atributos claros y útiles, con los responsables asignados, y actualizadas en la columna correspondiente.

Con respecto a esto, creo que también ayudó lo cuidadosos que fuimos al tomar una tarjeta y pasarla a “In progress”, ya que buscamos no pasarnos de un cierto límite en el WIP (Work in Progress). Si bien por momentos pasamos la cantidad de integrantes (que es lo recomendado para que nadie esté trabajando sobre más de una tarjeta a la vez), siempre fue por motivos válidos. En esta entrega, al ser tareas de gestión, muchas veces se llevaban a cabo más de una a la vez. Por ejemplo, el testing exploratorio, análisis de código o la lectura de la letra, fueron tareas que se fueron llevando a cabo en forma paralela a otras.

En cuanto a los issues encontrados, se puede encontrar un resumen en la sección Documento resumen de las issues del Anexo. Los fuimos registrando en una columna llamada “Product Backlog” en la sección de “Entrega 2”. En el Anexo justificamos esta decisión de mejor forma, pero resumiendo, es porque consideramos que iba a ser útil ya tenerlos registrados ahí para la próxima instancia. Descartamos ponerlas en el “To do” porque por lo que nos indicaron, no vamos a resolver todos los issues encontrados. Mientras que también descartamos sólo ingresarlos en la sección de “Issues” de GitHub porque sentimos que quedaba muy alejado de la herramienta diaria y ágil que es el tablero.

Dificultades encontradas y formas de solución.

Una de las dificultades con la que nos encontramos fue que los IDs no se asignan automáticamente en las tarjetas del tablero. Creímos de gran importancia identificar las tareas con un código identificador, que en este caso, bastaba con un número. Sin embargo, las teníamos que asignar a mano, y a medida que el tablero fue creciendo, se fue haciendo cada vez más difícil mantener el orden y evitar duplicados.

Es por esto, que consideramos que estábamos perdiendo lo más importante que es la agilidad, lo que nos llevó a tomar la decisión de completar dicho campo. Además, en algunas ocasiones resultamos con IDs repetidos, que consideramos que tenía un impacto muy negativo como para no tomar medidas.

De todas formas, la solución implementada no es la que nos hubiera gustado. Antes de deshacernos de los IDs, preferimos tenerlos automatizados por los motivos ya mencionados. Pero dependerá en las próximas entregas si encontramos una solución que cumpla con todos los requisitos que buscamos.

Lecciones aprendidas y mejoras en el proceso.

Como conclusión del proyecto, llevamos a cabo la retrospectiva como sugiere el marco de Kanban. Buscamos analizar el proceso de trabajo en el proyecto, con el fin de identificar puntos de fortaleza y puntos de mejora.

Con todos los integrantes del equipo, realizamos la ceremonia, donde el Scrum Master se encargó de grabar y elegir la plantilla, de modo que todos pudieran brindar sus opiniones y aportar para crecer como equipo. Todo el análisis de este proceso se puede encontrar en la sección del Anexo, Análisis de la retrospectiva. Ahí, también se encontrará la evidencia.

Anexo

Definición de marco general de KANBAN

Con respecto a este punto, elegimos utilizar un tablero ágil con las columnas “To do”, “In Progress” y “Done”. Consideramos suficientes esas columnas porque para esta primera entrega se requerían procesos enfocados más en la gestión, seteo de proyecto, análisis de deuda técnica e informes.

La columna de “To do” la utilizamos para visualizar y priorizar las tareas de análisis que necesitan ser completadas. Las tareas en esta columna representan la carga de trabajo pendiente en el proceso de análisis.

“In Progress”, por otro lado, cumplió la función de proporcionar visibilidad sobre el progreso y los responsables de cada tarea. Sirve para monitorear quién se está encargando de cada cosa y poder evaluar si todos los puntos están siendo cubiertos.

Por último, definimos la columna “Done”, que como lo indica el nombre, la utilizamos para las tareas finalizadas. Esta columna refleja todos los logros del equipo y el avance realizado. Además, buscamos registrar en cada tarjeta los atributos necesarios para más adelante utilizar métricas.

Además, decidimos incluir una columna “Product Backlog” que definimos en un tablero del proyecto “Entrega 2”. Tomamos esta decisión basándonos en que los issues reportados nos servirán para la segunda entrega, luego de priorizarlos. Consideramos que iba a ser lo más ordenado para registrarlos.

Creación y posterior mantenimiento del repositorio: elementos que contiene y cómo se versionan

En cuanto a la creación de repositorio, un integrante creó un grupo en la organización de IngSoft-ISA2, y luego los otros dos miembros se unieron al grupo. Dentro del repositorio creamos los proyectos “Entrega 1” y “Entrega 2”, donde agregamos los tableros que utilizamos para la gestión de la entrega.

Para esta primera entrega, utilizamos solo la rama main ya que no hicimos cambios sobre el código y habían pocas cosas que quisiéramos mantener versionadas. Lo relacionado con la documentación y registro de horas lo manejamos con Google Drive, Google Docs y Google Sheets respectivamente. Además, agregamos la evidencia de la retrospectiva que subió una sola persona, por lo que no identificamos ninguna amenaza de conflicto.

De todas formas, ya dejamos creada la rama “develop” para las siguientes instancias dado que en ese caso si consideramos relevante separar en ramas para evitar conflictos de archivos.

Creación de primera versión del tablero (en GitHub) en función del proceso de ingeniería enmarcado en KANBAN.

Luego de realizado lo anterior procedimos a crear el tablero en GitHub basado en KANBAN. Como se explicó anteriormente, el tablero cuenta con tres columnas: To Do, In Progress y Done. Comenzamos creando las tarjetas con las actividades que debíamos realizar y posicionándolas en la columna TO DO.

A las tarjetas le agregamos las siguientes características:

- Assignees
- Status
- Id Number
- Kind
- Severity
- Priority
- Start Date
- Finish Date
- Effort
- To Do Date

Esta información adicional nos proporciona un mayor conocimiento de cada tarjeta lo que nos permitirá conocer más en detalle nuestro trabajo.

En concreto, el tener tres fechas “Start Date”, “Finish Date” y “To Do Date”, los agregamos para posteriormente lograr calcular las métricas de manera más sencilla. Esto se debe a que el Lead Time indica el tiempo de una tarjeta en la columna “In progress”, por lo que se necesitan los datos de “Start Date” y “Finish Date”. Mientras que para el Cycle Time, que indica el tiempo de entrega total, se necesitan los datos de “To Do Date” y “Finish Date”.

Los IDs no se asignan automáticamente.

Por otra parte, nos parece relevante mencionar que en un principio decidimos agregar el atributo “Id Number” a las tarjetas, pero nos terminó complicando y mareando más de lo que ayudaba. El problema es que no se asignan automáticamente por lo que tenemos que estar viendo siempre en todas las tarjetas a ver cuál era el id mas grande, muchas veces resultando en ids repetidos. Es por eso que terminamos dejando muchas tarjetas con ese campo vacío. Para la próxima entrega evaluaremos si nos resulta conveniente aplicarlo o encontramos una forma de automatizarlo.

Primera versión del proceso de ingeniería

Por otra parte, decidimos adoptar GitFlow como estrategia de branching. Si bien en Kanban se estila más usar Trunk-based, nosotros consideramos la otra estrategia como más apropiada para nosotros por varios motivos.

El principal motivo es la comodidad que sentimos con GitFlow dado que lo venimos aplicando hace un tiempo ya. Además, consideramos que podemos obtener una mayor estructura y organización, separando el flujo de trabajo con ramas específicas para características y basando los merge en la finalización de las features.

Definición de roles del equipo (PO, SM, DESA, TES, etc.).

En esta fase del proyecto, comenzamos por establecer los roles de Scrum Master (SM), que fue asumido por Marcel, y los roles de Tester (TES) y Development team (DESA) a cargo de Emiliano y Facundo junto con Marcel, ya que se nos indicó que todos los integrantes debían cumplir estos roles.

El rol de Product Owner (PO) no se asignó a nadie en particular para esta primera entrega. Consideramos la opción de asignar a alguien para esta función, pero dado que los tres integrantes carecían de conocimiento del proyecto y del alcance, decidimos posponer la asignación de rol PO a un integrante particular, ya que no era crítico en este momento.

Por este motivo, para clasificar los issues por prioridad, planificamos reuniones para poder discutir y tomar decisiones. Como ya mencionamos, a medida que encontrábamos bugs o issues de código, íbamos conociendo la solución cada vez más, entonces, consideramos que esta metodología nos iba a traer mejores resultados para definir prioridades en esta entrega. Planeamos designar a alguien para este rol en la próxima entrega de este proyecto y así no dedicar esfuerzo en discutir sobre las prioridades.

Análisis de deuda técnica.

Como ya mencionamos en el informe principal, separamos las tareas de buena forma para poder realizar un análisis más abarcativo. El análisis constó de dos tipos de tareas: análisis de código y detección de bugs, que vienen a ser features solicitadas y no implementadas o mal implementadas.

Para reportarlas usamos como template el ya mencionado en secciones anteriores. Buscamos además agregar una buena descripción que incluya, en los casos necesarios, evidencia del issue o pasos para su reproducción (que en nuestros casos no consideramos tan necesario).

Además, ya los clasificamos según su severidad (a los bugs, porque los issues por análisis de código no son severos) y por prioridad, que explicaremos como lo determinamos en otras secciones.

Documento resumen de los issues

- Issues de tipo bug: 14
- Issues de tipo analisis de codigo: 8
- Bugs de prioridad Inmediata: 1
- Bugs de prioridad alta: 3
- Bugs de prioridad baja: 3
- Bugs de prioridad media: 7
- Issues de severidad alta: 2
- Issues de severidad crítica: 0
- Issues de severidad baja: 15

El esfuerzo requerido para encontrar cada uno de estos issues se encuentra detallado en el product backlog del tablero de la entrega 2.

En esta entrega, cuyo objetivo era identificar issues en el proyecto, ya sea en el código o en su funcionamiento, se identificaron un total de 22 issues. Cada problema fue categorizado como análisis de código o bug, y se le asignaron prioridades que podían ser intermedias, altas, bajas o inmediatas. Además, se les asignó una severidad, que en casos necesarios podía ser alta, crítica o baja.

Es importante mencionar que, si bien todos los issues, ya sean bugs o análisis de código, recibieron una prioridad por parte del Product Owner, los análisis de código solo se les asignó una severidad, generalmente determinada por el desarrollador.

Para la próxima entrega, cuyo propósito es abordar y arreglar algunos de los issues con mayor prioridad, es probable que algunas de las tarjetas de issues sean reevaluadas en cuanto a su prioridad. Esto se llevará a cabo con un Product Owner más definido, quien buscará optimizar el proyecto y su eficiencia.

Buenas prácticas que asumimos.

Con respecto al análisis de código, se debe decir que se llevó a cabo asumiendo por nosotros ciertas prácticas que consideramos que hacen a un código limpio. Basamos estas decisiones en enseñanzas previas de la facultad, cuya base además está en el libro *Clean Code: A Handbook of Agile Software Craftsmanship* de Robert C. Martin.

A continuación, mencionamos algunas prácticas que asumimos para reportar issues:

- SRP: sobre todo aplicado a métodos muy extensos.
- Claridad de expresión, sobre todo en mensajes de error.
- Los nombres de interfaces deberán comenzar con "I".
- Los booleanos siempre deberían tener un prefijo del estilo "is, are, has, have, can,..." siempre deben denotar pregunta, que se responde con true o false.
- Un método recibe la menor cantidad de parámetros posibles, más de tres ya debería evitarse.
- Eliminar código muerto (variables, métodos, etc.)

Registro de esfuerzo por tipo de tarea.

Para registrar el esfuerzo para cada tarea decidimos crear una hoja de cálculo en Google Sheets para anotar la tarea que realizamos, la cantidad de personas y el tiempo invertido en ella. De esta manera, mediante unas simples funciones, logramos calcular el esfuerzo requerido para cada tarea.

A modo de resumen, podríamos armar la siguiente tabla:

Tarea	Esfuerzo(HP)
Descargar el proyecto	1:30
Creación del tablero	0:30
Configuración del tablero	1:00

Setup del ambiente	3:00
Leer la letra del obligatorio	1:30
Reporte de Issues	34:30
Escribir Informe	22:30
Retrospectiva	3:00

Registro de esfuerzo total

En total, como se observa en la tabla, el equipo alcanzó un esfuerzo de 67:30 horas-persona.

Análisis de la retrospectiva.

En esta ceremonia, buscamos reunirnos, discutir, evaluar y tomar acciones sobre los procesos que efectuamos durante estas dos semanas para realizar la entrega solicitada. Para esto, utilizamos la herramienta Metro Retro y elegimos la plantilla DAKI (Drop, Add, Keep, Improve).

Nos reunimos por zoom, entramos todos a la reunión de Metro Retro, marcamos las notas como privadas y nos pusimos a escribir notas hasta que consideramos que nadie tenía nada más para agregar. Luego, decidimos empezar por las notas de Drop, discutimos cada una, argumentando o brindando más información y luego, entre todos, consideramos si era necesario tomar acciones. Lo mismo lo repetimos para el Add, Keep, Improve.

Por lo tanto, luego de la retrospectiva resultó una lista de acciones que vamos a tomar para mejorar el funcionamiento del equipo en futuras entregas de proyecto. En conclusión creemos que llevamos a cabo la ceremonia de muy buena manera; en todo momento los integrantes se sintieron cómodos para realizar comentarios sobre mejoras o aportes al proceso de trabajo.

A continuación, entonces, vamos a intentar: centralizar la comunicación, definir prioridades con tiempo, preguntar más, tener un standup más formal y definir roles claros desde un principio.

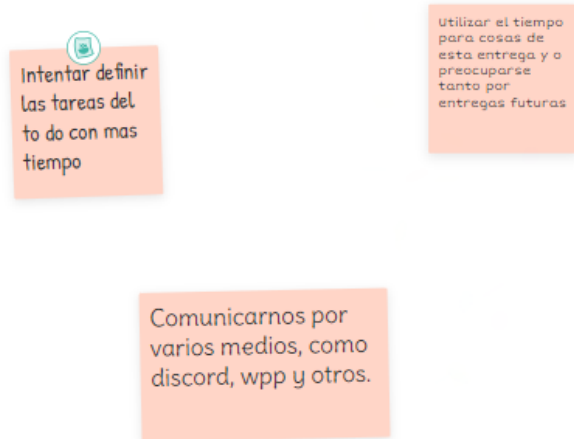
Por otra parte, queremos mencionar que buscamos finalizar la mayoría de las tareas, aunque alguna nos quedó pendiente después de la retrospectiva. Si bien sabemos que el caso ideal es tener todo terminado antes de empezar con esta ceremonia, no fue posible y consideramos que iba a resultar con mayor valor hacerla con tiempo. Tomamos esta decisión porque preferimos tomar un día que nos permitiera evaluar el proceso con tranquilidad. Además, como ya mencionamos, la discusión está enfocada en el proceso de trabajo, por lo que no consideramos trascendentes las tareas pendientes.

Por último, dejamos la evidencia de los distintos aspectos de la ceremonia.

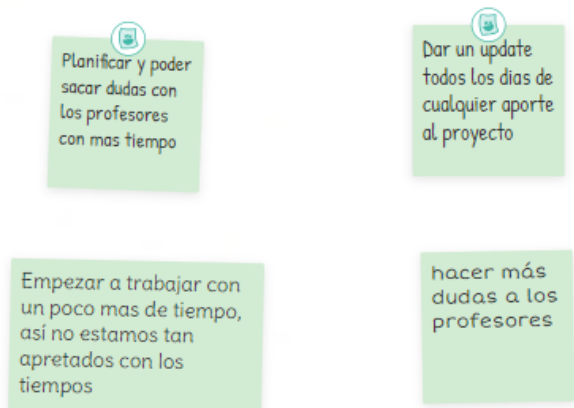
Enlace de video

<https://youtu.be/BGR-xBrWUak>

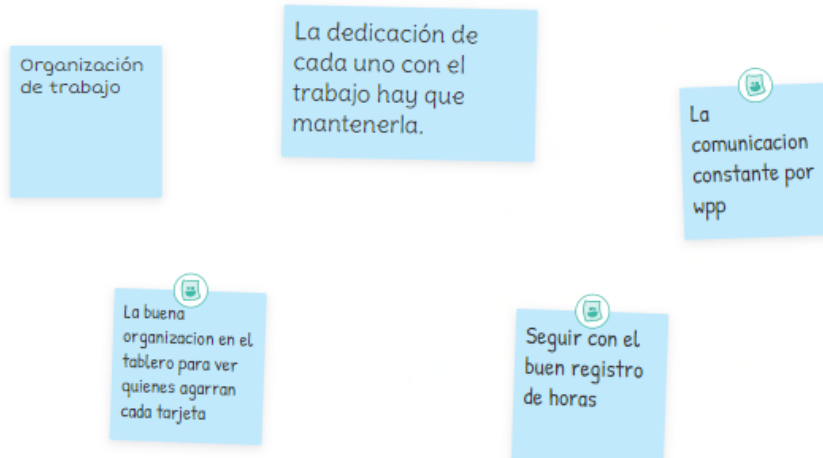
Drop



Add




Keep




Improve

Coincidir más en los horarios para tener discusiones más efectivas


Agregar con más frecuencia el trabajo realizado al excel, para así luego poder contabilizar correctamente las horas persona


 Definir mejor los roles


Deberíamos organizarnos un poco mejor, con nuestros horarios.


 Tener el standup diario mas seguido


Actions

 Centralizar la comunicacion

 Definir prioridades con tiempo

 Definir con tiempo y preguntar mas

 Tener un standup mas formal

 Definir roles claros