ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

SISTEMA DE PRE-PLANIFICACIÓN DE ASIGNATURAS DEL PERÍODO ACADEMICO ORDINARIO PARA LA FIS: UN ENFOQUE ÁGIL PARA LA ENTREGA CONTINUA DE INCREMENTOS FUNCIONALES Y LA GESTIÓN DEL PRODUCTO SOFTWARE

HERRAMIENTAS PARA LA GESTIÓN DEL DESARROLLO ÁGIL DE SOFTWARE CON EL MARCO SCRUM

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SOFTWARE

DAVID ALEXANDER AGUIRRE GALLARDO

david.aguirre@epn.edu.ec

DIRECTOR: JULIO CESAR SANDOBALÍN GUAMÁN

julio.sandobalin@epn.edu.ec

DMQ, septiembre 2023

CERTIFICACIONES

Yo, DAVID ALEXANDER AGUIRRE GALLARDO declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

_____

DAVID AGUIRRE

Certifico que el presente trabajo de integración curricular fue desarrollado por DAVID AGUIRRE, bajo mi supervisión.

_____

JULIO CESAR SANDOBALÍN GUAMÁN

DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

DAVID AGUIRRE

JULIO SANDOBALÍN

DEDICATORIA

Este logro va dedicado para todos mis familiares y amigos, los cuales han sido fuente de motivación y apoyo en los momentos difíciles. En especial, se lo dedico a mis padres y hermanos, que han sido un soporte fundamental para alcanzar esta meta.

AGRADECIMIENTO

En primer lugar, agradezco a Dios por concederme la vida, la salud y las capacidades intelectuales que me han permitido culminar con éxito la carrera. A mis padres, por sus sacrificios, oraciones, apoyo y guía. A mi director de tesis Julio Sandobalín y a mis compañeros de equipo, por su ayuda y compromiso con el desarrollo del proyecto.

ÍNDICE DE CONTENIDO

RESUMEN

En este documento se describe la gestión de herramientas que apoyan al marco Scrum para el desarrollo de un sistema de preplanificación de asignaturas para la Facultad de Ingeniería de Sistemas FIS. Primero, se realizó un estudio de las herramientas en base a criterios como la experiencia en el uso, cantidad de usuarios y licencias. Luego, se implementó un conjunto de herramientas que soportan el marco Scrum, tales como Microsoft Teams para apoyar el Daily Scrum; Azure DevOps para la gestión del Product Backlog y el Sprint Backlog; la aplicación web Planning Poker para la estimación de historias de usuario; y la aplicación Miro para las Sprint Retrospectives. En el desarrollo de cada Sprint el equipo afrontó diversos problemas como falencias de comunicación, falta de compromiso, falta de experiencia, entre otros. Sin embargo, en cada Daily Scrum, Sprint Review y Sprint Retrospective el equipo ganó confianza, conocimiento y experiencia; al mismo tiempo que el sistema mejoró la entrega de valor mediante la entrega de incrementos funcionales de software al usuario. Finalmente, luego de siete Sprints el equipo liberó la primera versión del sistema de preplanificación de asignaturas para la FIS.

PALABRAS CLAVE: Scrum, Agilismo, Azure DevOps, Desarrollo de Software.

ABSTRACT

The document describes a management of tools to support the Scrum framework in the develop of a subjects pre-planning system for the Facultad de Ingeniería de Sistemas FIS. First, a study of tools based on experience, users quantity, and licenses was carried out. Then, a toolchain to support the Scrum framework was implemented. In this scenario, the Microsoft Teams tool helped the Daily Scrum, Azure DevOps managed the Product and Spring Backlog, the web Planning Poker application let the team estimate user stories, and Miro provided support in the retrospective meetings. During each Sprint, the team faced communication problems, lack of experience and compromise, etc. However, the Scrum events, such as Daily Scrum, Sprint Retrospectives, and Sprint Reviews, helped the team to achieve experience, confidence, and knowledge. At the same time, the system improves the delivery value by delivering functional software increments to the user. Finally, the first version of the subjects pre-planning system for the FIS was released after seven Sprints.

KEYWORDS: Scrum, Agile, Azure DevOps, Software Development.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

El componente inició con el análisis de las propiedades y limitaciones de herramientas utilizadas por equipos Scrum para gestionar los eventos y generar los artefactos propios del marco. De manera general, Scrum posee cinco eventos principales: Sprint, Sprint Planning, Daily Scrum, Sprint Review y Sprint Retrospective. Estos eventos de Scrum permiten la aplicación de los pilares del marco, que son la inspección, adaptación y transparencia. Por otro lado, los artefactos de Scrum son el Product Backlog, Sprint backlog y el Incremento.

Para facilitar la implementación de eventos y la construcción de los artefactos de Scrum, se realizó un estudio de varias herramientas. El objetivo del estudio fue recopilar información de cada herramienta y analizar las propiedades que se adaptaban a la experiencia y conocimiento del equipo. Posterior al análisis, se llevó a cabo la selección de herramientas por medio de los criterios importantes para el equipo, tales como: licencia, experiencia, integración con herramientas y cantidad de usuarios.

El componente incluyó la implementación de herramientas para facilitar la realización de los eventos y la construcción de artefactos del marco Scrum. Por ejemplo, para el Daily Scrum se configuró en Microsoft Teams una reunión recurrente en un horario acordado por el equipo. Igualmente, se utilizó la aplicación web de Planning Poker[1] para llevar a cabo la estimación de esfuerzo de historias de usuario y la Sprint Retrospective. Adicionalmente, Microsoft Teams enviaba notificaciones de recordatorio de la reunión automáticamente a los respectivos correos electrónicos de cada integrante.

Se realizó la configuración de la plataforma Azure DevOps para la elaboración y actualización del Product Backlog. Esta herramienta facilitó las tareas de planificación, estimación y priorización de las historias de usuario. Así mismo, la integración con el sistema de control de versiones facilitó la trazabilidad entre el desarrollo y las tareas técnicas.

La implementación de la herramienta web Planning Poker facilitó al equipo a realizar la estimación de las historias de usuario de manera online. Esta herramienta se utilizó para realizar el Planning Poker, que es una actividad del Sprint Planning. Un factor fundamental es que la aplicación web Planning Poker mantiene en secreto la votación de cada integrante, lo cual hace que la estimación sea individual.

[1] https://planningpokeronline.com/

1.1 Objetivo general

Implementar herramientas que apoyen al marco Scrum para dar soporte a la construcción de un sistema web de preplanificación de asignaturas para la Facultad de Ingeniería de Sistemas FIS.

1.2 Objetivos específicos

1. Analizar herramientas que apoyan a Scrum para gestionar sus eventos y artefactos, y verificar que se adapten a las capacidades del equipo Scrum.
2. Implementar herramientas para facilitar las prácticas, eventos y obtención de artefactos de Scrum.
3. Verificar la implementación de herramientas de apoyo a Scrum durante el desarrollo de un sistema de preplanificación de asignaturas para la FIS.

1.3 Alcance

En el presente documento se describe la implementación de herramientas que apoyan el marco Scrum para gestionar el desarrollo de un sistema de preplanificación de asignaturas. Se utilizó Microsoft Teams, Azure DevOps, Planning Poker y Miro para apoyar la gestión de Scrum. De esta manera, Microsoft Teams se utilizó para el Daily Scrum por medio de la configuración de reuniones virtuales. Así mismo, Azure DevOps ayudó en la elaboración del Product Backlog, Sprint backlog, Integración y Despliegue Continuos. La aplicación web Planning Poker se utilizó para las estimaciones de historias de usuario y la aplicación Miro[2] para realizar las Sprint Retrospectives.

La implementación de estas herramientas proporcionó experiencia en su uso y soporte al desarrollo del sistema de preplanificación de asignaturas para la FIS. Se destaca la importancia de las planificaciones iterativas, vídeos de reuniones, Product Backlog y listas de chequeo como evidencia del proceso.

[2] https://miro.com/

1.4 Marco teórico

Ágilismo

En el contexto de desarrollo de software, el término ágil se refiere a la adopción de un enfoque alineado con el Manifiesto ágil (Kent Beck et al., 2001) establecido para la gestión, desarrollo y construcción de productos software. Este manifiesto contiene las prioridades y principios que se utilizan para construir software con un enfoque ágil.

El manifiesto ágil da mayor valor a los individuos e interacciones sobre herramientas y procesos; al software funcional sobre documentación excesiva, a la colaboración del cliente sobre negociación del contrato; y la respuesta al cambio sobre el seguimiento de un plan.

El manifiesto ágil propone doce principios para los equipos que decidan adoptar la filosofía del agilismo. Estos principios son la guía para que los equipos puedan entender y poner en práctica el enfoque ágil en el desarrollo de software.

El principio de la **satisfacción del cliente** se basa en realizar entregas continuas del producto software con alto valor, de tal manera que el cliente valide si el producto construido tiene relación con el producto esperado. Este fundamento también permite al equipo de desarrollo recibir retroalimentación temprana para adaptarse a las necesidades del cliente.

La **entrega continua** de funcionalidad hace énfasis en que se debe presentar avances funcionales en cortos periodos de tiempo. Según el manifiesto ágil (Kent Beck et al., 2001) el periodo de tiempo oscila entre 2 a 4 semanas, dando preferencia al menor plazo posible.

El **progreso basado en software funcional** implica que las entregas continuas son la medida del avance del equipo de desarrollo. Sin embargo, este principio puede conducir a que la entrega realizada sea de baja calidad, debido a que se toma en cuenta la entrega de funcionalidad y no la calidad de esta (Williams, 2012).

Para el principio de **mejora continua** se considera fundamental la búsqueda de la excelencia técnica y el buen diseño. La importancia de este principio radica en que se construya software de manera flexible y clara, en la cual el diseño es una actividad continua (Fowler & Highsmith, 2001).

La **adaptación al cambio** invita a los equipos ágiles a que exista apertura a realizar modificaciones sobre lo que se ha trabajado. Sin embargo, este principio también se puede considerar como una desventaja debido a la cantidad de cambios que pueden ser introducidos en el desarrollo del producto software (Sunner, 2017).

Paralelamente, el principio de **desarrollo sostenible** impulsa al equipo a mantener un ritmo constante de esfuerzo en un periodo de tiempo determinado (Fowler & Highsmith, 2001). Sin embargo, el trabajo realizado puede ser afectado por diversos factores tales como: las horas de disponibilidad, la motivación del equipo, la experiencia, los conocimientos, los días festivos, entre otros (De Melo et al., 2013).

En relación con la **simplicidad** que se establece en el manifiesto ágil, compromete a los equipos ágiles a implementar cada necesidad del cliente de la forma más simple posible. Así, la simplicidad puede tener diferentes enfoques tales como simplicidad del producto, para el cliente y para el proceso (Santos, 2016).

Por otra parte, los principios de **colaboración y comunicación** en equipos ágiles se refieren a la capacidad de intercambiar información de manera efectiva y apoyarse mutuamente para alcanzar el objetivo del proyecto. De este modo, la comunicación es un factor fundamental que abarca la retroalimentación, la voluntad de expresar problemas o ideas y la participación continua (Kropp et al., 2014).

La **motivación** del equipo promueve la correcta provisión de recursos para que el equipo adquiera confianza y realice su trabajo (Fowler & Highsmith, 2001). Además, la motivación permite obtener resultados importantes, tales como: rendimiento de alta calidad, optimización del tiempo de entrega del proyecto y adaptación al presupuesto (Sharp et al., 2009).

Finalmente, los principios de **autoorganización** y **autoevaluación** instruyen a los equipos ágiles a ser autónomos. Para alcanzar la autonomía de un equipo es importante que se incluyan tres características que son: la variedad de disciplinas, la libertad de manejar el esfuerzo realizado cada día y la autorregulación junto con la mejora continua (Takeuchi & Nonaka, 1986).

Scrum

Ken Schwaber & Jeff Sutherland (2020) lo definen de la siguiente manera:

“Es un marco de trabajo liviano que ayuda a las personas, equipos y organizaciones a generar valor a través de soluciones adaptables para problemas complejos” (p. 3)

Por otro lado, según Rubin (2012):

“Es un enfoque ágil para desarrollar productos y servicios innovadores” (p. 1)

Siguiendo la misma línea, este marco de trabajo permite gestionar la construcción de productos software por medio de iteraciones que ocurren en un periodo de tiempo. A estas

iteraciones temporales se las conoce como Sprints y tienen una duración de dos hasta cuatro semanas como máximo.

Scrum está basado en el empirismo y en el pensamiento Lean. El equipo aprende en base a la experiencia, lo cual hace que sea importante adaptarse continuamente a mejoras en el proceso o en la forma de realizar las actividades. El pensamiento Lean apoya a Scrum en cuanto a mantener bajo desperdicio, empoderar al equipo y en el aprendizaje continuo.

En Scrum, se deben aplicar los valores de **apertura al cambio**, **coraje**, **respeto**, **enfoque** y **compromiso**. La **apertura al cambio** hace que el marco sea adaptable y el **coraje** se necesita para aceptar los cambios e implementarlos. De igual manera, se fomenta el **respeto** mutuo y la confianza en las capacidades del equipo, mientras que el **enfoque** y el **compromiso** son esenciales para lograr los objetivos en cada iteración. El trabajo conjunto y el apoyo son fundamentales en este enfoque.

Por otro lado, los pilares de Scrum son la inspección, la adaptación y la transparencia. El marco implementa estos tres pilares por medio de cinco eventos que el equipo mantiene durante el proceso. De manera general, la inspección permite analizar mejoras en cuanto al proceso, la adaptación permite que las mejoras se reflejen en el equipo y la transparencia ayuda al equipo a compartir el conocimiento.

El equipo de Scrum está compuesto por tres roles que son: el Scrum Master, Product Owner y Scrum Developers o Equipo de Desarrollo. Cada rol tiene sus propias funciones y aportan en el desarrollo del producto software. Es importante considerar que Scrum recomienda un equipo desde 6 hasta 10 personas.

El **Scrum Master** cuida que el equipo adopte las prácticas, valores y principios de Scrum. El rol de Scrum Master facilita el Daily Scrum retirando los bloqueos que el equipo pueda tener (Noll et al., 2017). Adicionalmente, el Scrum Master influye en el equipo para alcanzar la apertura al cambio, enfoque en la excelencia y mantener la motivación (Ramos & Vilela Junior, 2017).

Otro de los roles de Scrum es el **Product Owner**, este integrante del equipo es el medio para comunicarse con el cliente. Por medio de negociación y conversación con el cliente, el Product Owner transmite las ideas y necesidades al equipo de desarrollo. El Product Owner maximiza los resultados del equipo de desarrollo y de las salidas que se obtienen de cada actividad por medio del retorno de la inversión (Sverrisdottir et al., 2014). De esta manera, el Product Owner debe conocer ampliamente del negocio y relacionarse con los

clientes y usuarios, con el objetivo de asegurar que el producto genera valor (McGreal & Jocham, 2018).

El **equipo de desarrollo** se encarga de la construcción del producto que satisface las necesidades del cliente. Para ello, los integrantes del equipo dividen las historias de usuario en tareas técnicas que serán implementadas en un Sprint. Además, estiman el esfuerzo de cada historia de usuario en base a la complejidad y al finalizar el Sprint el equipo debe entregar un incremento del producto software.

Los eventos de Scrum que soportan la inspección, la adaptación y transparencia son: el Sprint, la Sprint Planning, la Daily Scrum, la Sprint Review y la Sprint Retrospective. En cada evento intervienen los miembros del equipo en diferentes maneras, así como cada evento tiene su razón de ser.

La **Sprint Planning** se realiza al inicio de cada Sprint y se basa en la selección de historias de usuario que se implementarán durante el tiempo que dura el Sprint. También, se estima el esfuerzo de cada historia de usuario y se prioriza las que generan mayor valor para el cliente. Anterior a la planificación del Sprint, se debe tener una lista de historias de usuario refinadas y priorizadas, la cual se conoce como Product Backlog.

La **Daily Scrum** permite al equipo conocer los avances que se han obtenido por medio de las preguntas: ¿qué hice ayer?, ¿qué voy a hacer hoy? y ¿si existe algún obstáculo? Cada integrante del equipo debe responder estas tres preguntas. En este punto, el Scrum Master puede intervenir para resolver los posibles obstáculos o para resolver dudas sobre el marco. Una consideración importante es que estas reuniones deben tener una duración máxima de 15 minutos. Sin embargo, estas reuniones pueden tornarse difíciles de agendar cuando se agrupan elementos en el equipo que se ubican en diferentes zonas horarias (Berczuk, 2007).

La **Sprint Review** es un encuentro donde se realiza una presentación sobre la versión potencialmente entregable del producto software. En este evento se evidencia el esfuerzo realizado por los miembros del equipo en base a las historias de usuario planificadas. En este punto se inspecciona si el producto que se está construyendo es igual al producto esperado y se recibe retroalimentación de parte del cliente.

Por su parte, la **Sprint Retrospective** es una forma de evaluar mejoras para el proceso. Por ejemplo, en este punto los integrantes realizan actividades interactivas que permitan inspeccionar las fortalezas del equipo, las debilidades, los errores en la interacción y las falencias de comunicación.

Los equipos ágiles que adoptan el marco Scrum para la gestión de un producto software generan entregables a partir de las actividades realizadas. Ken Schwaber & Jeff Sutherland (2020) mencionan tres artefactos: Product Backlog, Sprint backlog y el Incremento.

El **Product Backlog** hace referencia al conjunto de historias de usuario que representan las necesidades y son usualmente generadas por el Product Owner. El Product Owner debe ayudar a los Scrum Developers a entender las historias de usuario para que puedan seleccionarlas e incluirlas en la implementación de un Sprint. Luego, los Scrum Developers se encargan dividir las historias de usuario en tareas técnicas que serán implementadas.

El **Sprint Backlog** surge a partir del Sprint Planning, ya que contiene una parte de las historias de usuario del Product Backlog. El Sprint Backlog está asociado a un objetivo específico sobre el incremento del producto que se requiere construir. Una vez que se seleccionaron las historias de usuario del Sprint, el equipo debe estimar estas historias para determinar el esfuerzo que se realizará en el Sprint. Adicionalmente, el Sprint Backlog representa el trabajo que el equipo de desarrollo realizará durante un periodo de tiempo.

Al finalizar un Sprint, el equipo obtiene un avance del sistema que puede ser utilizado por los usuarios. Este avance es lo que se conoce como **incremento** potencialmente entregable.

Historias de Usuario

En el contexto de desarrollo de software, son elementos que siguen una estructura determinada y ayudan al entendimiento, negociación e implementación de las necesidades de un cliente específico. Sin embargo, una definición más actual establece que las historias de usuario son requerimientos plasmados en lenguaje natural, que responden a las preguntas quién, qué y para qué (Wautelet et al., 2014).

Según Jeffries (2001): “las historias de usuarios se componen por tres partes que son card, conversation y confirmation; también denominadas como las tres C”. Primero, una tarjeta (card) permite plasmar por escrito la necesidad del cliente. La tarjeta es una promesa de una conversación (conversation) con el cliente para entender su necesidad y solicitar aclaraciones. Finalmente, la confirmación (confirmation) permite definir criterios de aceptación para verificar la correcta implementación de la historia de usuario en el producto software.

Las **historias de usuario** son ampliamente utilizadas en equipos que adoptan un enfoque ágil, aunque son difíciles de abordar. Por un lado, la forma en que se escriben permite que sea fácil de entender para el usuario, mientras que por otro lado se puede presentar

ambigüedad o inconsistencia (Raharjana et al., 2021). Para asegurar la calidad de las historias de usuario se pueden utilizar ciertos atributos como independencia, negociable, estimable, capacidad de generar valor, indivisible, evaluable, entendible, entre otros (Heck & Zaidman, 2018). Una consideración importante es que, a los seis primeros criterios mencionados se los conoce como atributos INVEST, por sus siglas en inglés.

El atributo de **independencia** se refiere a la capacidad de la historia de usuario de mantener la coherencia y la legibilidad, sin necesidad de acudir a otra historia (Cohn, 2004). Este atributo es importante para que los Scrum Developers puedan realizar una correcta estimación y posterior a ello la implementación.

De la misma manera, una historia de usuario debe ser **negociable** porque permite al Scrum Team junto con el cliente actualizar y refinar la descripción con el paso del tiempo (Koelsch, 2016). La negociabilidad se genera cuando se describe una funcionalidad o proceso que es núcleo del negocio, lo cual abre paso a la conversación entre el cliente y el equipo de desarrollo sobre los detalles de la historia de usuario (Welin Matti, 2013).

Las historias de usuario **estimables** permiten a los usuarios dar valoraciones sobre la complejidad que puede tener y el tiempo que se consumirá en su implementación (Koelsch, 2016). Este atributo es importante para que los equipos organicen su esfuerzo y planifiquen el trabajo que será implementado en un determinado período de tiempo.

La capacidad de **generar valor** debe ser direccionada a la persona que usará el sistema o a la persona que adquiere el producto software. Con tal consideración, se debe evitar las historias de usuario que solamente generan valor para los desarrolladores (Cohn, 2004).

Por otro lado, las historias de usuario **indivisibles** son aquellas que abordan los requerimientos del cliente con una alta precisión. Por tal motivo, al intentar dividir una historia de usuario indivisible, esta perderá sentido o carecerá de coherencia, inclusive se tornará dependiente de otras.

De la misma manera, las historias de usuario que son **evaluables** son aquellas que admiten formas de comprobación para verificar que se han implementado correctamente. Esta característica es importante porque permite a los Scrum Developers determinar cuándo una historia ha sido completada y les permite comprobar el funcionamiento mediante pruebas (Cohn, 2004).

Azure DevOps

Es una plataforma de Microsoft que apoya al proceso de desarrollo de software a través de tableros para la gestión de requerimientos, sistema de control de versiones, pipelines de

integración y entrega continua, planes de prueba y gestor de artefactos. Con esta plataforma se puede abarcar el proceso de desarrollo de software desde los requerimientos hasta el despliegue, incluyendo la ejecución de pruebas.

Por un lado, los tableros de Azure DevOps permiten la gestión de requerimientos en base al enfoque de desarrollo utilizado. Por ejemplo, para la implementación de Scrum permite el uso de Backlog para gestionar las historias de usuario y tableros Kanban para apoyar el flujo de trabajo de las tareas que participan en la ejecución de Sprints. Azure DevOps permite la clasificación de componentes de trabajo por medio de historias de usuario, tareas y bugs.

Los repositorios de Azure DevOps son un componente fundamental para controlar los cambios en el código y poder llevar a cabo la integración continua. Estos repositorios se pueden conectar de manera sencilla a los pipelines de integración por medio de una interfaz gráfica. De igual forma, los repositorios poseen las características de implementación de ramas, control de fusión de ramas y gestión de cambios de código fuente. Además, los repositorios de Azure DevOps permiten controlar las modificaciones del código para cada funcionalidad, al mismo tiempo que pueden integrarse con los requerimientos de los tableros. La integración entre los repositorios y los tableros de Azure DevOps facilita la trazabilidad del código con cada requerimiento implementado.

Los pipelines de Azure DevOps son utilizados para configurar y gestionar flujos de trabajo con tareas como la compilación del código, la construcción de ejecutables y ejecución de pruebas automatizadas. Los pipelines se pueden configurar y verificar su ejecución en una interfaz gráfica.

Una característica importante de los pipelines de integración continua de Azure DevOps es la facilidad de integrar dependencias o componentes. Azure DevOps proporciona diversos componentes ya implementados que permiten instalar dependencias o paquetes necesarios de manera sencilla. En otras palabras, Azure Devops ahorra tiempo y esfuerzo en la búsqueda de paquetes para el funcionamiento de dependencias o paquetes necesarios.

De igual manera, Azure DevOps permite la creación de casos de prueba y los agrupa en planes de prueba. Este módulo de Azure DevOps está enfocado en asegurar la calidad del Software. En cada caso de prueba Azure DevOps permite agregar entradas, salidas e instrucciones específicas para probar el correcto funcionamiento del software.

2 METODOLOGÍA

Estudio de Herramientas

El estudio de herramientas que soportan el marco Scrum fue complejo debido a la diversidad de herramientas disponibles. El estudio centró sus esfuerzos en buscar herramientas que apoyen la gestión del Product Backlog, la estimación de las historias de usuario, las Daily Scrum y las Sprint Retrospectives.

El Scrum Team fue integrado por el Prof. Julio Sandobalín, Ph.D. como Product Owner, el Prof. Carlos Iñiguez, Ph.D. como Scrum Master y por David Aguirre, César Santacruz, Daniel Velasteguí y Christian Morán, estudiantes de la carrera de Ingeniería en Software con el rol de Scrum Developers.

Al seleccionar las herramientas para la gestión del Product Backlog, se tuvieron en cuenta los siguientes criterios: la cantidad de usuarios, el tipo de licenciamiento, la capacidad de almacenamiento, la integración con el sistema de control de versiones y la experiencia previa del Scrum Team.

La Tabla 1 muestra un cuadro comparativo de herramientas para gestionar el Product Backlog. Se seleccionó Azure DevOps para gestionar el Product Backlog debido a la experiencia del equipo y la integración con las herramientas de desarrollo de software.

Tabla 1. Cuadro comparativo de herramientas para gestión del Product Backlog.

Criterios\Herramientas	GitLab	Azure DevOps	Jira	Taiga
Cantidad de Usuarios	5	5	10	15
Licenciamiento	Open Source	Private	Open Source	Open Source
Capacidad de Almacenamiento (GB)	5	2	2	0.3
Experiencia del equipo	Sí	Sí	Sí	No
Integración con SCV	Sí	Sí	Sí	No

De igual manera, los criterios para seleccionar la herramienta para el Daily Scrum fueron: el licenciamiento, tiempo de duración de las reuniones, consumo de recursos, almacenamiento de grabaciones, experiencia del equipo e integración con calendario. La Tabla 2 muestra el cuadro comparativo de herramientas para el Daily Scrum. Como resultado, se seleccionó Microsoft Teams debido a la experiencia del equipo y a sus licencias académicas de uso.

Tabla 2. Cuadro comparativo de herramientas para Daily Scrum.

Criterios\Herramientas	Microsoft Teams	Google Meets	Zoom
Licenciamiento	De Estudiante	Gratuito	Gratuito
Duración de reuniones sin costo(minutos)	60	60	40
Capacidad de Almacenamiento (GB)	1000	15	0
Experiencia del equipo	Sí	Sí	Sí
Integración con Calendario	Sí	Sí	Sí

Las herramientas para las Sprint Retrospectives fueron Click Up, Miro y Mural. De estas tres herramientas se seleccionó a Miro principalmente por el tema de la experiencia previa de los integrantes del Scrum Team. En la Tabla 3 se aprecia una comparación entre las características de estas herramientas.

Tabla 3. Cuadro comparativo de herramientas Sprint Retrospective

Criterios\Herramientas	ClickUp	Miro	Mural
Licenciamiento	Gratuito	Gratuito	Gratuito
Cantidad de Usuarios	Ilimitado	Ilimitado	Ilimitado
Servicio	Espacio de trabajo de tableros y tareas	Un espacio de trabajo y tres pizarras	Tres murales
Experiencia del equipo	No	Sí	No

A pesar de que el Agilismo promueve la comunicación frente a frente (Fowler & Highsmith, 2001), se optó por realizar las Daily Scrum de manera virtual para facilitar los horarios de conectividad de los integrantes del Scrum Team. Al tomar la decisión de la hora para las reuniones se notó que la disponibilidad de horarios de los integrantes del equipo se limitaba por temas de trabajo, transporte y estudios. Esta situación obligó al equipo a escoger el horario de la reunión a las 21:00 horas de lunes a viernes.

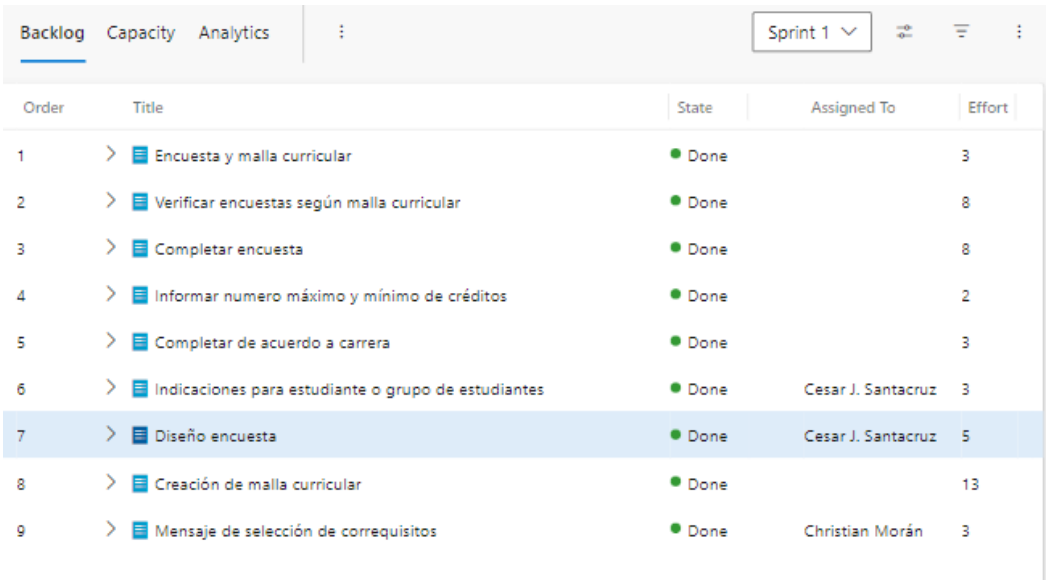
Microsoft Teams y Azure DevOps se convirtieron en las herramientas seleccionadas para implementar las Daily Scrum y gestionar el Product Backlog. De igual forma, la aplicación web Planning Poker se utilizó en las Sprint Planning para la estimación de las historias de usuario, así como la aplicación Miro se utilizó para las Sprint Retrospectives. Finalmente, la Sprint Review se la realizó mediante una presentación del incremento de la aplicación web al Product Owner en una sala de audiovisuales de la Facultad de Ingeniería de Sistemas FIS.

En conclusión, las herramientas implementadas para gestionar la construcción del sistema fueron: Microsoft Teams, Azure DevOps, la aplicación web Planning Poker y Miro. Las Daily Scrum se implementaron con Microsoft Teams; para la gestión del Sprint Planning, Producto Backlog y Sprint Backlog se utilizó Azure DevOps; para la estimación de historias de usuario se utilizó la aplicación web Planning Poker; y para las Sprint Retrospectives se optó por la aplicación Miro.

Sprint Uno: Elaboración de Formularios por Carrera y Malla Curricular

El Product Owner estableció como objetivo del Sprint la elaboración de formularios por carrera y por malla curricular. El sistema debía mostrar un formulario basado en una carrera y malla curricular respectiva, considerando que los datos de prueba relacionados con carrera y malla debían ser cargados por los Scrum Developers.

En base al objetivo propuesto por el Product Owner, se seleccionaron un conjunto de historias de usuario que fueron escritas en el Backlog de Azure DevOps para ser implementadas en el Sprint. De igual forma, el Sprint se configuró en Azure DevOps con una duración de diez días. Las historias seleccionadas se aprecian en Figura 1.



Order	Title	State	Assigned To	Effort
1	Encuesta y malla curricular	Done		3
2	Verificar encuestas según malla curricular	Done		8
3	Completar encuesta	Done		8
4	Informar numero máximo y mínimo de créditos	Done		2
5	Completar de acuerdo a carrera	Done		3
6	Indicaciones para estudiante o grupo de estudiantes	Done	Cesar J. Santacruz	3
7	Diseño encuesta	Done	Cesar J. Santacruz	5
8	Creación de malla curricular	Done		13
9	Mensaje de selección de correquisitos	Done	Christian Morán	3

Figura 1. Product Backlog del Sprint 1

Para este Sprint el esfuerzo estimado total fue de 48 story points. La estimación de cada historia de usuario se realizó con la aplicación web de Planning Poker. Para ello, se implementó un nuevo juego de Planning Poker con la serie de Fibonacci como escala y se envió un código para que los Scrum Developers se unan al juego.

Debido a la falta de comunicación entre los Scrum Developers y el Product Owner, la ejecución del Sprint no centro sus esfuerzos en generar valor al usuario mediante el

desarrollo del formulario por carrera y malla curricular. Por lo tanto, en este Sprint el equipo no fue transparente con las dudas sobre el desarrollo del sistema.

Durante el primer Sprint, el equipo presentó dificultades para adaptarse a los valores y pilares del marco Scrum. Además, la inspección y adaptación fueron deficientes debido a la falta de experiencia de los Scrum Developers con el marco Scrum.

En las Daily Scrum se presentaron problemas de comunicación entre los Scrum Developers. Cada integrante mencionaba de manera superficial en qué se encontraba trabajando, pero no existía la trazabilidad con el Sprint Backlog. En este escenario, se perdió el sentido de las Daily Scrum, que es la relación con la inspección y adaptación que sostiene Scrum. Para solucionar este problema se acordó mencionar el número de Azure DevOps de la tarea (recuadro rojo de Figura 2) que se está implementando (o que se implementó) para guiar la inspección, adaptación y facilitar la transparencia.

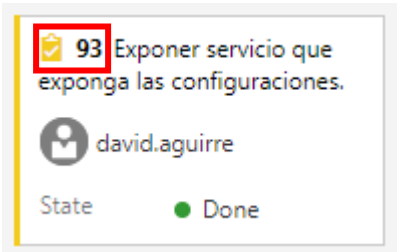


Figura 2. Ejemplo de tarea con su número respectivo

El primer Sprint sirvió para que el equipo reflexione sobre los valores, los pilares y los principios del agilismo y de Scrum. Al llegar al final del Sprint el equipo detectó los problemas de comunicación y falta de experiencia con el marco Scrum. Para solucionar estos problemas el Scrum Master y el Product Owner comprometieron al Equipo de Desarrollo a vivir los valores y los principios de Scrum. Además, se incentivó al equipo para que la implementación del incremento se la realice en base a los criterios de aceptación establecidos en las historias de usuario.

La Tabla 4 contiene la definición de terminado (Definition of Done) del primer Sprint. El equipo determinó antes del Sprint Planning los elementos que van a componer la definición de terminado. De esta manera, se esperaba que los Scrum Developers obtengan el incremento del producto basándose en cada elemento de la definición.

Tabla 4. Definición de Terminado de Sprint 1

Definición de Terminado	
Historias de Usuario	Se cumple con criterios de aceptación
Desarrollo	Verificar estructura de commit
	Commits descriptivos
	Código comentado
	Código refactorizado
Pruebas	Ejecutar pruebas unitarias
	Ejecutar pruebas end to end
Repositorio	Los cambios en Git se probaron en rama Dev
	Realizar un pull request a Dev por cada funcionalidad nueva
	Verificar que todos los conflictos estén solucionados
Documentación	Se actualiza documentación de API
Producción	Se realiza un pull request desde dev a main

Al llegar a la Sprint Review hubo una demora por parte de los Scrum Developers debido a la falta de optimización de la integración continua. Por tal motivo, fue necesario empezar la presentación del incremento funcional de manera local, mientras se ejecutaba el despliegue en producción. Por supuesto, el Product Owner manifestó su disconformidad por la demora, ocasionando que el Scrum Master comprometa a los Scrum Developers a llegar a la Sprint Review con el sistema funcional. De igual manera, los Scrum Developers se comprometieron a optimizar el tiempo de la integración continua para evitar este tipo de inconvenientes.

La Sprint Review sirvió para que el equipo de desarrollo adquiera mayor conocimiento sobre el sistema que esperaba el Product Owner. Al ser el primer uso del sistema, el Product Owner resaltó observaciones de la interfaz visual para los estudiantes, propuso parámetros de configuración y restricciones del sistema. Hasta ese momento, el sistema desplegaba la malla curricular de un estudiante de software o de computación y validaba los créditos, prerrequisitos y correquisitos de cada asignatura. El incremento del sistema se puede apreciar en Figura 3.

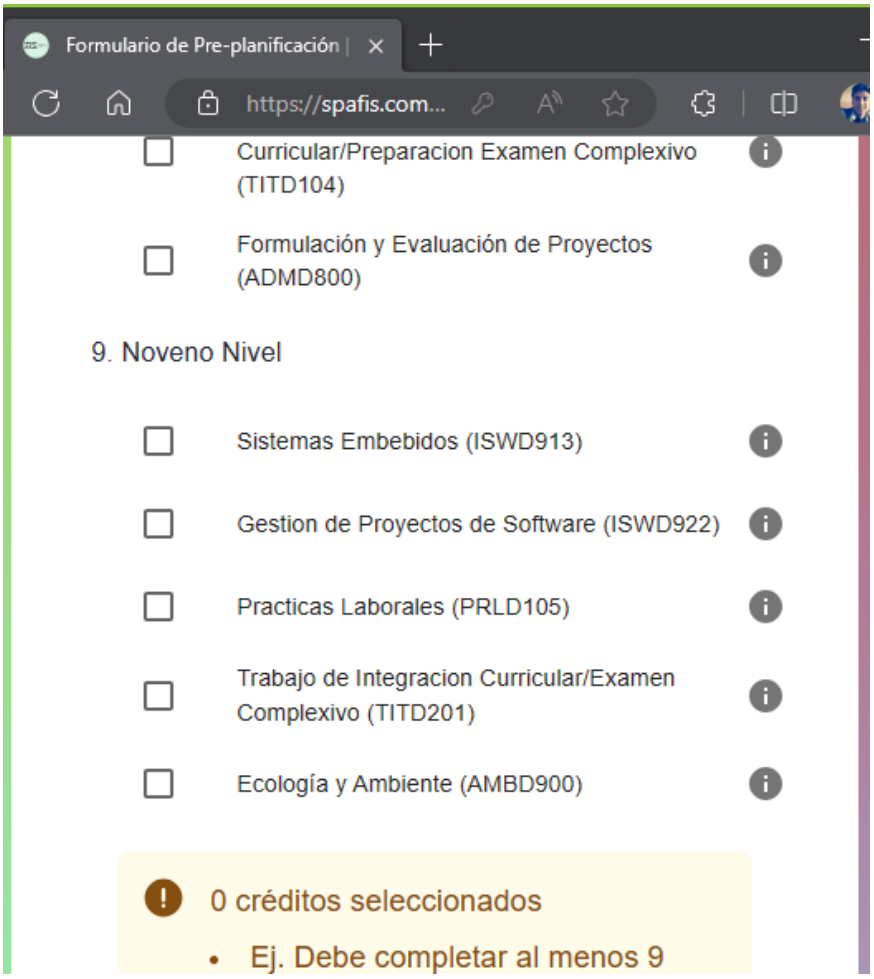


Figura 3. Incremento Funcional de Sprint 1

Finalmente, la Sprint Retrospective se realizó utilizando la aplicación llamada Miro. En la Sprint Retrospective se comprometió a los Scrum Developers a vivir los valores y los principios de Scrum. Para ello el equipo debía comunicar los avances con la trazabilidad al Product Backlog en las Daily Scrum.

Igualmente, el enfoque del equipo debía centrarse en cumplir el objetivo propuesto por el Product Owner, mientras que la definición de terminado debía ser actualizada para asegurar la calidad del incremento. De igual manera, la centralización de la información en una Wiki fue otra sugerencia de mejora para tener en cuenta, así como mantener el compromiso de preparación previa para la Sprint Review.

Por otra parte, en la Sprint Retrospective existieron observaciones en cuanto a la forma de redacción y en cuanto a la especificación de cada elemento de la definición de terminado. Por ejemplo, para validar el código refactorizado se debía especificar la forma de verificar que se ha refactorizado. Por tal motivo, la definición de terminado tuvo que ser actualizada para el siguiente Sprint.

Sprint Dos: Personalizar Formularios por Estudiante

El objetivo del segundo Sprint fue la personalización de la encuesta por cada estudiante. El Product Owner explicó que el incremento funcional debía permitir que una encuesta se adapte a las condiciones específicas de un estudiante. Por ejemplo, para cualquier estudiante, la encuesta debía validar que seleccione solamente las asignaturas que aún no aprueba y aquellas que está cursando, en el caso de realizar segunda matrícula. Además, la encuesta tenía que validar las restricciones de cada asignatura, lo que incluye a los prerrequisitos, correquisitos, pisos y techos. En este contexto, el sistema debía presentar el formulario según las materias que el estudiante ya había aprobado, las asignaturas pendientes y las restricciones de mínimo de créditos por asignatura.

Con el objetivo propuesto para el segundo Sprint, los Scrum Developers se reunieron utilizando la herramienta Microsoft Teams para seleccionar las historias de usuario que se implementaron en ese Sprint. En la Figura 4 se aprecia el Sprint Backlog de Azure DevOps con las respectivas historias de usuario.

El esfuerzo del Sprint fue de 25 story points, es decir, hay una diferencia de 23 story points con el esfuerzo realizado en el primer Sprint. Es importante mencionar que en el segundo Sprint la estimación de historias de usuario se realizó utilizando la experiencia del primer Sprint y mejorando la experiencia de estimar como un equipo.

La comunicación frecuente y el compromiso de los Scrum Developers se alinearon para completar el objetivo propuesto por el Product Owner. De este modo, los Developers mantuvieron el enfoque en el objetivo por medio de una comunicación frecuente con el Product Owner. Utilizando Microsoft Teams, el Product Owner aclaraba dudas propias del negocio sobre la actualización de la encuesta y las restricciones de créditos.

La diferencia entre la estimación del primero y segundo Sprint se debe también a la falta de experiencia en la estimación las historias de usuario. Esta falta de experiencia suele ocasionar que las historias sean subestimadas (Liskin et al., 2014). De esta manera, la diferencia entre la estimación del primer Sprint y el segundo Sprint se puede explicar por una sobreestimación de las historias de usuario.

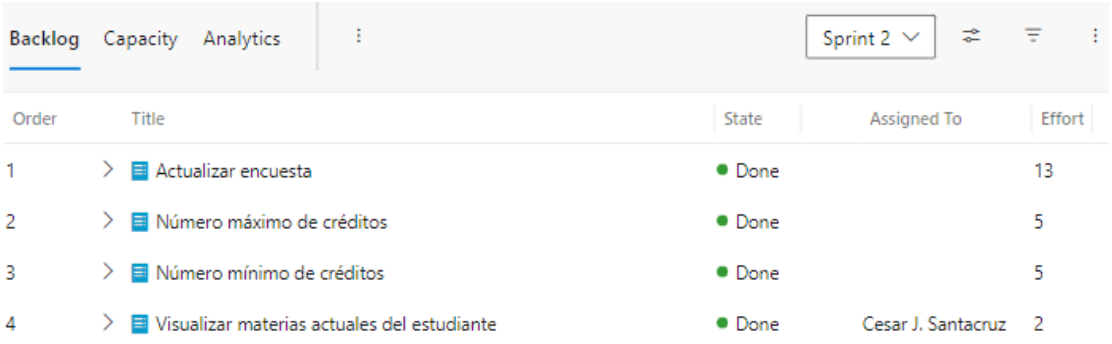


Figura 4. Sprint Backlog 2

Durante las Daily Scrum el equipo de desarrollo comunicaban sus avances basándose en las historias de usuario y sus tareas técnicas. De esta forma, se reflejaba la trazabilidad de los avances diarios del equipo en el Product Backlog. El equipo de desarrollo conocía las tareas exactas que se estaban ejecutando, adoptando así la inspección que se requiere en las Daily Scrum. No obstante, el equipo avanzaba una tarea durante dos o tres días debido a la disponibilidad de tiempo de los integrantes.

Aunque el equipo haya adoptado la comunicación, el compromiso y el enfoque de Scrum, los problemas persistían debido a la falta de comunicación y sinceridad con los obstáculos presentes. La importancia de reconocer obstáculos y comunicarlos al equipo en las Daily Scrum se evidenció en el retraso de la implementación de las tareas técnicas. Esta falta de honestidad con los obstáculos del equipo no permitía que el Scrum Master actúe para solucionar estos problemas.

La apertura al cambio del equipo se evidenció con la inclusión de la visualización de materias actuales de un estudiante en un formulario. Esto fue una demostración de aceptación al cambio por parte de los Scrum Developers para asegurase de seleccionar todas las historias de usuario que cumplan con el objetivo del Sprint. De esta manera, el equipo demostró el interés por construir el producto deseado por el Product Owner.

La falta de transparencia permaneció durante el segundo Sprint porque aún no se evidenciaba una centralización de la información del proyecto. A pesar de que en el primero Sprint se sugirió la implementación de una Wiki, el equipo continuó manejando la información del proyecto sin centralizarla. Por lo tanto, la transparencia del equipo no se fortaleció en el segundo Sprint.

A pesar del problema de la disponibilidad de tiempo de los Scrum Developers, se demostró su compromiso cuando se obtuvo el incremento funcional al finalizar el Sprint. Incluso,

cuando las historias de usuario se completaron el día anterior de finalizar el Sprint, se implementaron todas las previstas y se verificaron por medio de los criterios de aceptación.

La definición de terminado es otro de los puntos que los Scrum Developers no consiguieron completarla de la forma esperada. En la Tabla 5 se aprecia que en el segundo Sprint la definición de terminado no tiene ninguna especificación de un estándar para comentar el código y cada elemento de la lista está redactado de una forma superficial. En otras palabras, la definición de terminado no está redactada en una forma en que cada elemento puede ser revisado y marcado al final como completado.

Tabla 5. Definición de Terminado de Sprint 2

Definición de Terminado	
Desarrollo	Cammel Case en variables y métodos
	Código comentado
Pruebas	Pruebas unitarias
	Pruebas end to end
	Pruebas de regresión
Git	Rama de desarrollo actualizada
	Rama de producción actualizada
Pipelines	Pipelines de instalación de dependencias y pruebas
	Pipelines de optimización de build y release
Documentación	Documentación API actualizada
Producción	Deploy Satisfactorio
	Criterios de Aceptación

En comparación con la lista de verificación del primer Sprint se percibe un mayor desglose de elementos, tales como: el sistema de control de versiones, la documentación detallada y la actualización del sistema.

Un detalle de importante es que la documentación que se aprecia en la definición de terminado del segundo Sprint no se encontraba centralizada. En otras palabras, la documentación no era transparente para el equipo. En este escenario, la documentación no era accesible para todo el equipo y de esa manera se evidenciaba la ausencia de transparencia.

Anterior a la Sprint Review, los Scrum Developers realizaron la validación del sistema con la definición de terminado. De la revisión realizada se obtuvo que el elemento que no se cumplió totalmente fue aquel que menciona las pruebas de regresión.

El equipo decidió no automatizar las pruebas de regresión, sino crear tres casos de prueba que se debían ejecutar manualmente en cada cambio realizado. Los casos de prueba se cumplieron satisfactoriamente cuando las historias de usuario se implementaron

totalmente. Sin embargo, la ejecución de los casos de prueba durante cada cambio consumía al menos una hora, lo cual retrasaba el avance del equipo.

Es importante considerar que estos casos de prueba requerían de datos de prueba cargados en el sistema. De igual manera, los datos de prueba debían ser accesibles a todos los miembros del equipo. No obstante, los datos de prueba no se compartieron durante este Sprint, lo cual es otra evidencia de la falta de transparencia.

Finalmente, llego el momento de la Sprint Review. El incremento funcional presentado estaba adaptado a la personalización del formulario para la situación particular de cada estudiante. En este punto fue necesario utilizar datos de pruebas que permitan evidenciar que el formulario se adapta a las asignaturas de cada estudiante. No obstante, el Product Owner realizó observaciones sobre la forma de ingreso de estudiantes. Entonces, los Scrum Developers intervinieron para aclarar que hasta ese momento no se había negociado el tema del inicio de sesión. En consecuencia, como una adaptación se llegó al acuerdo de que el ingreso de estudiantes se realizará utilizando el email institucional y una contraseña. El incremento presentado se puede apreciar en Figura 5.

Como consecuencia de la Sprint Review, se generó una historia de usuario para atender al requerimiento de ingreso de los estudiantes a la encuesta. Adicionalmente, el equipo de desarrollo, el Product Owner y el Scrum Master acordaron que en ese momento el inicio de sesión se lo realizará solamente con el email institucional de cada estudiante y con una contraseña. En otras palabras, este requerimiento no incluía el manejo de sesiones y de autenticación. La razón por la que no se implementaron sesiones es porque el tiempo para acceder a la autenticación de la universidad es mínimo de dos meses y el desarrollo no podía estancarse durante ese período de tiempo. Para tener en consideración, el tema de la autenticación se lo postergó para el quinto Sprint.

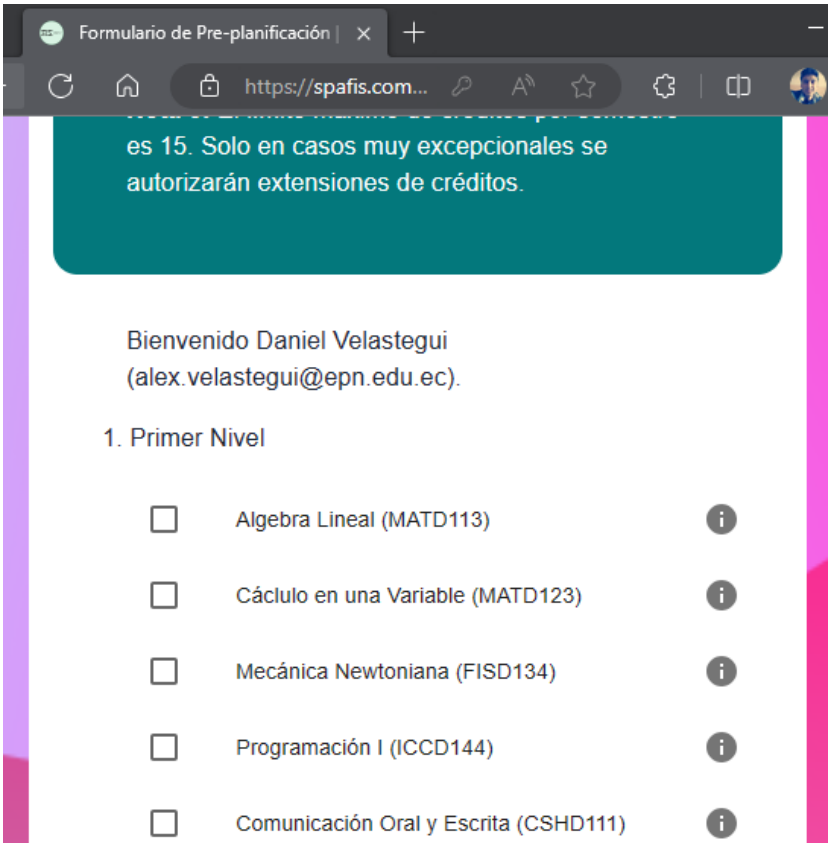


Figura 5. Incremento Funcional de Sprint 2

Durante la Sprint Retrospective, el equipo de desarrollo se comprometió a generar la Wiki y cargar la información importante para el equipo. Por ejemplo, los pasos para replicar la integración continua, enlaces para acceder al sistema de preplanificación, definición de terminado y objetivos de los Sprints. La Wiki se utilizaría para adoptar la transparencia entre el equipo y compartir el conocimiento.

Sprint Tres: Elaborar Formularios de Administración

Una reestructuración del backend fue el reto del equipo para el tercer Sprint. Esta reestructuración ocasionó que el equipo seleccionará solamente dos historias de usuario (ver Figura 6) para el tercer Sprint. La explicación de la reestructuración se debe a que anteriormente el sistema estaba limitando al usuario a utilizar solamente una estructura fija para las asignaturas. El equipo de desarrollo decidió realizar este cambio para que el usuario pueda manejar la información de las asignaturas de manera flexible.

El objetivo del tercer Sprint sobre la elaboración de formularios de administración tuvo la intención de aligerar el trabajo para que los Scrum Developers se centren en la reestructuración. Mientras tanto, dos de los Developers se dedicarían a la implementación de las interfaces para los administradores. En este punto, el Product Owner solicitó las

pantallas con la funcionalidad requerida, es decir, aún no se conversó sobre los roles que accederán al sistema.

Considerando esta reestructuración del backend, los Scrum Developers estimaron la historia de usuario con 21 story points. El Sprint Backlog se puede apreciar en Figura 6.

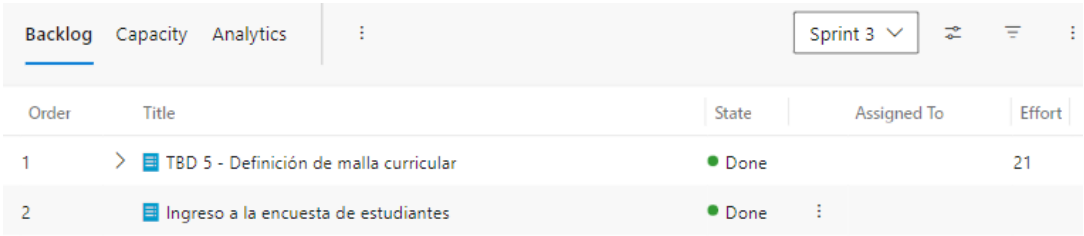


Figura 6. Sprint Backlog 3

Las Daily Scrum en Microsoft Teams durante el tercer Sprint fueron esenciales. En cada reunión, el equipo comunicaba los avances con trazabilidad al Sprint Backlog y siguiendo el orden establecido. Luego de cada Daily, los Scrum Developers que debían resolver errores permanecían conectados en la reunión de Microsoft Teams. En otras palabras, cada reunión diaria sirvió para inspeccionar el trabajo y adaptarse en lo que se requiera.

El equipo demostró la apertura al cambio y el coraje en el tercer Sprint. El cambio total del de la estructura del sistema comprometió al equipo a realizar el menor impacto posible para que el sistema permanezca estable y no cambiar su comportamiento. Para conseguirlo, el equipo realizó un estudio del código para modificar solamente las instrucciones necesarias. Por supuesto, realizar el estudio del código le tomó al equipo al menos unas 10 horas. Por lo tanto, la confrontación del reto de la reestructuración del sistema demostró el coraje y la apertura al cambio adoptados por el equipo.

La comunicación en este Sprint fue la protagonista para la implementación de las historias de usuario planificadas. Cada integrante del equipo debía comunicar las fallas en los métodos de las clases, la actualización de los servicios y la forma de consumir cada servicio para solventar los problemas presentados debido a la reestructuración.

En el tercer Sprint el equipo aprendió en base a la experiencia que la documentación es importante mantenerla actualizada. Aunque la comunicación se fortaleció, el equipo mostró falta de compromiso con la actualización de la documentación. La información errónea del consumo de los servicios y los ejemplos de los resultados esperados ocasionaron la falla de la creación de asignaturas para un período. La falta de actualización de la

documentación obligó al equipo a realizar reuniones virtuales en Microsoft Teams de al menos dos horas para solucionar los errores.

A excepción del compromiso con la documentación, el equipo demostró un evidente cambio en cuanto a los valores y principios de Scrum. La comunicación entre el equipo se tornó productiva; el equipo se comprometía con la implementación de funcionalidades y solución de errores; el equipo aceptó una reestructuración del sistema y permaneció enfocado en lograr la elaboración de mallas curriculares. Adicionalmente, cada integrante mostró absoluto respeto por el resto del equipo, puesto que no se criticó por los errores cometidos, sino que se buscaron soluciones.

A pesar de que los Scrum Developers intentaron cambiar lo mínimo del código, el impacto en el sistema fue inevitable. La comunicación de los errores era continua durante el transcurso del Sprint, pero eso fortaleció al Scrum Team para comprometerse con la entrega funcional y verificar el cumplimiento de los criterios de aceptación.

La definición de terminado para el tercer Sprint (ver Tabla 6) contiene una redacción con mayor explicación en comparación con la segunda versión. El equipo llegó al acuerdo de que cada elemento debía entenderse como una pregunta que puede responderse como sí o no.

En el tercer Sprint el equipo no tenía un estándar de comentarios adecuado para verificar que se cumpla este elemento de la definición de terminado. La razón de que los Scrum Developers no tengan un estándar de comentarios es la falta de transparencia de sus integrantes. De esta manera, los Developers debían definir una estructura para realizar comentarios en el código.

Tabla 6. Definición de Terminado de Sprint 3

Definición de Terminado	
Desarrollo	Cammel Case en variables y métodos
	Código incluye comentarios
Pruebas	Pruebas unitarias aprobadas satisfactoriamente
	Pruebas end to end aprobadas satisfacotriamente
	Pruebas de regresión
Git	Rama de desarrollo actualizada
	Rama de producción actualizada
Pipelines	Ejecución de pipelines de instalación de dependencias y pruebas finalizado
	Pipelines de optimización de build y reléase finalizado
Documentación	Documentación API Swagger actualizada
Producción	Despliegue en producción exitoso

Por otro lado, en la definición de terminado se consideró únicamente la documentación de la API del backend, es decir, los Scrum Developers no contaban con mayor documentación. Por lo tanto, el equipo de desarrollo no lograba alinearse completamente a lo que establece el agilismo (Kent Beck et al., 2001). Aunque el agilismo especifique que tiene mayor prioridad el software funcionando que la documentación excesiva, no quiere decir que no se debe documentar.

En la Sprint Review se realizó la presentación de las interfaces del panel administrativo que involucraba a la creación de períodos y mallas curriculares (ver Figura 7). Las observaciones del Product Owner se basaron principalmente en detalles sobre el flujo de creación de mallas curriculares y períodos. El problema con el flujo surgió porque el sistema consideraba la carga de datos de estudiantes y de asignaturas y no permitía visualizarlos. Sin embargo, el Product Owner solicitó que se puedan visualizar desde el sistema estos datos.

El acuerdo entre el equipo de desarrollo y el Product Owner fue que las asignaturas y los estudiantes serían cargados. Antes de llegar a un acuerdo, los Scrum Developers se reunieron para evaluar el impacto del requerimiento. El Product Owner, apoyado en la explicación de los Scrum Developers, explicó nuevamente el alcance del proyecto. Considerando el límite de tiempo para finalizar el producto software, el Product Owner acordó que por el momento no se implementaría la funcionalidad de visualización.

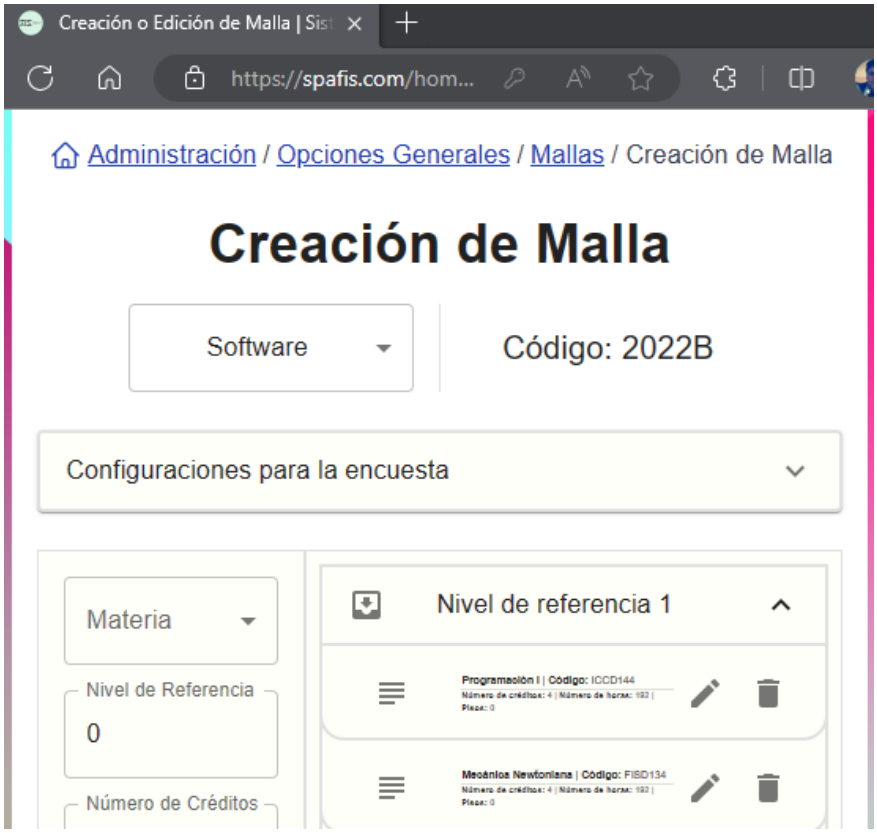


Figura 7. Incremento Funcional de Sprint 3

Otro de los problemas en la Sprint Review fue la aparición de errores en la configuración de mallas curriculares. El error que se presentaba se generaba al momento de configurar los correquisitos para una asignatura, ya que no permitía agregar más de dos correquisitos. Adicionalmente, los estudiantes no podían completar el formulario cuando involucraba a las asignaturas con más de dos correquisitos.

Finalmente, la Sprint Retrospective sirvió a los Scrum Developers para comprometerse con la generación de la Wiki para adoptar la transparencia en todas las actividades relacionadas con Scrum. Los Developers comprobaron que la Wiki era una forma de que los enlaces de acceso a los servicios, la definición de terminado, los objetivos del Sprint y los datos de prueba sean transparentes para todo el equipo. Por tal motivo, ese fue el compromiso principal para el siguiente Sprint.

Sprint Cuatro: Generar Malla Curricular por PAO

El objetivo del Sprint fue la generación de mallas curriculares por período académico ordinario PAO. Además, el Sprint se enfocó en la corrección del flujo de creación de mallas

curriculares, configuración de mensaje para la encuesta y el ingreso al sistema por roles. En este escenario, al finalizar el Sprint el sistema de preplanificación debía permitir el acceso de una autoridad al sistema mediante una sesión y configurar una malla de un período académico ordinario.

Es importante considerar que la Sprint Planning se realizó por medio de Microsoft Teams. Al estar presente todo el equipo la comunicación era fluida, se aclaró en ese momento las dudas sobre las historias de usuario junto con el Product Owner. El Scrum Master facilitaba y comprometía al equipo a mantener el principio de transparencia y sinceridad en las estimaciones para lograr asignar un esfuerzo real.

Todo el equipo seleccionó cuatro historias de usuario (ver Figura 8) para el Sprint Backlog. Estas historias de usuario se estimaron en un total de 36 story points. Las dos historias con mayor esfuerzo están relacionadas con el acceso al sistema por roles. La razón de asignar esos valores a esas historias es que el equipo debía investigar una tecnología para el manejo de las sesiones. Además de la investigación, se tenía que comprobar el funcionamiento y como se acoplará la tecnología al sistema.

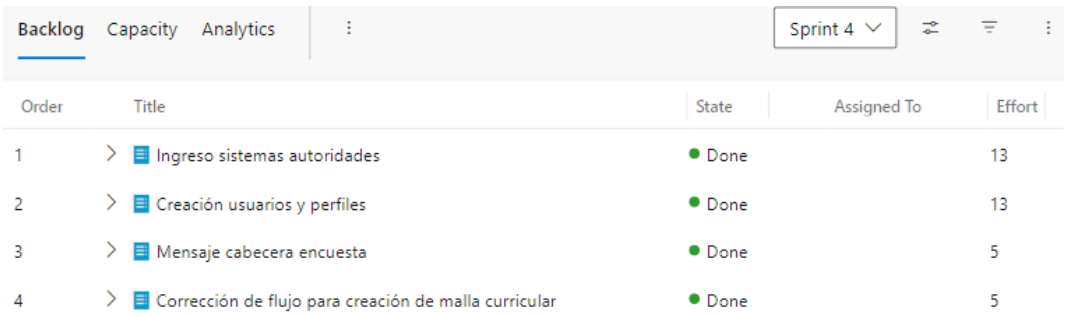


Figura 8. Sprint Backlog 4

Las tareas técnicas se crearon y se asignaron después de la Sprint Planning. Luego de realizar la planificación, los Scrum Developers se reunieron para lograr separar las historias de usuario en tareas técnicas para ingresarla en lo tableros de Azure DevOps. Cada tarea técnica se asignó a uno de los integrantes, considerando el componente en el que ha trabajado durante todos los Sprints. La asignación de tareas principalmente se la realizó por módulos como configuración de mallas curriculares, creación de períodos, activación de períodos y activación de encuesta.

Las Daily Scrum del cuarto Sprint se realizaron con la misma estructura que los anteriores. En otras palabras, cada integrante del equipo debía contestar a las tres preguntas de las Daily Scrum haciendo referencia a las tareas del Product Backlog. Incluso en dos ocasiones los Scrum Developers decidieron mantenerse en la reunión después de la

reunión diaria porque se necesitaba solucionar una duda entre el manejo de sesiones y la forma de asociar cada usuario con la sesión en la base de datos.

De esta manera, los Scrum Developers detectaron que se debía tomar una decisión con respecto al manejo de sesiones. Cuando el equipo llegó a un acuerdo sobre la forma que se mapearán las sesiones de los usuarios en la base de datos se evidenció la capacidad de adaptación del equipo.

En la implementación del Sprint los esfuerzos principales estaban enfocados en el acceso al sistema por roles utilizando sesiones y en las mallas curriculares. La comunicación entre el equipo fue fundamental para comprender los distintos roles del sistema. Además, cada rol tenía distintos permisos de acceso, por lo cual fue necesario que el Product Owner detalle cada autoridad con los permisos correspondientes.

La decisión entre los Scrum Developers y el Product Owner para resolver el acceso por roles fue la implementación de acceso por capas. De esta manera, los permisos para acceder se concederían conforme el rol del usuario tenga mayores privilegios. Específicamente, el rol coordinador tenía permiso para configuración de la encuesta; al rol autoridad se le agregó el acceso a la activación de encuesta por usuarios y el rol administrativo incluye el acceso a reportes.

El manejo de sesiones generó también un problema con la asignación de roles. En la solución de este problema fue necesario que el Scrum Master facilite una negociación con el Product Owner para acordar la forma de asignar distintos roles a un determinado usuario. El acuerdo fue que se implementaría una interfaz donde se permita al usuario administrativo cambiar el rol de un usuario específico. En este escenario, el Product Owner explicó a los Scrum Developers el acuerdo y el Scrum Master comprometió a cada integrante para realizar este esfuerzo de la asignación de roles.

Por otro lado, en este Sprint se implementó una Wiki para centralizar todo tipo de información importante relacionada con el acceso al sistema y configuraciones. La Wiki se implementó en Azure DevOps, que es la plataforma para gestionar el Product Backlog, con el fin de unificar las herramientas y no tener la información dispersa. La Wiki logró que el equipo tenga un espacio colaborativo donde podía acceder a la información relacionada con el sistema de preplanificación de asignaturas. Además, el uso de Wikis con Scrum es común como herramienta de colaboración y aprendizaje (Kudikyala & Dulhare, 2016).

Continuando con la implementación del Sprint, es importante considerar un cambio acordado por los Scrum Developers en cuanto a la forma de implementar las tareas

técnicas asignadas. El cambio afectó a la forma en que se asignaban las tareas técnicas de cada historia de usuario. En este Sprint, se consideró que las tareas se asignen diariamente. Con este enfoque de asignación de tareas, los Scrum Developers cumplían con un pequeño porcentaje del objetivo cada día. De esta forma, se consiguió que las tareas se implementen en partes hasta alcanzar los criterios de aceptación.

La definición de terminado del cuarto Sprint tuvo modificaciones respecto a la redacción de cada elemento y se agregó ciertas características importantes (ver Tabla 7). Una de las características agregadas tiene que ver con tipo de formateado del código fuente, el cual se realizó con ESLINT. En la Wiki se agregó la forma que deben tener los comentarios del código fuente. Adicionalmente, se introduce el uso de inglés y el estándar para nombrar variables y métodos.

Tabla 7. Definición de Terminado de Sprint 4

Definición de Terminado	
Desarrollo	Código formateado con ESLINT
	Código comentado
	Uso de Inglés para codificación
	Cammel Case en variables y métodos
Pruebas	Pruebas unitarias
	Pruebas end to end
Git	Rama de desarrollo actualizada
	Rama de producción actualizada
Pipelines	Ejecución de pipelines de instalación de dependencias y pruebas
	Pipelines de optimización de build y release
Documentación	Documentación API actualizada
Producción	Deploy satisfactorio
	Criterios de aceptación

Para el formato de los comentarios del código se agregó una página en la Wiki (ver Figura 9) en Azure DevOps. El equipo de desarrollo llegó al acuerdo de que se debe comentar principalmente cada función y las líneas de código que contengan validaciones, como por ejemplo la validación de correquisitos al intentar crear una asignatura. El formato requería que cada función tenga su descripción de parámetros de entrada, datos de respuesta y funcionamiento.

Ejemplo de comentario:

```
/*
*Se obtiene un estudiante por su código
*@params: código del estudiante
*@return: Promise<Response> con el estudiante
*/
getStudentsByCode(code:string):Promise<Response>{
...
...
...
}
```

Figura 9. Formato de comentario en Wiki

Antes de la Sprint Review, se ejecutó un cambio en el código para corregir errores de experiencia de usuario. Debido a este cambio la revisión se retrasó al menos treinta minutos. La enseñanza de este cambio al último momento enseñó a los Scrum Developers la importancia de una revisión general del sistema software para resolver los errores antes de la Sprint Review.

A pesar del retraso para la presentación del sistema, el Product Owner revisó las funcionalidades del sistema. Sin embargo, tres observaciones importantes se encontraron sobre la usabilidad y experiencia de usuario del sistema; bug en la configuración de mallas curriculares y lenguaje familiar con el usuario final. El incremento funcional se puede apreciar en Figura 10.

Los problemas de usabilidad y experiencia de usuario se basaron principalmente en la navegabilidad y visibilidad del estado del sistema. En funcionalidades que demoraban su ejecución y no mostraba retroalimentación sobre el estado del sistema. Por otro lado, en las funcionalidades crear período y configurar malla el Product Owner expresó que no existía confianza sobre la navegación del sistema porque no podía salir de estas interfaces para volver al menú principal.

Un bug que el Product Owner descubrió se ocasionaba al configurar una asignatura con más de dos correquisitos, ya que el sistema no registraba el cambio realizado. Al descubrir este bug se lo anotó para poder considerarlo en la planificación del siguiente Sprint. De igual manera, el Scrum Master comprometió a los Scrum Developers a resolver este defecto en el siguiente Sprint.

La observación sobre el lenguaje familiar con el usuario final se refería a que en las interfaces de configuración de mallas los campos de créditos requeridos, período y niveles

no eran familiares con el usuario. De igual manera, se consideró esta observación para completarla durante el siguiente Sprint.

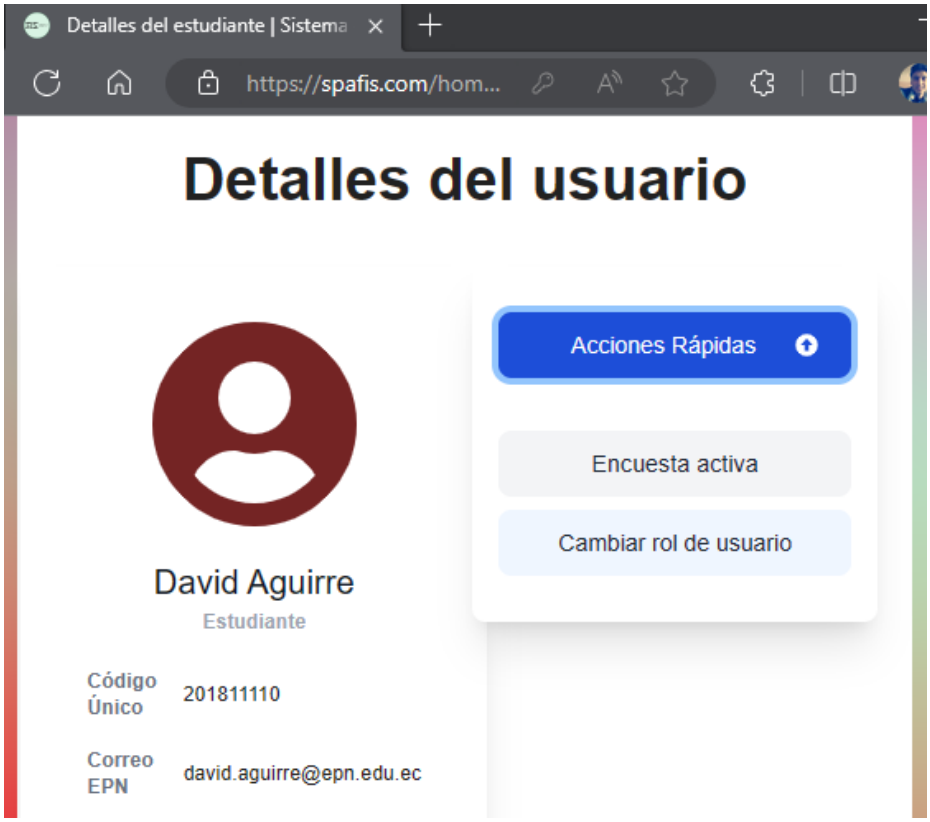


Figura 10. Incremento Funcional de Sprint 4

La Sprint Retrospective se realizó de manera presencial. El enfoque para realizar la inspección y adaptación fue que cada integrante escriba en un tablero de la herramienta Miro aquello que les gustó, lo que aprendió, la que faltó y lo que se desearía hacer.

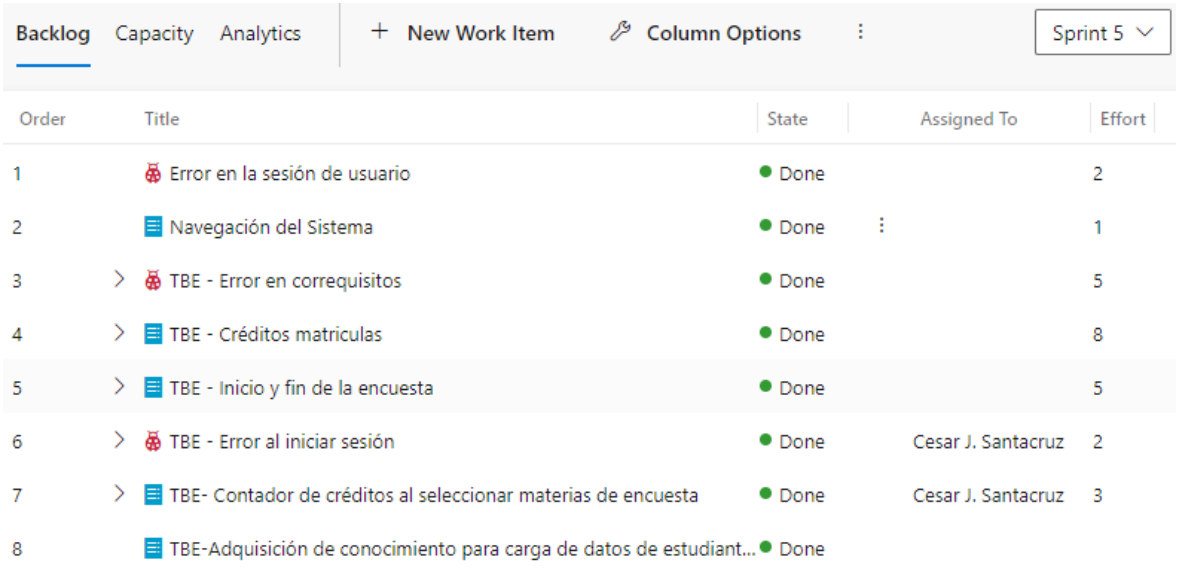
Las principales mejoras mencionadas en la Sprint Retrospective se basaron en documentación actualizada, generación de manual de usuario, comunicación de problemas y manejo de las Daily Scrum. La documentación actualizada se debía principalmente a una falta de actualización en la Wiki sobre datos de prueba de acceso al sistema. La comunicación de problemas sucedió debido a la interconexión entre interacciones del sistema, en las cuales se debían enviar parámetros. En cuanto a las Daily Scrum realizadas en Microsoft Teams, se propuso que el tiempo de la reunión permita abarcar problemas después de haber contestados las preguntas de las Daily Scrum (¿Qué hice ayer? ¿Qué voy a hacer hoy? ¿Tengo un obstáculo?)

Sprint Cinco: Corregir Errores y Mejorar Experiencia de Usuario

El quinto Sprint tiene como objetivo de mejorar la experiencia de usuario y corregir los errores encontrados. El Sprint Backlog se gestionó en los tableros de Azure DevOps (Ver Figura 11).

Los Scrum Developers estimaron las historias de usuario de manera presencial utilizando la aplicación web Planning Poker. Para ello, cada Developer seleccionaba un número acorde a la serie de Fibonacci y exponía las razones de la calificación propuesta. Cuando había conflictos por la estimación se debatía entre todo el grupo hasta llegar a un acuerdo en la cantidad de story points asignados a la historia de usuario. De igual forma, los Scrum Developers exponían las inquietudes sobre las historias de usuario y el Product Owner las resolvía.

Como se puede apreciar en Figura 11, para el quinto Sprint el esfuerzo realizado suma un total de 26 story points. Dentro del esfuerzo estimado se consideraron los bugs que se encontraron en el sistema durante el desarrollo del producto software. Adicionalmente, se seleccionaron cinco historias de usuario del Product Backlog para que sean implementadas.



Order	Title	State	Assigned To	Effort
1	Error en la sesión de usuario	Done		2
2	Navegación del Sistema	Done		1
3	TBE - Error en correquisitos	Done		5
4	TBE - Créditos matriculas	Done		8
5	TBE - Inicio y fin de la encuesta	Done		5
6	TBE - Error al iniciar sesión	Done	Cesar J. Santacruz	2
7	TBE- Contador de créditos al seleccionar materias de encuesta	Done	Cesar J. Santacruz	3
8	TBE-Adquisición de conocimiento para carga de datos de estudiant...	Done		

Figura 11. Sprint Backlog 5

Durante el Sprint, el tiempo de las Daily Scrum en Microsoft Teams fue planificado para que se puedan solventar dudas entre los Scrum Developers. El manejo de la reunión daba un espacio de tiempo para que, luego de responder las preguntas propias del evento, se pueda dialogar sobre las dependencias entre módulos del sistema. De igual manera, este

tiempo se utilizaba para hablar dudas específicas, tales como la forma de recrear los bugs para poder solucionarlos.

El problema de la recreación de los bugs obligó a adoptar el compromiso, la comunicación y el coraje de Scrum, Por medio del compromiso, cada Scrum Developer se esforzaba por recrear los bugs para poder encontrar el defecto en el código que se debía corregir. De igual manera, era importante mencionar que el equipo se comunicó constantemente por medio de un canal en Whatsapp. Una vez que se encontraba la causa de la falla, el equipo debía tener el coraje de resolver el problema para solventar la deuda técnica acumulada hasta el actual Sprint.

El Scrum Master motivó al compromiso con la solución de los errores y facilitó los medios para recrear el bug sobre los correquisitos. En el intento de recrear este bug, se utilizó Miro para realizar un diagrama conceptual de la arquitectura del sistema. Con este diagrama se facilitó la comprensión del problema para poder solventar el error.

Un entregable solicitado por el Product Owner para este Sprint fue el manual de usuario. Este requerimiento demuestra que Scrum también incluye documentación. Sin embargo, se consideró únicamente la documentación que genera valor para el usuario.

La definición de terminado del Sprint (ver Tabla 8) fue subida a la Wiki para que sea accesible para todo el equipo. La diferencia con la anterior versión de la definición de terminado se basa en la inclusión de la Wiki como parte de la documentación.

Tabla 8. Definición de Terminado de Sprint 5

Definición de Terminado	
Desarrollo	Código formateado con ESLINT
	Código comentado
	Uso de Inglés para codificación
	Cammel Case en variables y métodos
Pruebas	Pruebas unitarias
	Pruebas end to end
Git	Rama de desarrollo actualizada
	Rama de producción actualizada
Pipelines	Pipelines de instalación de dependencias y pruebas
Documentación	Documentación API actualizada
	Wiki Actualizada
Producción	Deploy satisfactorio
	Criterios de aceptación

La Wiki implementada permitía mantener la transparencia de la información en el aprendizaje de tecnologías, objetivos de cada Sprint, la definición de terminado, entre otros.

Cualquier información que permitía al equipo entender el negocio o que ayudaba a la implementación del sistema se colocaba en la Wiki.

Para la Sprint Review, los Scrum Developers acordaron que luego de la Daily Scrum anterior al evento, se debía verificar que todas las funcionalidades previstas se hayan implementado. Con esta verificación, los Developers estarían preparados para presentar el incremento funcional al Product Owner y demostrar el cumplimiento de los criterios de aceptación y la definición de terminado.

El incremento funcional del Sprint (ver Figura 12) se realizó sin inconvenientes y de manera consecuente, el Product Owner pudo realizar sus observaciones en cuanto a los tipos de mensajes y el error en correquisitos que aún no se había resuelto. De manera específica, el Product Owner detalló que los mensajes debían mostrarse con símbolos que permitan identificar si es error, advertencia o informativo. Por otro lado, el error en correquisitos generó que la encuesta no sea accesible cuando se configuraban varios correquisitos para una asignatura. Tanto el error en correquisitos como la visualización de tipos de mensajes se anotaron para el siguiente Sprint.

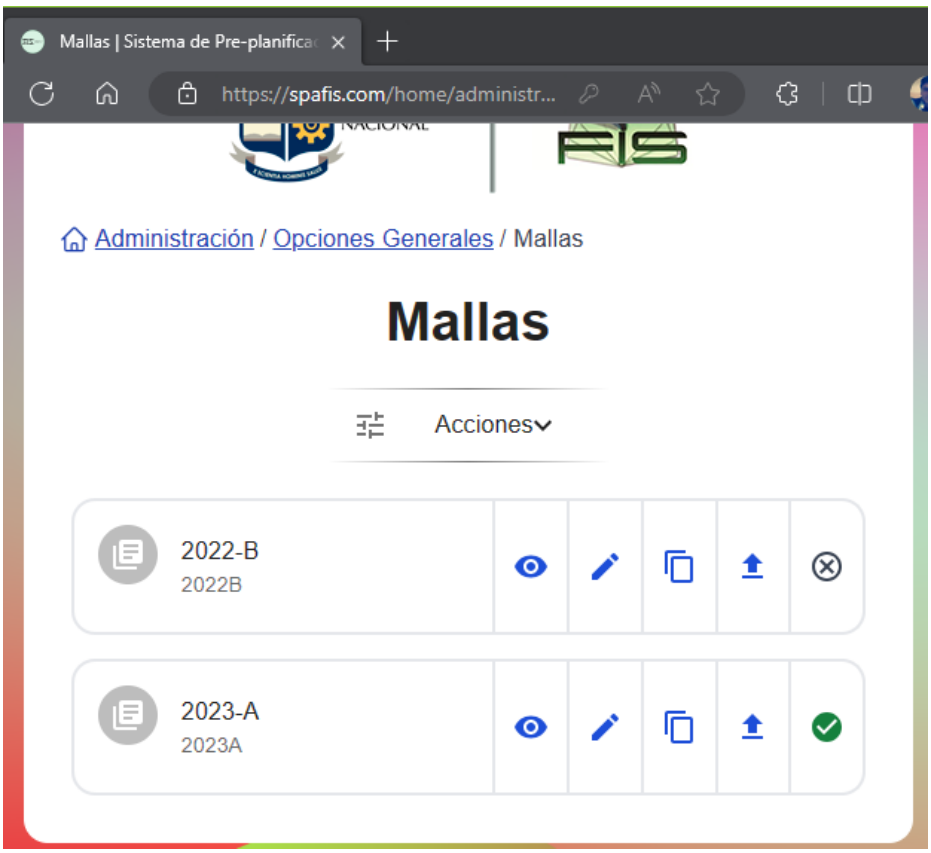


Figura 12. Incremento Funcional de Sprint 5

El Scrum Master realizó la Sprint Retrospective con la herramienta Miro, generando un tablero compartido con el equipo. Esta forma de realizar la retrospectiva del Sprint ayudó a que el equipo se sienta libre de expresar opiniones sin temor a ser observado. Es importante considerar que cuando una persona debe pararse y expresar su opinión siente miedo debido a que todo el enfoque está sobre la persona. Por otro lado, al realizarlo en tableros virtuales todos los integrantes pueden escribir al mismo tiempo y no sentirse observados.

La dinámica de la Sprint Retrospective contenía cuatro puntos principales que se trataban de cosas que debíamos empezar a hacer, parar de hacer, hacer diferente y hacer con mayor frecuencia, cada una con su propio color en la herramienta Miro. Cada punto permitió al equipo inspeccionar el proceso de tal manera que el equipo se adapte a la evolución del producto software, y que reflexione sobre sus cualidades y debilidades como equipo. Por ejemplo, se debía parar de llegar tarde a las Daily Scrum, así como el equipo se dio cuenta que se debía actualizar la documentación con mayor frecuencia.

La Sprint Retrospective generó observaciones principalmente en la usabilidad y pruebas del sistema. Aunque el sistema era funcional, la configuración de mallas aún era compleja de usar debido a la cantidad de información que contenía. De la misma manera, se sugirió que se debe empezar a realizar pruebas del sistema de manera reiterada, ya sea por casos de prueba o con pruebas automatizadas.

Entre las cosas que debemos dejar de hacer se consideró la falta de transparencia de los Scrum Developers, ya que no se compartían el conocimiento sobre las tecnologías y sobre los problemas de comprensión del negocio. Como consecuencia de esta falta de transparencia un problema de solucionar en dos horas se convertía en 2 días de trabajo.

Otro punto que se debía parar de hacer es respetar el tiempo establecido por las Daily Scrum y dejar las pruebas para el final. Cuando uno de los integrantes llegaba tarde a la reunión diaria, el equipo debía esperar por lo menos cinco minutos para comenzar la reunión. Esta tardanza de realizar la reunión diaria bloqueaba otras tareas personales de los demás integrantes del equipo. Por otro lado, la procrastinación de las pruebas funcionales del sistema ocasionaba que se encuentren fallas en la interfaz gráfica o en la lógica del sistema, lo cual se debía solventar con anterioridad.

Entre las cosas que se podían hacer diferente resaltaron el manejo de los tiempos de las Daily Scrum, así como el desarrollo basado en diseño de interfaces y pruebas. De manera general, la optimización del tiempo de las Daily Scrum buscaba que los Scrum Developers

sean claros y concisos al momento de expresar los avances. Por otro lado, se sugirió que el desarrollo de las interfaces se lo implemente en base a un diseño previo.

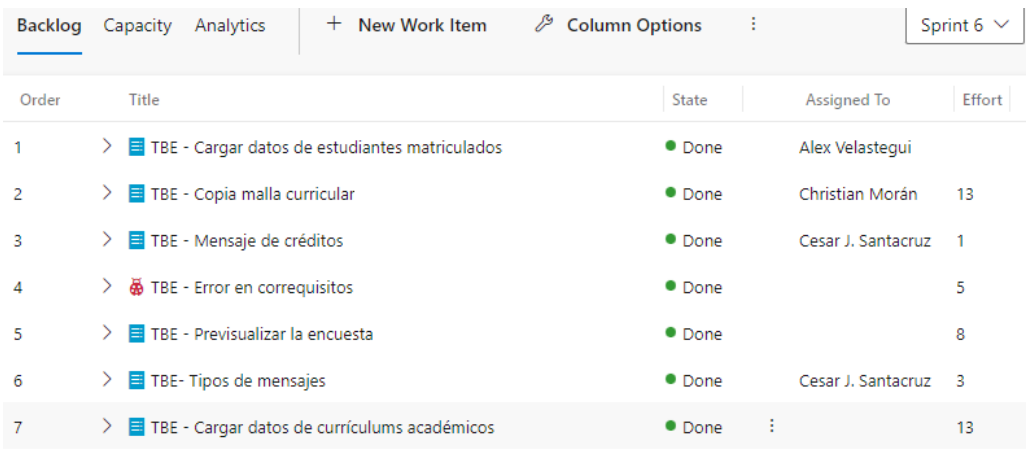
Entre aquellos elementos que se deben realizar con mayor frecuencia se destacaron la gratificación del esfuerzo realizado, compartir el conocimiento y las pruebas de funcionalidad. En primer lugar, el Scrum Master mencionó que es importante el gratificar el esfuerzo realizado para que el equipo se mantenga motivado y enfocado en el objetivo. De igual manera, el compartir el conocimiento es fundamental para que se cumpla la transparencia de Scrum. Las pruebas de funcionalidad permiten detectar los errores en etapas tempranas del Sprint para poder solventarlos con anterioridad.

Sprint Seis: Carga de Curriculums Académicos de Estudiantes

El sexto Sprint tenía como objetivo la carga de información de estudiantes y mallas curriculares. Además, se solicitó la mejora de la usabilidad del producto software por falencias en las pantallas de la configuración de mallas curriculares.

La Sprint Planning se realizó en dos partes, la estimación de las historias de usuario y la separación de tareas técnicas. La estimación de las historias de usuario se realizó de manera presencial por medio de la aplicación web Planning Poker, obteniendo como resultado un total de 51 story points. Mientras tanto, la separación de tareas técnicas se realizó de manera online con una reunión en Microsoft Teams. El Sprint Backlog se lo puede apreciar en Figura 13.

Durante la Sprint Planning el equipo acordó un cambio de horario de las Daily Scrum, debido a ciertas actividades personales que coincidían con el anterior horario. Desde entonces se configuró la reunión diaria en Teams para que inicie a las 19:30 de lunes a viernes.



Order	Title	State	Assigned To	Effort
1	TBE - Cargar datos de estudiantes matriculados	Done	Alex Velastegui	
2	TBE - Copia malla curricular	Done	Christian Morán	13
3	TBE - Mensaje de créditos	Done	Cesar J. Santacruz	1
4	TBE - Error en correquisitos	Done		5
5	TBE - Previsualizar la encuesta	Done		8
6	TBE- Tipos de mensajes	Done	Cesar J. Santacruz	3
7	TBE - Cargar datos de currículums académicos	Done		13

Figura 13. Sprint Backlog 6

Es importante considerar que en la historia de usuario de carga de datos no se aprecia el esfuerzo, la cual fue estimada con 8 story points. La razón por la cual no se muestra el esfuerzo de la historia de usuario de carga de datos es debido a un error en el ingreso de los story points, ya que se ingresaron en un campo incorrecto.

Por otra parte, el error en correquisitos que se detectó fue estimado con cinco story points. Estos cinco story points se agregaron debido a que es una falla que se mantiene desde el anterior Sprint y, por lo tanto, debe ser resuelta por el equipo. Al no ser resuelta en el anterior Sprint y al ser parte clave para la configuración de mallas curriculares el equipo no podía retrasarlo para otro Sprint. En definitiva, el total de esfuerzo realizado por el equipo en el sexto Sprint es de 46 story points.

En las Daily Scrum el tiempo se gestionó de tal manera que se respondan a las tres preguntas y si existe un bloqueo se debía mencionar a la persona que puede solventar el bloqueo. Luego de contestar las tres preguntas, se resolvían los bloqueos entre las personas involucradas.

En cuanto al respeto mutuo del tiempo del equipo, cada integrante debía informar diez minutos antes si se conectará a la reunión diaria para minimizar los tiempos de espera. Esta medida fortaleció la confianza y el compromiso del equipo, generando fluidez en las Daily Scrum.

La transparencia de la información sobre problemas de entendimiento del negocio y de aprendizaje de tecnología fue notoria. Por un lado, los problemas de entendimiento de las funcionalidades que se debían implementar se solventaron por medio de reuniones cortas en Microsoft Teams. Por otra parte, la Wiki ayudó a centralizar la información de tecnología y cualquier tipo de información relacionada con el desarrollo del sistema.

Las pruebas funcionales se realizaron al menos cada dos días con datos de pruebas, para detectar posibles fallas en el sistema. Además, el equipo expresó al Scrum Master que agregar ese tipo de pruebas hubiese requerido la incorporación de un miembro al equipo que se encargue del aseguramiento de calidad de software.

El error pendiente de solucionar respecto a la configuración de correquisitos se solventó por medio de un análisis de código, recreando la situación que generaba el error. Es importante destacar que la comunicación entre los Scrum Developers fue importante para dar solución a este error.

Para la implementación de la historia de usuario correspondiente a la carga de datos fue necesario que el equipo comunique las plantillas necesarias que el sistema requiere. Este

proceso se lo realizó analizando la estructura requerida de la información y conversando sobre los datos disponibles con el Scrum Master y el Product Owner. Fue necesario realizar reuniones por medio de Microsoft Teams para abarcar la comunicación directa en horarios realmente extendidos. Todas estas reuniones realizadas tenían la intención de cumplir con los criterios de aceptación de cada historia de usuario del Sprint Backlog.

La definición de terminado se puede apreciar en la Tabla 9. Para este Sprint se consideraron la información de los bugs, se enfatizó en el código de las funciones implementadas y la documentación actualizada en la Wiki. Adicionalmente, se mejoró la redacción de cada elemento de la definición de terminado para que se pueda interpretar como características que debe cumplir el sistema.

Tabla 9. Definición de Terminado de Sprint 6

Definición de Terminado	
Desarrollo	Código formateado con ESLINT
	Los nombres de variables y métodos siguen la convención camel case
	Código de funciones completo y funcional
	Código fuente revisado, comentado y en idioma inglés
	Información sobre bugs en desarrollo completa y actualizada
Pruebas	Pruebas unitarias escritas y todas pasan satisfactoriamente
	Pruebas end to end escritas y todas pasan satisfactoriamente
Git	Rama de desarrollo actualizada
	Rama de producción actualizada
Pipelines	Pipelines de instalación de dependencias y pruebas funcionando
	Pipelines de Build y Release funcionando
Documentación	Documentación API actualizada
	Wiki de Azure Actualizada
Producción	Deploy satisfactorio
	Criterios de aceptación

La información de los bugs en la definición de terminado tenía como objetivo que el equipo verifique que se haya corregido antes de la revisión del Sprint. Adicionalmente, era importante que el equipo comunique cada vez que se encuentren o se resuelvan los bugs encontrados.

El equipo determinó que la implementación de funciones en el código era importante porque esto reducía la probabilidad de encontrar bugs. Una vez que se haya implementado una funcionalidad, como la de cargar estudiantes, uno de los integrantes debía revisar que

el código sea funcional. En caso de encontrar que el código no funcionaba o no cumplía con la necesidad del negocio, se agendaba una reunión por Teams para solventar el código. Cuando el cambio necesario para que el código sea funcional era mínimo, el equipo se comunicaba por medio de un canal de WhatsApp.

Un punto importante es que WhatsApp realizó una actualización que permitía dividir a un grupo en varias comunidades. El equipo consideró que se podía implementar una estrategia de comunicación creando una comunidad dentro del grupo solamente para atender bugs del sistema. Esto con el fin de que todos los bugs se solventen al instante que el equipo lea el mensaje en la comunidad dedicada a los errores.

Respecto a la definición de terminado, la actualización del Wiki era un factor clave para que el equipo abarque la transparencia de Scrum. Por tal motivo, el equipo acordó que toda la información relacionada con el proyecto debía ser cargado en la Wiki. Específicamente se colocó información de mallas curriculares, objetivos de Sprint, definición de terminado, datos de pruebas, enlaces de acceso al sistema, pasos para ejecutar pipelines de integración continua, manual de usuario, plantillas para carga de datos de estudiantes y asignaturas y recursos de aprendizaje.

Para la Sprint Review, todos los Scrum Developers habían realizado varias pruebas del sistema en producción y habían generado datos de pruebas para la presentación del incremento. Finalmente, el Product Owner realizó algunas observaciones en cuanto al tiempo de carga de las funcionalidades de copiar malla y cargar datos de mallas curriculares o estudiantes. Sin embargo, los Scrum Developers expresaron que el tiempo de carga se debe a los recursos del sistema, los cuales podrían incrementarse con un mayor presupuesto. Además, el equipo de desarrollo explicó que el rendimiento del sistema incrementaría cuando se escale los recursos disponibles. La funcionalidad de carga de curriculums se lo puede apreciar en Figura 14.

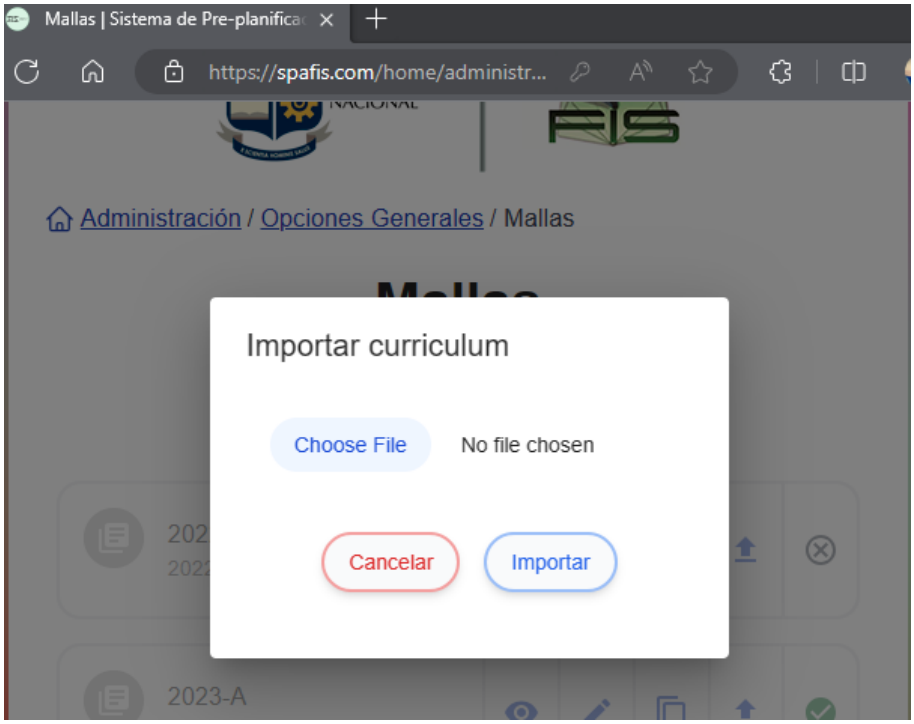


Figura 14. Incremento Funcional de Sprint 6

La Sprint Retrospective se realizó con Miro utilizando el mismo pizarrón virtual del anterior Sprint. Para ello se agregaron nuevos espacios donde se debían colocar cada nota adhesiva para abarcar la inspección. Las notas adhesivas colocados en el anterior Sprint no se borraron para poder observar si el equipo se adaptó a las propuestas de mejoras.

Agregar pruebas de carga y estrés y un manual de usuario de Azure fueron las cosas que el equipo consideró que se deben empezar a hacer. El objetivo de las pruebas de carga y estrés permitirían al equipo obtener conocimiento de la concurrencia de usuarios que aceptará el sistema y la capacidad de aumentar el límite de concurrencia. Por otro lado, el manual de usuario de Azure permitiría a las autoridades controlar los permisos en cuanto a la infraestructura donde se ejecuta el sistema.

La razón principal de las pruebas de cargas y estrés es la cantidad de estudiantes presentes en la Facultad de Ingeniería de Sistemas durante el período que se implementó el sistema. Al ser aproximadamente 1000 estudiantes matriculados, el sistema debía soportar varios cientos de estudiantes realizando peticiones simultáneamente. Sin embargo, debido a las limitaciones de presupuesto de la licencia académica de Azure, no se podía incrementar los recursos de la infraestructura.

Los cambios en el diseño y en el sistema, la impuntualidad en la reunión diaria y la complejidad de los cambios fueron las cosas que el equipo sugirió parar de hacer. En cuanto a los cambios en el diseño y en el sistema, el problema se presentaba por falta de

la presentación de un diseño guía. Por otra parte, la impuntualidad de las reuniones es un problema que se presentó por limitaciones en la disponibilidad de tiempo de cada integrante. De la misma manera, la subestimación de complejidad de los cambios propuestos surgió por falta de experiencia, ya que no se consideraban aspectos como el impacto del cambio en el resto de las funcionalidades ya implementadas.

El equipo determinó en la Sprint Retrospective que se debía comenzar a hacer prototipos, las pruebas de carga y estrés y pruebas de usabilidad. Los prototipos se enfocaban en solucionar los problemas de cambios no planificados de diseño. De igual manera, las pruebas de carga y estrés tenían la intención de obtener información de la concurrencia del sistema. Así mismo, las pruebas de usabilidad determinarían información sobre la facilidad de uso del sistema para usuarios reales.

El equipo determinó que se debe “Hacer más” es adquirir una velocidad constante de desarrollo. Esto es fundamental para que el Product Owner determine la cantidad de esfuerzo que los Scrum Developers son capaces de realizar en cada Sprint y de esta manera se pueda determinar una fecha de entrega del sistema.

Sprint Siete: Generar Reportes

El séptimo Sprint tenía como objetivo la generación de reportes personalizados. En base al objetivo se realizó la planificación del Sprint. Estos reportes buscaban conocer de manera resumida la demanda de materias de estudiantes y los datos cargados al sistema.

El equipo seleccionó las historias de usuario relacionadas con el objetivo, obteniendo así el Sprint Backlog (Ver Figura 15). Adicionalmente, se agregó un método de autenticación para el consumo de datos, lo cual fue una característica importante que al principio no se consideró. Es importante tener en cuenta que el Product Owner atendía las inquietudes de las historias de usuario para que los Scrum Developers tengan claridad de los requerimientos.

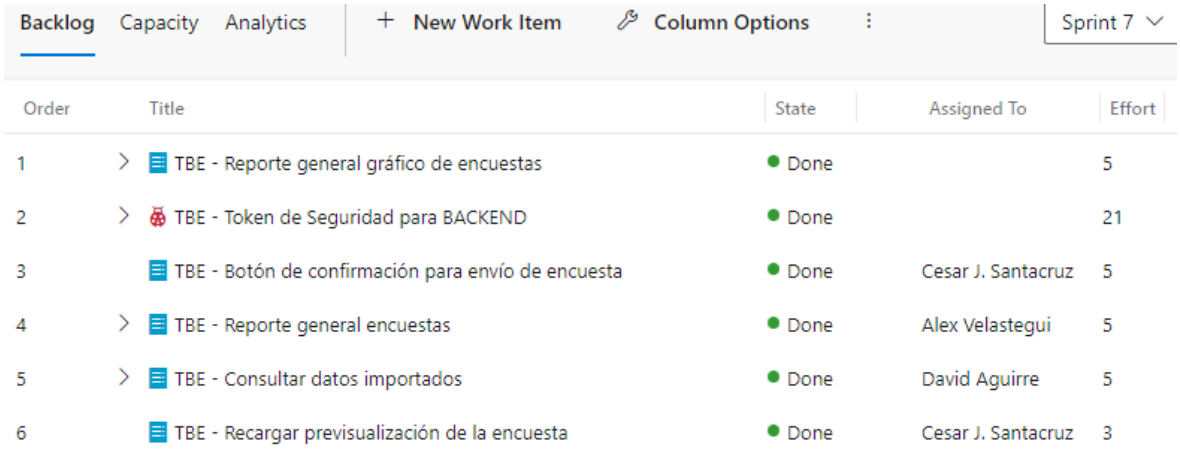


Figura 15. Sprint Backlog 7

La estimación de las historias de usuario del Sprint acumuló un esfuerzo de 44 story points. El defecto relacionado con la seguridad del backend es el que ocupó 21 story points, lo cual requería un gran esfuerzo técnico para la implementación.

Mejorar el mensaje de confirmación de la encuesta fue la única historia de usuario que surgió a partir del anterior Sprint. Este mensaje es aquel se debía mostrar para aceptar las condiciones del envío de la encuesta de parte de los estudiantes. Esta característica fue de alto valor para el usuario porque permitía asegurarse que los estudiantes leyeron y entendieron las consecuencias del envío de la encuesta.

Continuando con la implementación del Sprint, la asistencia a las Daily Scrum fue constante y a tiempo. A diferencia de los anteriores Sprints, en este Sprint el equipo mantuvo el compromiso de llegar a tiempo o avisar con anticipación si no se podían conectar.

La necesidad de las Daily Scrum para inspeccionar los avances de los reportes y de la seguridad fue fundamental. En una de las Daily Scrum se logró solventar una falla del sistema debido a la incorporación de la seguridad en el consumo de información. Por tal motivo, las Daily Scrum permitieron al equipo inspeccionar los avances del sistema.

La honestidad y el compromiso fueron los valores que resaltaron en el equipo durante el séptimo Sprint. En cada reunión diaria, los miembros del equipo decían sin temor a ser juzgados los avances realizados. Por ejemplo, si uno de los integrantes no había realizado ningún avance lo decía sin temor. Este hecho refuerza el compromiso que el equipo tiene para alcanzar el objetivo del Sprint. Al inspeccionar los avances del resto del equipo, el integrante que no ha podido avanzar se sentía motivado a realizar las tareas asignadas.

El coraje de poder rehacer el código que se había implementado para incorporar el nivel de seguridad en el consumo de servicios fue otro factor que influyó en el alcance del Sprint.

A pesar de que se tuvieron que cambiar al menos 100 líneas de código, el equipo estaba comprometido con alcanzar el objetivo.

Por otro lado, uno de los principales bloqueos fue la elección de tecnología para realizar los reportes. El Scrum Master participó activamente en reuniones para unir al equipo y facilitar la que el equipo de desarrollo provea ideas de soluciones tecnológicas para realizar reportes. Para solucionar este problema fue necesario una intermediación entre el Product Owner y los Scrum Developers. Por una parte, el Product Owner expresaba su necesidad con respecto a los reportes de encuestas, mallas curriculares y demanda de asignaturas. Mientras que los Developers exponían sus dudas respecto a cada reporte. En este punto fue indispensable el principio de "Mantenerlo simple" para cumplir con el objetivo del Sprint y satisfacer las necesidades del cliente.

La definición de terminado no evolucionó en este Sprint. La definición de terminado se puede apreciar en Tabla 10.

Tabla 10. Definición de Terminado de Sprint 7

Definición de Terminado	
Desarrollo	Código formateado con ESLINT
	Los nombres de variables y métodos siguen la convención camel case
	Código de funciones completo y funcional
	Código fuente revisado, comentado y en idioma inglés
	Información sobre bugs en desarrollo completa y actualizada
Pruebas	Pruebas unitarias escritas y todas pasan satisfactoriamente
	Pruebas end to end escritas y todas pasan satisfactoriamente
Git	Rama de desarrollo actualizada
	Rama de producción actualizada
Pipelines	Pipelines de instalación de dependencias y pruebas funcionando
	Pipelines de Build y Release funcionando
Documentación	Documentación API actualizada
	Wiki de Azure Actualizada
Producción	Deploy satisfactorio
	Criterios de aceptación

En la Sprint Review, los Scrum Developers demostraron el funcionamiento del módulo de reportes y el cumplimiento de los criterios de aceptación al Product Owner. Para ello fue necesario cargar varios datos de prueba y actualizarlos en la Wiki. Una vez que finalizó la presentación, el Product Owner aceptó el incremento del producto software. Sin embargo,

el Producto Owner agregó que el sistema debía mostrar retroalimentación a la hora de realizar procesos que duren más de 2 segundos, como la carga de curriculums. El incremento funcional se puede apreciar en Figura 16.

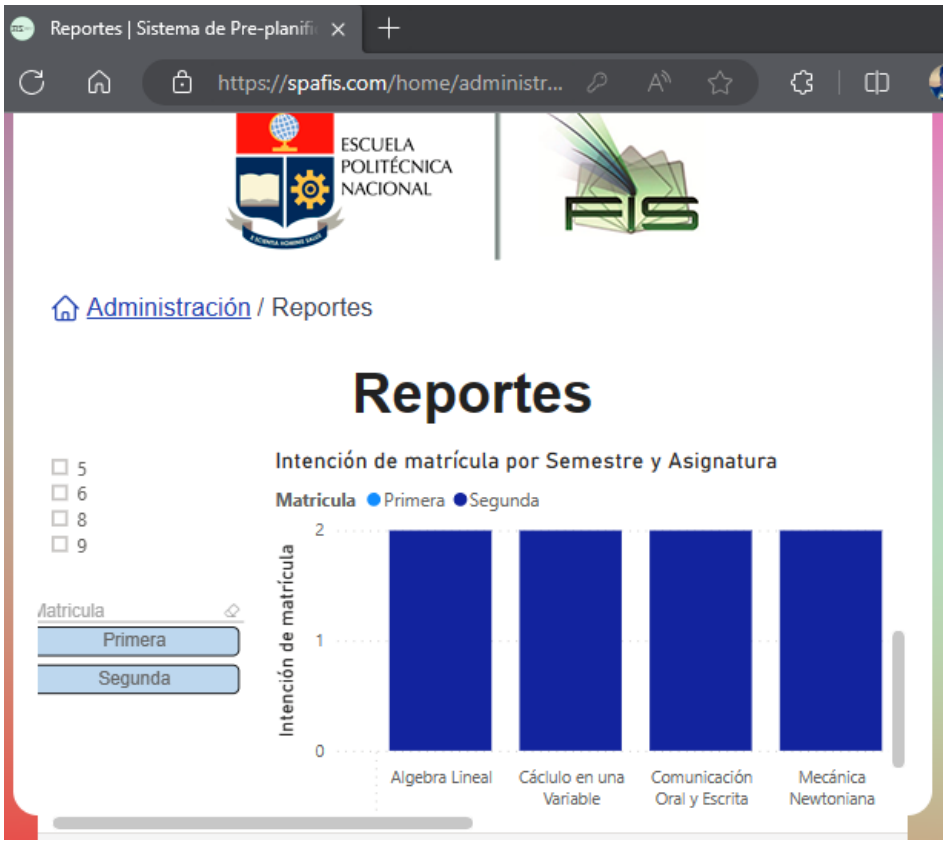


Figura 16. Incremento Funcional de Sprint 7

La Sprint Retrospective dejó como compromiso a los Scrum Developers que deben concentrarse en la actualización frecuente de la definición de terminado. Este compromiso tenía el objetivo de que la definición de terminado evolucione conforme a los cambios en las funcionalidades del sistema.

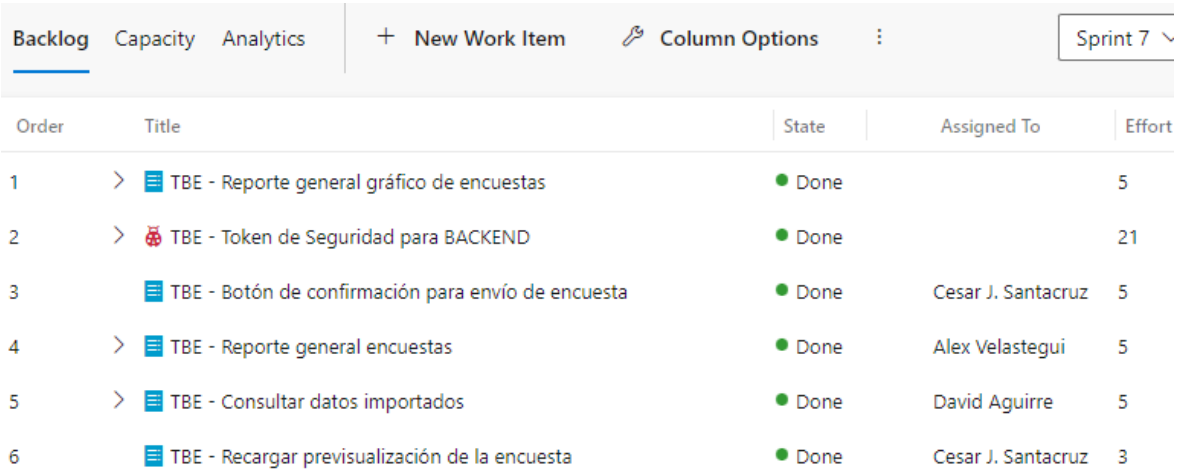
3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 Resultados

La plataforma Azure DevOps permitió al equipo tener la información accesible en cualquier momento. De esta forma, cada integrante tenía la capacidad de actualizar y revisar artefactos importantes, tales como: el Product Backlog, Sprint Backlog, Wiki, pipelines de

integración continua y repositorios. Como resultado, el equipo alcanzó progresivamente la inspección, adaptación y transparencia en el desarrollo del producto software.

Durante cada Sprint el equipo planificó su trabajo en base a un objetivo acordado entre el Product Owner y los Scrum Developers. Azure DevOps ayudó al equipo en la gestión del Product Backlog (ver la Figura 17) para crear, actualizar, eliminar, estimar y priorizar las historias de usuario, así como para obtener el Sprint Backlog para la ejecución del Sprint.



Order	Title	State	Assigned To	Effort
1	TBE - Reporte general gráfico de encuestas	Done		5
2	TBE - Token de Seguridad para BACKEND	Done		21
3	TBE - Botón de confirmación para envío de encuesta	Done	Cesar J. Santacruz	5
4	TBE - Reporte general encuestas	Done	Alex Velastegui	5
5	TBE - Consultar datos importados	Done	David Aguirre	5
6	TBE - Recargar previsualización de la encuesta	Done	Cesar J. Santacruz	3

Figura 17. Sprint Backlog

La herramienta Microsoft Teams permitió administrar y grabar las Daily Scrum. Las grabaciones están disponibles en una carpeta en la nube (ver Figura 18) lo que permitió transparencia en estos eventos.

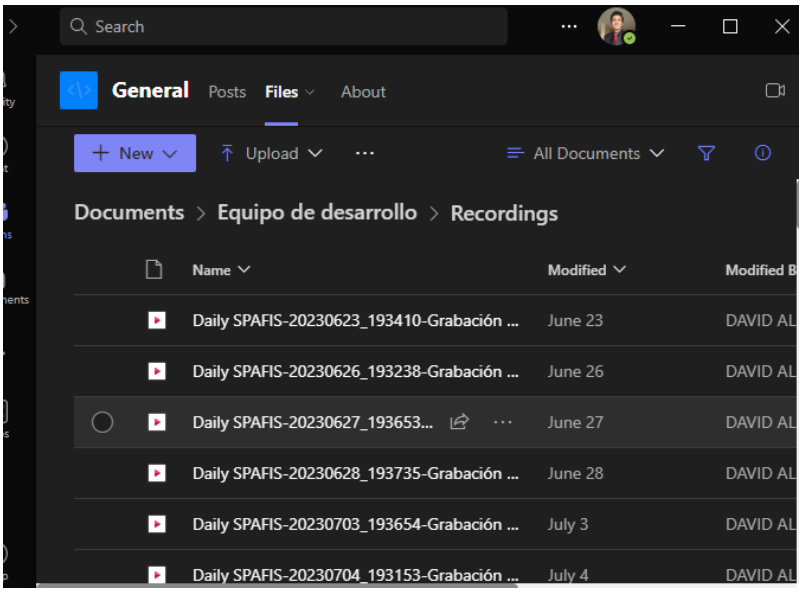


Figura 18. Carpeta con grabaciones de las Daily Scrum

La aplicación web Planning Poker facilitó la estimación de las historias de usuario por medio de la creación de un juego colaborativo. En la Figura 19 se muestra uno de los juegos del Planning Poker realizado con los Scrum Developers.

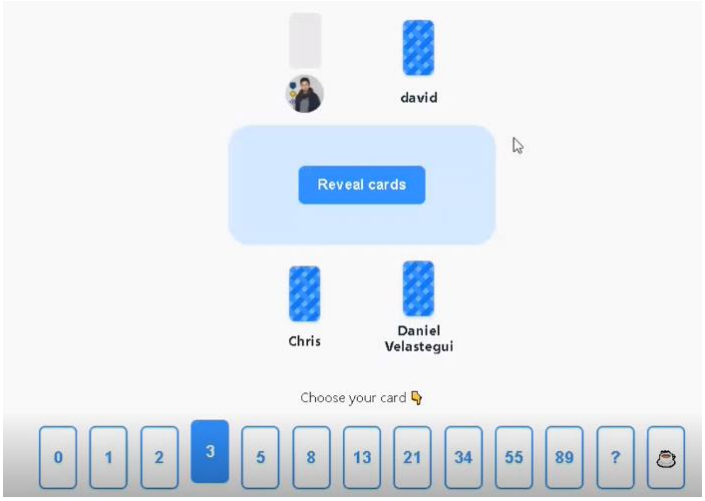


Figura 19. Planning Poker Online

Azure DevOps también facilitó la transparencia de la información por medio de la creación de Wikis. En las Wikis se colocó información importante para el desarrollo del producto software, tales como: los objetivos de los Sprints, los datos de prueba, el manual de usuario, entre otros (ver Figura 20).

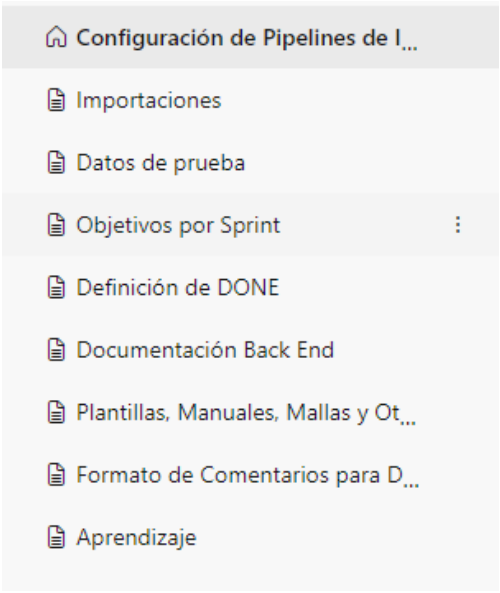


Figura 20. Wiki con Información del Sistema

La integración de Azure DevOps con los repositorios de control de versiones de Azure permitió llevar trazabilidad de cada funcionalidad implementada. Esto ayudó a que los Scrum Developers puedan implementar código a partir de las tareas técnicas del Sprint

Backlog. La Figura 21 es un ejemplo de integración entre los tableros de Azure con el repositorio.

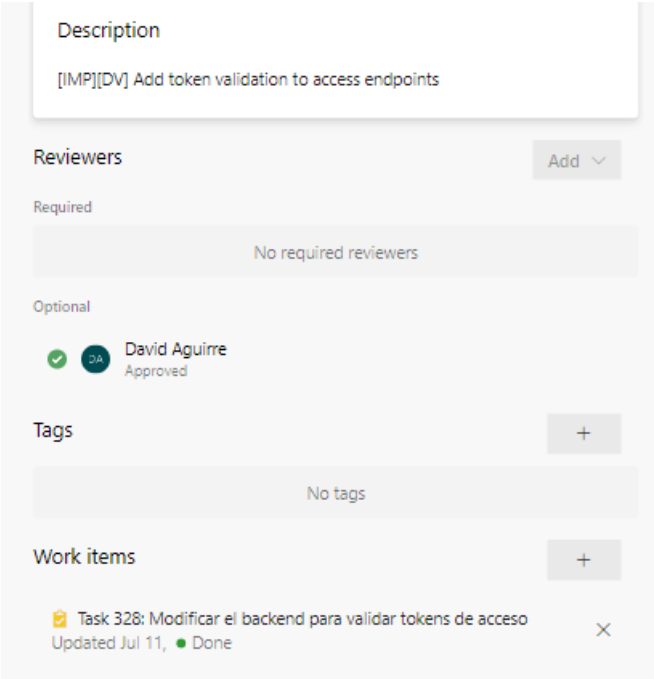


Figura 21. Pull Request mapeada con una tarea técnica

Azure DevOps ayudó a los Scrum Developers en el proceso de integración continua por medio de los pipelines de pruebas y despliegue. Cada vez que se realizaba un cambio en desarrollo se ejecutaban automáticamente los pipelines para el despliegue en producción. La Figura 22 muestra un resumen del pipeline de integración continua.

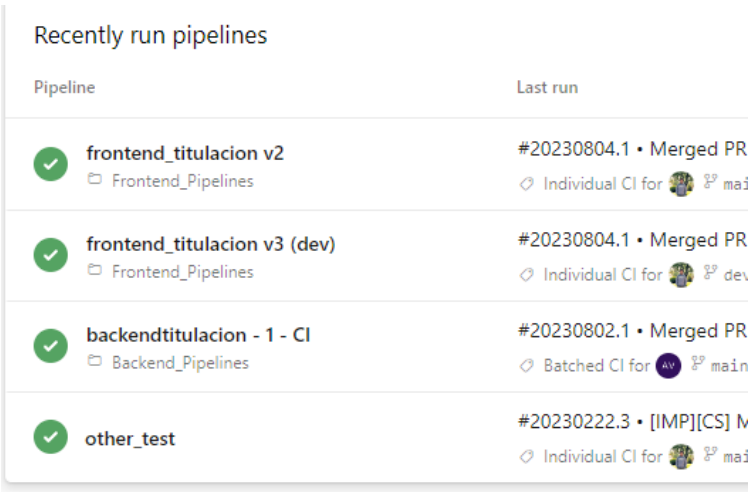


Figura 22. Pipelines de Integración Continua

La aplicación web Miro ayudó al equipo a plasmar sus fortalezas, debilidades y mejoras de proceso de manera colaborativa en tableros virtuales. Esta herramienta facilitó la

implementación de las Sprint Retrospectives, así como ayudó a la resolución de dudas por medio de diagramas gráficos. En la Figura 23 se muestra el tablero utilizado para la Sprint Retrospective.

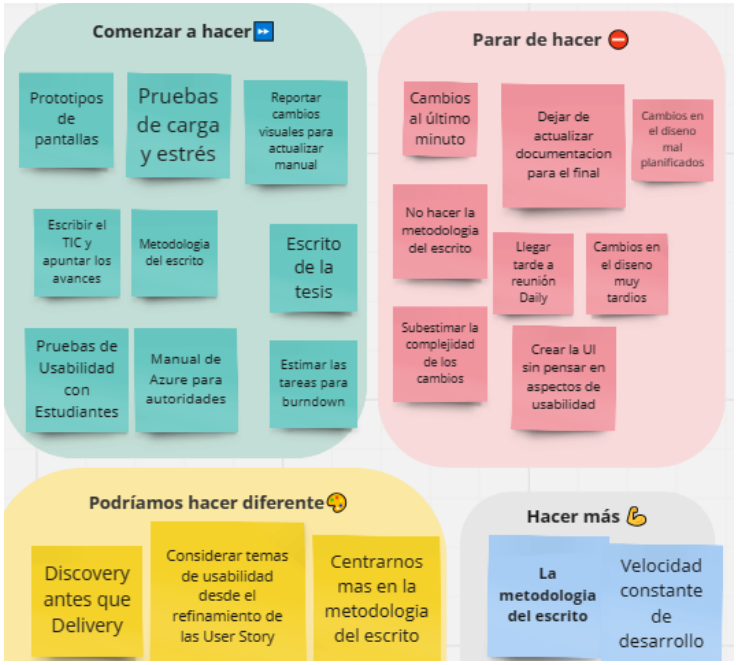


Figura 23. Tablero Colaborativo de Sprint Retrospective

El equipo implementó un total de 241 story points durante los 7 Sprints. La velocidad promedio de desarrollo del equipo fue de 34 story points en cada Sprint. Es importante mencionar que en cada Sprint influyeron varios factores, tales como: la disponibilidad de tiempo de cada integrante, días de feriado, emergencias médicas, entre otros. En Tabla 11 se presenta la cantidad de story points por cada Sprint. Los Scrum Developers intentaron mantener una velocidad constante en el desarrollo del producto software, lo cual se complicó debido a actividades académicas y laborales.

Tabla 11. Total de Story Points por Sprint

Sprint	Story Points
1	48
2	25
3	21
4	36
5	24
6	43
7	44

De igual manera, la Figura 24 muestra las variaciones de la velocidad del equipo en cada Sprint.

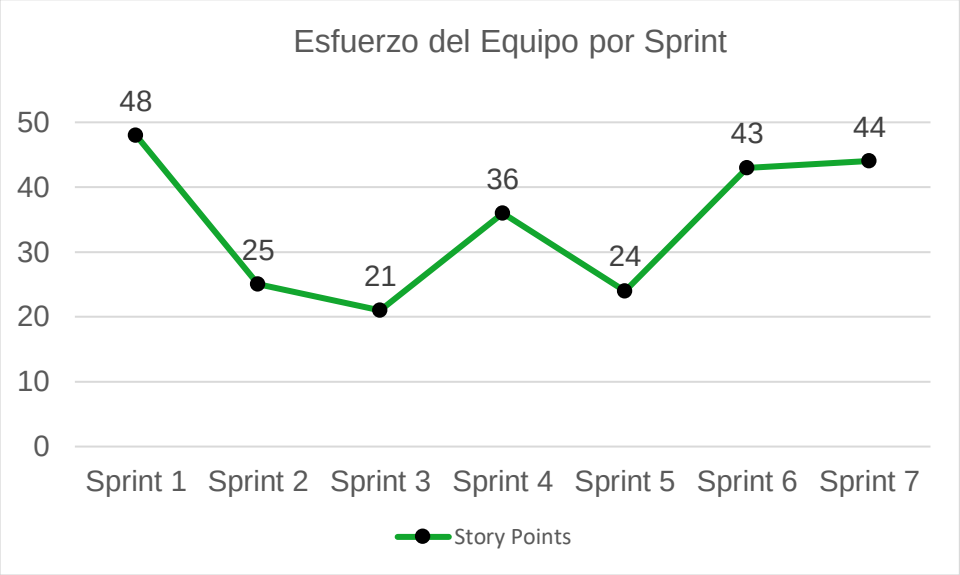


Figura 24. Resultado de Esfuerzo por Sprint

Finalmente, la versión de la aplicación web desarrollada al finalizar el último Sprint se puede apreciar mediante un video en el Anexo I.

3.2 Conclusiones

El estudio de herramientas utilizando criterios relevantes para el equipo permite seleccionar las herramientas que apoyan al marco Scrum. En este caso específico, el criterio determinante fue la experiencia del equipo con la herramienta.

Cuando el Scrum Team no tiene claridad de los cinco valores de Scrum, las herramientas utilizadas para la gestión de eventos y artefactos no son relevantes. Por tal motivo, cada integrante del equipo debe esforzarse para adquirir el compromiso, la honestidad, el enfoque, la apertura y el coraje de Scrum.

Azure DevOps ayuda a la implementación efectiva de la transparencia, inspección y adaptación de Scrum. Los Scrum Developers tienen a su disposición los tableros de Azure DevOps y la Wiki para colocar información relevante en el desarrollo del sistema. De esta forma, el equipo puede realizar inspecciones y ajustes en tiempo real a medida que avanza el desarrollo del sistema.

Los tableros para la gestión de historias de usuario de Azure DevOps facilitan al Scrum Team la gestión del Product Backlog y el Sprint Backlog. Con esta herramienta, el Scrum Team controla los flujos de trabajo de tareas técnicas para la construcción del incremento funcional. De manera similar, los repositorios de Azure permiten a los Scrum Developers llevar trazabilidad de las tareas implementadas y los avances diarios.

El uso de la aplicación web Planning Poker para las estimaciones de las historias de usuario ayuda a realizar esta actividad de manera colaborativa, sin importar la ubicación de los integrantes del Scrum Team. La capacidad de realizar la estimación remotamente permite a los Scrum Developers participar de la estimación desde su hogar u oficina.

La aplicación Miro facilita las Sprint Retrospectives por medio de la colaboración remota del equipo. La experiencia previa en el uso de esta herramienta ayuda a que los equipos tengan confianza en expresar sus ideas de mejora sin el temor a ser observados.

3.3 Recomendaciones

El equipo llevó a la práctica los tres pilares de Scrum en el desarrollo de software: inspección constante, adaptación continua y total transparencia. La aplicación de estos tres pilares de Scrum requiere de retroalimentación constante, motivación y esfuerzo.

Por otro lado, en los primeros Sprints, la comunicación se presentó como una de las debilidades fundamentales del equipo. No obstante, la constante motivación para aplicar los valores de Scrum, así como las reuniones en Microsoft Teams, lograron mejorar la confianza y mantener una comunicación efectiva entre el equipo.

El uso de la Wiki de Azure DevOps para mantener la transparencia fue una de las lecciones que el equipo aprendió durante el desarrollo del Sistema. Una vez que el equipo es transparente con la información, se puede implementar la inspección y adaptación.

Los tableros de Azure DevOps para la gestión del Product Backog y el Sprint Backlog ayudó a la implementación del Sprint Planning. Por medio del Sprint Backlog de Azure DevOps, el equipo aprendió a inspeccionar los avances de las tareas técnicas.

Durante el desarrollo del sistema, el equipo aprendió diversos aspectos enfocados al desarrollo de software con Scrum, tales como: la evolución de la definición de "terminado", la gestión del riesgo asociado a cambios de último momento y la relevancia de la honestidad en la Daily Scrum. Estas lecciones enriquecedoras se fueron integrando a medida que avanzaba el desarrollo del sistema.

La evolución de la definición de terminado fue notoria en cada Sprint. Al comparar la definición de terminado del primer Sprint con la del séptimo Sprint se puede apreciar la diferencia de la claridad de escritura y de sección de sus elementos.

Por otra parte, la sinceridad de la comunicación de los obstáculos juega un rol fundamental en la Daily Scrum. Cuando los Scrum Developers no expresan los obstáculos que impiden que avancen con sus tareas, se pierde la visión general del desarrollo del sistema. La omisión de los obstáculos en la Daily Scrum ocasiona que Microsoft Teams parezca inefectivo y limita la función del Scrum Master.

Durante los primeros Sprints, los Scrum Developers solamente exponían sus avances de manera superficial, lo cual no permitía una correcta inspección de las tareas implementadas. Por lo tanto, los Developers decidieron expresar sus avances mencionando el identificador de la tarea en Azure DevOps. Esto se realizó para que el equipo conozca las tareas que ya fueron implementadas y las que están pendientes.

Otra lección aprendida fue la importancia de llegar a la Sprint Review con el sistema funcional. Por tal motivo, el equipo aprendió que es indispensable la integración de un analista de control de calidad. De esta manera, el producto podía someterse a pruebas constantes que verifiquen continuamente el correcto funcionamiento del sistema.

4 REFERENCIAS BIBLIOGRÁFICAS

Berczuk, S. (2007). Back to basics: The role of agile principles in success with an distributed scrum team. *Proceedings - AGILE 2007*. https://doi.org/10.1109/AGILE.2007.17

Cohn, M. (2004). User Stories Applied: For Agile Software Development (Addison Wesley Signature Series). En *Writing* (Vol. 1, Número 0).

De Melo, C. O., S. Cruzes, D., Kon, F., & Conradi, R. (2013). Interpretative case studies on agile team productivity and management. *Information and Software Technology*, *55*(2). https://doi.org/10.1016/j.infsof.2012.09.004

Fowler, M., & Highsmith, J. (2001). The Agile Manifesto. *SpringerBriefs in Computer Science*, *9*(8), 28–35.

Heck, P., & Zaidman, A. (2018). A systematic literature review on quality criteria for agile requirements specifications. En *Software Quality Journal* (Vol. 26, Número 1, pp. 127–160). Springer New York LLC. https://doi.org/10.1007/s11219-016-9336-4

Jeffries, R. (2001). Essential XP: Card, conversation, confirmation. *Ronjeffries.Com*.

Ken Schwaber, & Jeff Sutherland. (2020). *The Definitive Guide to Scrum: The Rules of the Game*.

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, & Dave Thomas. (2001). *Manifesto for Agile Software Development*. https://agilemanifesto.org/

Koelsch, G. (2016). Requirements Writing for System Engineering. En *Requirements Writing for System Engineering*. https://doi.org/10.1007/978-1-4842-2099-3

Kropp, M., Meier, A., Mateescu, M., & Zahn, C. (2014). Teaching and learning agile collaboration. *2014 IEEE 27th Conference on Software Engineering Education and Training, CSEE and T 2014 - Proceedings*. https://doi.org/10.1109/CSEET.2014.6816791

Kudikyala, U. K., & Dulhare, U. N. (2016). Using Scrum and Wikis to manage student major projects. *Proceedings of the 2015 IEEE 3rd International Conference on MOOCs, Innovation and Technology in Education, MITE 2015*. https://doi.org/10.1109/MITE.2015.7375279

Liskin, O., Pham, R., Kiesling, S., & Schneider, K. (2014). Why we need a granularity concept for user stories. *Lecture Notes in Business Information Processing*, *179 LNBIP*. https://doi.org/10.1007/978-3-319-06862-6_8

McGreal, D., & Jocham, R. (2018). *The professional product owner: Leveraging scrum as a competitive advantage*. Addison-Wesley Professional.

Noll, J., Razzak, M. A., Bass, J. M., & Beecham, S. (2017). A study of the scrum master's role. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10611 LNCS*. https://doi.org/10.1007/978-3-319-69926-4_22

Raharjana, I. K., Siahaan, D., & Fatichah, C. (2021). User Stories and Natural Language Processing: A Systematic Literature Review. *IEEE Access*, *9*, 53811–53826. https://doi.org/10.1109/ACCESS.2021.3070606

Ramos, A. B., & Vilela Junior, D. C. (2017). Scrum Master's Role Influence On Scrum Projects Development. *Revista de Gestão e Projetos*, *08*(03).

Rubin, K. S. (2012). Essential Scrum. En *Essential Scrum: A Practical Guide to the Most Popular Agile Process*.

Santos, W. (2016). Towards a better understanding of simplicity in agile software development projects. *ACM International Conference Proceeding Series*, *01-03-June-2016*. https://doi.org/10.1145/2915970.2915978

Sharp, H., Baddoo, N., Beecham, S., Hall, T., & Robinson, H. (2009). Models of motivation in software engineering. *Information and Software Technology*, *51*(1). https://doi.org/10.1016/j.infsof.2008.05.009

Sunner, D. (2017). Agile: Adapting to need of the hour: Understanding Agile methodology and Agile techniques. *Proceedings of the 2016 2nd International Conference on Applied and Theoretical Computing and Communication Technology, iCATccT 2016*. https://doi.org/10.1109/ICATCCT.2016.7911978

Sverrisdottir, H. S., Ingason, H. T., & Jonasson, H. I. (2014). The Role of the Product Owner in Scrum-comparison between Theory and Practices. *Procedia - Social and Behavioral Sciences*, *119*. https://doi.org/10.1016/j.sbspro.2014.03.030

Takeuchi, H., & Nonaka, I. (1986). The New New Product Development Game Harvard Business Review. *Harvard Business Review*.

Wautelet, Y., Heng, S., Kolp, M., & Mirbel, I. (2014). Unifying and extending user story models. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8484 LNCS*. https://doi.org/10.1007/978-3-319-07881-6_15

Welin Matti. (2013). USER STORIES IN SOFTWARE DEVELOPMENT. *Cleantech insinöörikoulutuksessa*, 108–117.

Williams, L. (2012). What agile teams think of agile principles. *Communications of the ACM*, *55*(4). https://doi.org/10.1145/2133806.2133823

5 ANEXOS

ANEXO I

Este anexo contiene el enlace a un vídeo donde se muestra el sistema implementado:

https://www.youtube.com/watch?v=0rKc9pdLqAY