Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

USO DE LA METODOLOGÍA SCRUM EN EL DESARROLLO DE APLICACIONES WEB

Ligia María Gaitán González
Asesorada por el Ing. Daniel Caciá Rivas

Guatemala, junio de 2017

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

USO DE LA METODOLOGÍA SCRUM EN EL DESARROLLO DE APLICACIONES WEB

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

LIGIA MARÍA GAITÁN GONZÁLEZ
ASESORADA POR EL ING. DANIEL CACIÁ RIVAS

AL CONFERÍRSELE EL TÍTULO DE

INGENIERA EN CIENCIAS Y SITEMAS

GUATEMALA, JUNIO DE 2017

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Pedro Antonio Aguilar Polanco
VOCAL I	Ing. Angel Roberto Sic García
VOCAL II	Ing. Pablo Christian de León Rodríguez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Jurgen Andoni Ramírez Ramírez
VOCAL V	Br. Oscar Humberto Galicia Nuñez
SECRETARIA	Inga. Lesbia Magalí Herrera López

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADORA	Inga. Virginia Victoria Tala Ayerdi
EXAMINADOR	Ing. Freiry Javier Gramajo López
EXAMINADOR	Ing. César Augusto Fernández Cáceres
SECRETARIA	Inga. Marcia Ivonne Véliz Vargas

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

USO DE LA METODOLOGÍA SCRUM EN EL DESARROLLO DE APLICACIONES WEB

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha 18 de enero de 2017.

Ligia María Gaitán González

Guatemala, 27 de abril de 2017

Señores
Comisión de Revisión de Trabajos de Tesis
Escuela de Ciencias y Sistemas
Facultad de Ingeniería
Universidad de San Carlos de Guatemala

Respetables Señores:

De manera atenta me dirijo a ustedes para hacer de su conocimiento que la estudiante Ligia María Gaitán González con número de carné 1998-11892 ha finalizado su Trabajo de Graduación, titulado "Uso de la metodología SCRUM en el desarrollo de aplicaciones web".

A dicho trabajo se le efectuaron las revisiones necesarias y se realizaron los ajustes pertinentes para que pueda continuar con el proceso de revisión. Me es grato suscribirme de ustedes.

Atentamente,

Daniel Caciá Rivas
Ingeniero en Ciencias y Sistemas
Colegiado No. 8882

Ing. Daniel Cacia Rivas
Colegiado 8,882

Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 10 de Mayo de 2017

Ingeniero
Marlon Antonio Pérez Türk
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Pérez:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación de la estudiante **LIGIA MARÍA GAITÁN GONZÁLEZ** con carné **199811892 y CUI 2604 34736 0608,** titulado **"USO DE LA METODOLOGIA SCRUM EN EL DESARROLLO DE APLICACIONES WEB",** y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,

Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación

ESCUELA DE INGENIERÍA EN CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÌA
ESCUELA DE INGENIERÍA EN
CIENCIAS Y SISTEMAS
TEL: 24767644

El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, del trabajo de graduación ***"USO DE LA METODOLOGÍA SCRUM EN EL DESARROLLO DE APLICACIONES WEB",*** *realizado por la estudiante LIGIA MARÍA GAITÁN GONZÁLEZ aprueba el presente trabajo y solicita la autorización del mismo.*

"ID Y ENSEÑAD A TODOS"

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION DE INGENIERIA EN CIENCIAS Y SISTEMAS

Ing. Marlon Antonio Pérez Türk
Director
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 31 de mayo de 2017

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

DTG. 262.2017

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al Trabajo de Graduación titulado: **USO DE LA METODOLOGÍA SCRUM EN EL DESARROLLO DE APLICACIONES WEB,** presentado por la estudiante universitaria: **Ligia María Gaitán González,** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, autoriza la impresión del mismo.

IMPRÍMASE:

Ing. Pedro Antonio Aguilar Polanco
Decano

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERÍA

Guatemala, junio de 2017

/gdech

ACTO QUE DEDICO A:

Mis padres	Mario Alberto Gaitán y Ligia Margarita González, por su inmenso amor y apoyo incondicional. Que esta meta alcanzada sea muestra de mi agradecimiento a todos sus esfuerzos y su dedicación.
Mis hermanos	Ana Margarita, Martha Zuliana, Ligia Sussely y Mario Alberto Gaitán González, por su amor y apoyo en cada etapa de mi vida.
Mi esposo	Carlos Fernando, por su amor y apoyo siempre en todo. Sos mi complemento.
Mis hijos	Carlos Ernesto y María Fernanda, por ser el motor que me impulsa día a día, mi motivación para cerrar este ciclo.

AGRADECIMIENTOS A:

Dios	Que ha sido bueno conmigo.
Universidad de San Carlos de Guatemala	Por el honor de transitar sus pasillos persiguiendo mi formación profesional.
Mis compañeros	Por su apoyo y compañía durante los años de estudio. En especial a Tito Álvarez, Alfredo Valdés, William Santisteban y Viviana Vaides.
Mis catedráticos	En general.
Mi asesor	Daniel Caciá Rivas, por su apoyo en la elaboración de este trabajo.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
%	Porcentaje

GLOSARIO

Aplicación Es un programa de computadoras diseñado como una herramienta para ayudar a los usuarios a realizar determinadas tareas.

Aplicación de escritorio Es aquella que se encuentra instalada en el computador o sistema de almacenamiento y se ejecuta sin internet en el sistema operativo.

CMM (*Capability maturity model*) Modelo de madurez de capacidades, es un modelo de evaluación de procesos de desarrollo.

CSS (*Cascade style sheet*) Lenguaje de hojas de estilo, creadas para controlar la apariencia de una página o sitio web.

Dirección URL (*Uniform resource locator*) Localizador uniforme de recursos, es una secuencia de caracteres que sigue un estándar y que permite denominar recursos dentro del entorno de Internet para que puedan ser localizados.

Framework Marco de trabajo, es una estructura conceptual y tecnológica de asistencia definida que se utiliza como base para el desarrollo de software.

Google drive	Servicio ofrecido de forma gratuita por Google que brinda a los usuarios hasta 15 Gb de espacio de almacenamiento en la web.
HTML	**(***HyperText markup language*) Lenguaje de marcación de hipertexto, es un lenguaje que se utiliza para establecer la estructura y el contenido de un sitio web de texto, objetos e imágenes.
HTTP	(*HyperText transfer protocol*) Protocolo de transferencia de hipertexto, protocolo de comunicación que permite la transferencia de información en la WWW.
Java	Lenguaje de programación orientado a objetos.
Javascript	Lenguaje de programación interpretado utilizado comúnmente para agregar funcionalidad a los sitios web.
MySQL	Sistema de gestión de base de datos relacionales de código abierto.
Navegador web	Software que permite el acceso a la web, interpretando la información de distintos tipos de archivos para que puedan ser visualizados.
Oopsla	(*Object-oriented programing, systems, languages & applications*) Es una conferencia anual de investigadores.

Oracle	Herramienta cliente/servidor para la gestión de bases de datos.
PHP	Lenguaje de programación de código del lado del servidor diseñado para el desarrollo web de contenido dinámico.
PMI	(*Project management institute*) Es una de las principales asociaciones dedicada de la profesión de gestión de proyectos. Su producto más reconocido es el PMBOK (*project management book of knowledge*).
Red	Conjunto de equipos informáticos conectados entre sí.
Rubi	Lenguaje de programación interpretado, reflexivo y orientado a objetos.
SEO	(*Search engine optimization*) Conjunto de técnicas que se utilizan para optimizar un sitio web para que alcance el mejor posicionamiento posible en los buscadores de Internet.
Servidor	Tipo de *software* cuyo propósito es proveer de datos a los usuarios. El término también se utiliza para referirse al ordenador físico en el cual funciona dicho software.

Sitio web	Conjunto de archivos electrónicos referentes a un tema en particular y que se comparten a través de la red.
SQL	(*Structured query language*) Lenguaje de consulta estructurada, es un lenguaje específico del dominio que da acceso a un sistema de gestión de bases de datos relacionales que permite especificar diversos tipos de operaciones en ellos.
Vbscript	Lenguaje de programación interpretado utilizado comúnmente para agregar funcionalidad a los sitios web.
Web	Vocablo inglés que significa red o malla. En el ámbito de tecnología se utiliza para nombrar a una red informática, o en general a Internet.
WWW	(*World wide web*) El sistema de documentos de hipertexto que se encuentran enlazados entre sí y a los que se accede por medio de Internet.
XP	(*eXtreme programming*) Programación extrema, es una metodología ágil de desarrollo de *software* con énfasis en la adaptabilidad más que en la previsibilidad.

RESUMEN

Scrum es un método ágil para el desarrollo de proyectos, con una estrategia de desarrollo incremental cuyas fases no requieren un ciclo secuencial, sino pueden solaparse entre ellas. Tiene un proceso simple basado en ciclos de desarrollo con duración definida por el equipo, un listado de historias de usuario que definen los requerimientos del proyecto, las historias se ponderan con el tiempo que tomará su desarrollo y, según la prioridad establecida por el dueño del producto, se selecciona cuáles se desarrollarán en cada ciclo.

Durante el ciclo o *sprint*, se realizan reuniones diarias, las cuales mantienen comunicación activa entre los miembros del equipo de forma tal, que todos conocen lo que todos están realizando siempre. Al finalizar el *sprint*, se presenta un incremento funcional del producto. El equipo realiza una retrospectiva a modo de resaltar lo bueno e intentar mejorar lo malo. Y el ciclo empieza de nuevo, planificando un nuevo sprint hasta que se haya concluido el proyecto.

En el caso de las aplicaciones web, en las que es muy común el cambio en los requerimientos durante la etapa de desarrollo, y donde conocer el tiempo exacto en el que el proyecto estará listo para salir a producción toma especial importancia pues generalmente se debe preparar el mercado objetivo, implementar SCRUM presenta una serie de ventajas, pues el equipo es capaz de definir con un alto porcentaje de exactitud el tiempo en el que desarrollará el producto, y por sus iteraciones que producen pequeños incrementos funcionales, el dueño del producto puede ir evaluando el resultado e ir haciendo

observaciones de cambios o mejoras, a las cuales el equipo es capaz de adaptarse fácilmente.

El objeto de este trabajo de tesis es ejemplificar la implementación de scrum en proyectos de desarrollo de aplicaciones web, ejemplificando las diferentes situaciones que pudieran suscitarse durante cualquier proyecto de desarrollo y mostrando cómo la metodología puede ayudar a manejarlas, haciendo al equipo fácilmente adaptable a cambios y, por la naturaleza de sus iteraciones, poder prever problemas o dificultades y si no es posible evitarlos, si pueden prepararse para lidiar con ellos.

OBJETIVOS

General

Ejemplificar la implementación de la metodología SCRUM para proyectos de desarrollo de aplicaciones web.

Específicos

1. Realizar una investigación documental para conocer la metodología SCRUM.

2. Definir las ventajas y desventajas de desarrollar aplicaciones web.

3. Definir teóricamente el uso de la metodología SCRUM para el desarrollo de aplicaciones web en al menos 3 entornos ficticios, definiendo diferentes circunstancias probables.

4. Realizar una encuesta a los estudiantes de Ingeniería en Ciencias y Sistemas de la Universidad de San Carlos de Guatemala sobre el conocimiento que poseen sobre la metodología SCRUM.

INTRODUCCIÓN

Al planificar el desarrollo de un proyecto es necesario definir las metas, el equipo de trabajo, los recursos que se asignan, el marco temporal y una estrategia de actuación. Esta estrategia puede definirse basándose en distintas metodologías ya existentes o diseñando la propia, según las necesidades y bondades del equipo encargado del proyecto.

Cuando se habla de proyectos de desarrollo de *software*, existen dos líneas de trabajo:

- Las metodologías formales que son consideradas rígidas, normativas y dependientes de planificación detallada previa al desarrollo mismo, basadas en una gestión de planificación y seguimiento y que requieren extensa documentación; ejemplo: CMM, PMI, etc.

- Las metodologías ágiles que fueron definidas como alternativas a las anteriores, con una gestión basada en la flexibilidad y adaptación al cambio; ejemplo: XP, scrum, etc.

En el presente trabajo de tesis se describe la metodología ágil SCRUM y su posible implementación en el desarrollo de proyectos web; se listan los elementos involucrados en la metodología; su ciclo de vida y las ventajas y desventajas que pudieran obtenerse de su utilización. Además, se describen casos sobre la implementación de SCRUM en distintos escenarios para poder evaluar cómo se puede adoptar la metodología para dar valor extra al desarrollo.

1. SCRUM

En 1986, Takeuchi y Nonaka describieron cómo Honda, Canon y Fuji-Xerox producen nuevos productos a nivel mundial con un enfoque escalable y basados en equipos autoorganizados, comparándolo con una jugada de rugby: "el enfoque de 'carrera de relevos' para el desarrollo de productos entra en conflicto con el objetivo de obtener la máxima velocidad y flexibilidad. En su lugar un enfoque como el rugby – donde el equipo intenta avanzar como equipo, enviando el balón hacia atrás y luego avanzar – sirve mejor a los desarrollos competitivos que se ven hoy en día."[1]

En 1993, Jeff Sutherland y su equipo crearon un proceso de desarrollo de software basado en los procesos descritos por Takeuchi y Nonaka, combinándolos con orientación a objetos. Ken Schwaber formaliza el proceso para la industria de desarrollo de software y en la OOPSLA (*Object-Oriented Programing, Systems, Languages & Applications*) de 1995 publican juntos el primer informe de SCRUM.

Scrum es un marco de trabajo, un conjunto de buenas prácticas que ofrece valor al producto final y agilidad en el desarrollo, dando respuesta a los principios de desarrollo ágil[2], especialmente:

- Entrega temprana y continua de *software* de valor.

[1] TAKEUCHI Hirotaka, NONAKA Ikujiro. *The new new product development game.* p. 5.
[2] BECK Kent; BEEDLE Mike; VAN BENNEKUM Arie; COCKBURN Alistair, CUNNINGHAM Ward; FOWLER Martin; GRENNING James; HIGHSMITH Jim; HUNT Andrew; JEFFRIES Ron; KERN Jon; MARICK Brian; MARTIN Rober Cecil; MELLOR Steve; SCHWABER Ken, SUTHERLAND Jeff y THOMAS Dave. *Manifiesto por el desarrollo ágil de software*. p. 187.

- Las personas del negocio y los desarrolladores deben trabajar juntos de forma cotidiana.
- Un *software* que funciona es la principal medida de progreso.

En palabras muy simples, scrum puede explicarse como un método de gestión de proyectos: el equipo de desarrollo toma las historias (requerimientos) ya priorizadas por el cliente; las organiza en grupos de tal manera que puedan ser completamente terminadas en un período de tiempo definido; trabajan en forma autónoma reuniéndose diariamente para actualizar al equipo sobre los avances, planificaciones y posibles problemas enfrentados.

1.1. Componentes

1.1.1. Roles

1.1.1.1. El dueño del producto

Es el cliente o su representante. Deberá estar empoderado para la toma de decisiones, conoce el producto, el plan de inversión y el retorno esperado de dicha inversión. Decide cómo será el producto final y el orden de desarrollo, es decir, es el encargado de establecer las prioridades del cliente y las refleja estableciendo la prioridad de las funcionalidades del producto final.

Es completamente necesario que el dueño del producto esté involucrado en el proceso y asista a las reuniones de planificación, pues es la única persona que puede incluir o eliminar historias a la pila del producto; sin embargo, no puede interferir en la planificación del tiempo que tomará su desarrollo.

1.1.1.2. El equipo de desarrollo

Es un grupo multifuncional, aunque pueden existir especialistas en temas específicos; no se tiene un diseñador, un programador y un tester designado, sino que todo el equipo es responsable del proyecto completo y colaboran entre sí para asignarse las tareas necesarias.

El número de miembros del equipo no debería ser muy grande (3-9 personas) ya que así es más fácil la autogestión. Son los encargados de valorar el tiempo que tomará el desarrollo de las historias ya priorizadas por el dueño del producto. Deben ser personas colaboradoras entre sí para que puedan trabajar en forma cohesionada y autoorganizada.

Se recomienda siempre que el equipo de desarrollo esté físicamente junto, de no ser así, al menos suficientemente cerca como para poder compartir ideas y solucionar problemas sin tener que perder tiempo en movilizarse; también, lo suficientemente aislado del resto de empleados para que puedan reunirse sin ser interrumpidos y no interrumpir a los demás.

1.1.1.3. Scrum master

Es el responsable del cumplimiento de las reglas, proporciona asesoría tanto al equipo de desarrollo como al dueño del producto.

Es el líder del grupo, encargado de dirigir las reuniones de planificación y scrum diarios; revisa y valida la pila del producto y de los *sprints*; debe procurar solucionar los problemas que puedan entorpecer el avance del trabajo y debe tener conocimientos avanzados del proceso de scrum para poder guiar al equipo y procurar las mejores prácticas.

1.1.2. Artefactos

1.1.2.1. La pila del producto (*backlog*)

Es una lista ordenada de todas las características, funcionalidades y requisitos que el cliente espera obtener del producto; debe incluirse todo lo que supone trabajo para el equipo de desarrollo. Es importante que sea sencilla y entendible, cada elemento debe ser expresado con una sentencia en lenguaje simple y hacer referencia a funcionalidades entregables. Cada elemento de la pila es conocido como historia de usuario.

Es el dueño del producto, el encargado de definir las historias y de priorizarlas según el valor que cada una aporte al producto final. El equipo de desarrollo será el encargado de estimar el tiempo de desarrollo para cada una.

Aunque la pila del producto no tiene un estándar definido en su formato, sí hay elementos que no pueden faltar:

- Descripción de la funcionalidad: conocida como historia
- Prioridad: establecida por el dueño del producto
- Estimación de tiempo: establecida por el equipo de desarrollo

Generalmente, pueden agregarse todos aquellos datos que el equipo considere necesarios o que los ayuden a organizar el trabajo de mejor manera, por ejemplo:

- Identificador: para facilitar el seguimiento.
- Observaciones: que se consideren importantes acerca de la historia.
- Descripción de las pruebas de funcionalidad: criterios de validación.

- Categoría: una categorización simple para identificar las historias como *backoffice* u optimización.
- Algún espacio para seguimiento de errores con lo que pudiera quedar de la siguiente forma.

Tabla I. **Ejemplo de la pila del producto**

Pila del Producto (ejemplo)							
id	**Historia**	**Imp**	**Est**	**Observaciones**	**Pruebas**	**Categoría**	**Bugs**
1	Registro de visitas	50	8		Llenar el formulario, ir al listado y que esté ingresado	Front-end	
2	Asignar turnos	20	3	Tomar en cuenta el grupo prioritario	Ingresar una visita de prioridad 0 y una de prioridad 1, ver que asigna primero al de prioridad 1.	Front-end	

Fuente: elaboración propia.

Bill Wake define el acrónimo en inglés INVEST con un listado de características que debe tener una buena historia de usuario.[3]

- Independientes: las historias de usuario deben ser independientes una de la otra; si existiera el caso en el que una historia depende de otra, el autor recomienda unirlas en una sola.

[3] WAKE, Bill. *INVEST in good stories, and SMART tasks*. http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/. Consulta: 11 de octubre de 2016.

- Negociables y negociadas con el equipo de desarrollo: las historias no pueden ser inamovibles.

- Valorables: el dueño del producto pone su valoración a cada una, dependiendo la importancia que tenga para el negocio, esto establece la prioridad de la historia.

- Estimables: de tal forma que el equipo pueda medir o estimar el tamaño de la historia y el tiempo o esfuerzo que requerirá completarla.

- Pequeñas: en la medida que puedan valorizarse y planificarse con cierto nivel de certeza.

- Testeables: cada historia debe tener una forma de validar si está completa o no. Es por eso que las historias deben ser descripciones funcionales del proyecto.

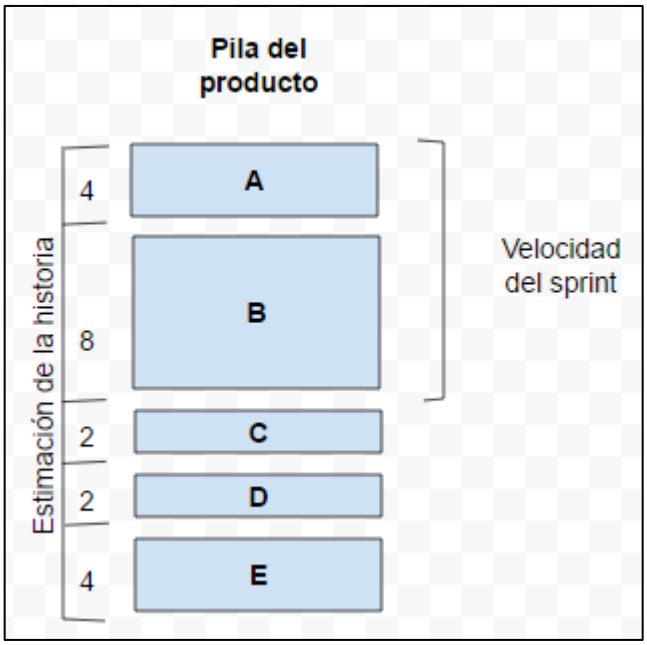
1.1.2.2. La pila del *sprint*

Es el grupo de historias en las que el equipo de desarrollo trabajará en la próxima iteración del proyecto; salen de la pila del producto; se seleccionan en orden según la prioridad que les ha calificado el dueño del producto. Para seleccionar cuántas historias incluirán en el *sprint*, es necesario que el equipo determine el tiempo que se dedicarán al desarrollo y, basado en la estimación de tiempo que se le dio a cada historia del proyecto, se incluyen al *sprint* las que se puedan terminar totalmente; recuérdese que una historia parcialmente terminada no es una historia terminada.

Es recomendable que las historias más grandes sean detalladas en tareas más pequeñas, preferiblemente realizables en un día. Esto ayudará al control del desarrollo del *sprint*. Cada tarea es asignada, entonces, a un miembro del equipo para que la lleve a cabo.

Es importante que la pila del *sprint* sea pública para el equipo; puede ponerse en una pizarra o en una pared con carteles para que todos la tengan presente y puedan llevar el control de lo que ya se ha terminado. Durante el *sprint*, la lista es actualizada con las tareas se han terminado y el esfuerzo que hace falta en las tareas pendientes.

Tabla II. **Selección de la pila del *sprint***



Fuente: elaboración propia.

En el momento en que se define la pila del *sprint*, es posible aún negociar con el dueño del producto; es decir, si este desea que se incluyan más historias en el *sprint*, se le puede dar la opción de: mover las prioridades de tal forma que C, D y E entren en el *sprint* en vez de B; de descomponer B en historias más cortas, tal vez con distintas prioridades y de este modo entrarían algunas subhistorias de B y C y D, por ejemplo. Lo importante es que las historias que abarque el *sprint* sean siempre totalmente terminadas.

1.1.2.3. Incremento

Es la parte del producto que se ha terminado en una iteración, debe ser totalmente funcional y potencialmente entregable: terminada y probada según lo especificado en la pila del proyecto. Si el sistema requiere documentación o intervención de terceros, estos deberán estar finalizados también para afirmar que el incremento está hecho.

1.1.3. Eventos

1.1.3.1. *Sprint*

Se conoce como *sprint* al período de tiempo en que se produce un incremento del proyecto. No debe ser un período muy largo, se recomienda entre 2 y 6 semanas.

1.1.3.2. Planificación del *sprint*

Es una reunión que debe coordinarse con el equipo de desarrollo, el scrum master y el dueño del producto, con el objetivo de planificar el *sprint*. La reunión suele ser extensa, al menos un día completo, por lo que cada uno de

los asistentes debe tener el tiempo suficiente; y se recomienda hacer pequeñas pausas para relajarse y poder mantener la atención en el tema.

Previo a la reunión de planificación, es necesario que se tenga la pila del producto ya organizada según las prioridades del dueño del proyecto. Los productos de esta reunión deben ser:

- Fecha y hora de la revisión del *sprint.*

- Lugar y hora de los scrum diarios.

- Meta del *sprint*: debe responder a la pregunta: ¿por qué está haciendo el sprint?, en palabras que todos entiendan. Debe ser una acción que no se ha logrado aún para darle sentido al trabajo y recordarle al equipo en todo momento el porqué de su trabajo. Ejemplos de meta podrían ser: ganar más dinero, o añadir soporte al *backoffice*.

- Duración del *sprint*: depende del nivel de compromiso del equipo. Los dueños del producto los prefieren más cortos y los desarrolladores los prefieren más largos; es solamente cuestión de comprometerse ambas partes e ir probando la duración perfecta para el equipo.

- Elementos que formarán la pila del *sprint*: son tomadas de la pila del producto. El equipo valora cuánto tiempo, ya sea en horas-hombre o días-hombre, es necesario para cada historia de la pila y así se podrá definir cuántas historias entrarán en el *sprint*. La tarea de estimar tiempos suele ser compleja y pueden existir varias formas para hacerlo; según la experiencia del equipo pueden basar sus estimaciones en cálculos informales de cuánto tiempo les tomaría el desarrollo dependiendo de la

complejidad de cada historia; también, es posible estimar los tiempos basados en el historial del equipo, si es que ya tienen tiempo trabajando juntos o basados en los tiempos del sprint anterior, pueden proyectar el tiempo que les tomará cada historia; por lo que la estimación más complicada se da siempre cuando se trata del primer *sprint*.

Para estimar los tiempos, debe considerarse el nivel de compromiso de cada miembro del equipo, esto es, cuánto tiempo del día realmente se dedicará al desarrollo del proyecto dependiendo si tiene otras responsabilidades.

Aunque los anteriores son los productos básicos de la reunión de planificación, es importante que se dedique tiempo para detallar las tareas que incluye cada historia y su responsable. De tal cuenta, el resultado sea un detalle de historias, tareas, tiempos y responsables. Esta información debe ser de conocimiento de todo el equipo y es aconsejable que se coloque en un lugar visible todo el tiempo: una pizarra una pared con carteles pegados. Esto servirá en los scrum diarios para la actualización del trabajo.

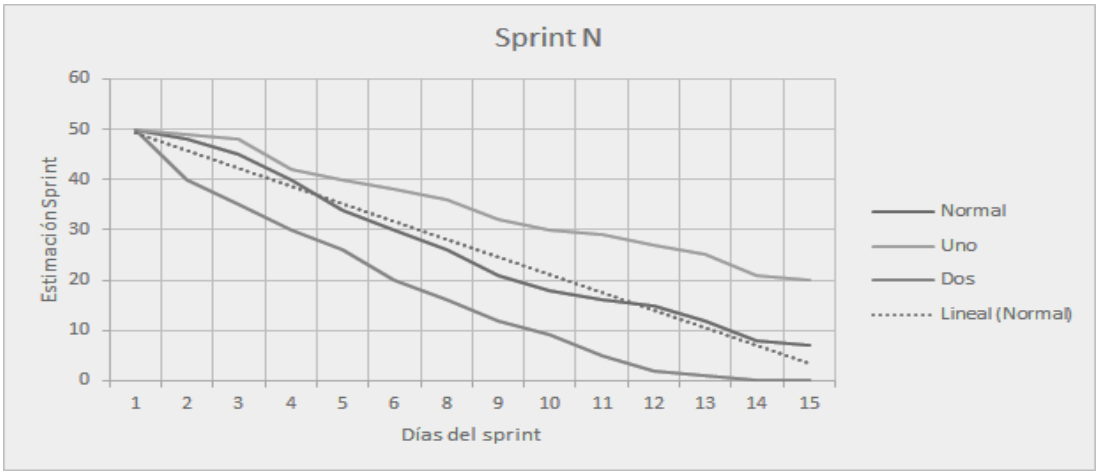
1.1.3.3. Scrum diario o reunión diaria

Reunión diaria breve de no más de 15 minutos. Los asistentes están integrados por el equipo de desarrollo y el scrum master; pueden presenciarla otras personas, pero no pueden participar en esta. Debe hacerse frente al tablero de la pila del *sprint* y con los participantes de pie, ayudando esto a mantener la atención de todos y a que sea corta. Cada miembro del equipo explica que hizo desde el scrum diario anterior y que tiene planificado hacer hasta el scrum diario próximo. Plantea, también, si ha tenido algún problema o alguna necesidad; el scrum master se encarga de facilitar las soluciones a dichos planteamientos. No se trata de una reunión de control del scrum master,

al contrario, la reunión la dirige el equipo de desarrollo; se trata de socializar la información, compartir los avances y solucionar problemas.

Se recomienda el uso de dos diagramas para controlar las actividades: el primero es un tablero que liste las tareas pendientes por hacer, las que se están haciendo y las que ya están terminadas, esto puede ser tan simple como utilizar pequeñas hojas autoadhesivas para cada tarea y de esta forma, el encargado de dicha tarea la mueve a la columna según el avance realizado; también, se utiliza un diagrama de tiempo vs avances programados, llamado *burndown*, donde cada día se mueve el punto según el avance alcanzado por el equipo.

Figura 1. ***Burndown***

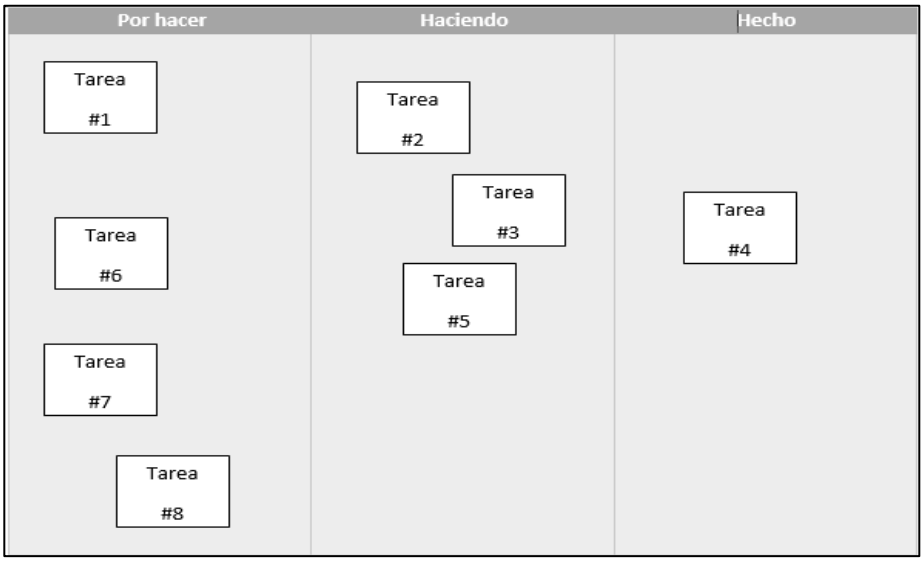


Fuente: elaboración propia.

En la figura anterior se observa cómo se ha ido avanzando en el trabajo respecto al tiempo; en la serie uno se muestra el avance cuando el equipo no calculó correctamente el esfuerzo a realizarse por cada tarea asignada, lo que los deja con poco tiempo y mucho trabajo pendiente. La serie dos muestra cómo luciría la gráfica cuando el equipo sobrevalúa el esfuerzo de las tareas

dando como resultado mucho tiempo sobrante en el *sprint*. En ambos casos el equipo aprende y cometerá menos errores en el siguiente *sprint*. Lo ideal será que se comporte como lo muestra la serie normal, aunque llegar a ese nivel de certidumbre en los cálculos dependerá de la experiencia del equipo.

Figura 2. **Tablero de tareas**



Fuente: elaboración propia.

1.1.3.4. Revisión del *sprint*

Reunión a la que asiste el dueño del producto, el equipo completo de desarrollo y el scrum master; como siempre otras personas son bienvenidas, pero no pueden participar en la misma.

El equipo presenta el incremento al dueño del producto y este verifica las funcionalidades del mismo evaluando cuáles considera hechas y cuáles no. Lo importante de esta reunión es obtener la retroalimentación de parte del dueño

del producto sobre las funcionalidades desarrolladas, la identificación de posibles cambios en los requerimientos y la detección de errores.

El scrum master actúa como moderador y, al final de la reunión, se acuerda cuándo y dónde será la reunión de planificación del siguiente *sprint*.

1.1.3.5. Retrospectiva

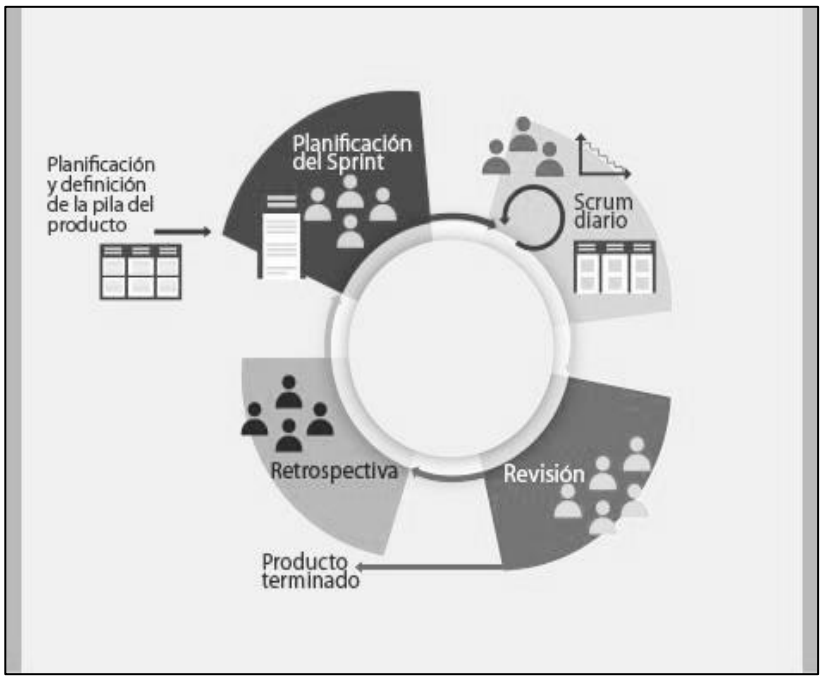
Reunión posterior a la revisión del *sprint*, ya solo con el scrum master y el equipo de desarrollo, incluso es un poco más informal, más en confianza de equipo. En esta, el equipo discute las dificultades que enfrentó durante el *sprint* y se trata de analizar cómo puede mejorarse para el siguiente. Se evalúa si la duración del *sprint* es la adecuada para el equipo y si se ha acertado en los cálculos de tiempos para las actividades. Toda esta información es considerada para la planificación del siguiente *sprint*. Esta retrospectiva interna del equipo ayuda a mantenerlo en mejora continua.

Debe considerarse que la reunión para hacer retrospectiva no debe tomarse como una oportunidad para acusar a los miembros del equipo si algo no sale respecto a lo planificado; al contrario, es mejor agradecer los esfuerzos realizados, aceptar los errores cometidos y comprometerse a las mejoras necesarias.

1.2. Ciclo de vida

Scrum plantea un desarrollo con incremento iterativo, es decir, al final de cada iteración o *sprint*, se obtendrá un producto totalmente funcional y potencialmente entregable.

Figura 3. **Ciclo de vida de scrum**



Fuente: elaboración propia.

Como se puede observar en la figura anterior, para iniciar el proceso de desarrollo con scrum es necesario que el cliente haya definido completamente la pila del producto; esto, claro está, puede darse con la ayuda y asesoría del equipo de trabajo. Esta pila es la entrada al ciclo que comienza con la planificación del *sprint*, a continuación, y durante toda la duración, se hacen las pequeñas reuniones o scrum diarios. Al finalizar el *sprint*, se presentan los incrementos en la revisión y, posteriormente, se reúne el equipo de desarrollo y el scrum master para realizar la retrospectiva del *sprint* para iniciar otra vez con la planificación del siguiente *sprint*.

Aunque se espera que el equipo de trabajo sea lo más acertado posible en la estimación de tiempos y cálculos de duración del *sprint*, con ciertas holguras

en el proceso, se debe considerar la posibilidad de que se llegue a un punto en el que el equipo definitivamente no logre cumplir con las tareas programadas.

Gracias a las reuniones diarias, en las que se va evaluando el avance de cada miembro del equipo e identificando los problemas que se van presentando, el scrum master, al ir actualizando el *burndown* del *sprint*, puede proyectar si es posible o no cumplir con lo acordado y de esta forma pueden ir buscando soluciones a los atrasos o readecuando el orden de tareas para cumplir con el mayor número posible.

Si aún con los mayores esfuerzos no es posible completar las tareas planificadas en el *sprint*, no queda más que presentar los pequeños avances en el día acordado para la revisión del *sprint*, evaluar bien la situación que afectó el proceso en la reunión de retrospectiva y, en la reunión para la planificación del próximo *sprint*, agregar las tareas pendientes del *sprint* anterior, corriendo así todas las tareas.

Si bien es comprensible que existan atrasos, sobre todo cuando son provocados por situaciones externas al equipo de desarrollo o situaciones emergentes de cualquier otra índole que no se relacione con el compromiso del equipo al proyecto, sí es importante observar si estos atrasos son muy frecuentes; de ser así, estará evidenciándose un problema de cálculo a la hora de la planificación y ponderación de las tareas y, contrario a lo que pueda suponerse, no se soluciona alargando la duración de los *sprint*. Lo recomendable en estos casos es por el contrario acortar el tiempo de duración. Si, acortarlo, y tratar de detallar mejor las tareas, a modo de que queden tareas pequeñas; esto porque es mucho más fácil ponderar la duración de trabajo en una tarea pequeña que en una más complicada (grande); de esta manera se logra minimizar el error en la planificación.

El ciclo se rompe o termina luego de la revisión del último *sprint*, es decir, cuando ya se han desarrollado todas las historias de la pila del producto y se presentan al cliente.

1.3. Ventajas y desventajas

1.3.1. Ventajas

- Su estrategia es el desarrollo incremental con fases que pueden solaparse en el proceso, no requieren un ciclo secuencial o de cascada.

- Las iteraciones en el proceso permiten verificar la calidad y funcionalidad, que ayudan a la detección temprana de errores y malos entendidos con el cliente, se pueden así ajustar las características del producto final sin impactar de manera significativa el tiempo del desarrollo total.

- Dadas las reuniones diarias del equipo durante el *sprint* en las que se plantean los problemas que el equipo pueda estar enfrentando, es posible proyectar el cumplimiento que se alcanzará en relación al incremento programado y, de no ser el esperado, es posible modificar el orden de la pila o tomar algunas acciones correctivas que puedan ayudar a que se logre cumplir con lo programado.

- La calidad del resultado se basa más en el conocimiento tácito de las personas en equipos autoorganizados que en la calidad de los procesos empleados.

- El proceso requiere que el dueño del producto participe activamente, haciéndolo parte del mismo, lo que facilita la comunicación con el equipo

de desarrollo para cuando surjan dudas respecto a las funcionalidades esperadas.

- El equipo de desarrollo es multidisciplinario, con lo cual todos pueden hacer de todo, por lo cual las tareas puedan solaparse. Por ejemplo, un miembro del equipo puede diseñar, desarrollar y probar la tarea que le fue asignada; al mismo tiempo que el otro compañero diseña, desarrolla y testea la suya.

1.3.2. Desventajas

- La dificultad de estimar los tiempos, tanto de la duración del *sprint* como de la duración de cada historia, sobre todo en equipos que son nuevos en scrum.

- No existe un manual de cómo implementar scrum, no hay una forma correcta o incorrecta de implementarlo, solamente son formas de hacer las cosas, pero cada equipo debe adecuarlo a su propia condición. Al principio tal vez no funcione como se esperaba, pero es viable adaptarlo de tal manera que funcione.

2. APLICACIONES WEB

Una aplicación web es un tipo de aplicación que se almacena en un servidor al que se accede por medio de la red utilizando un navegador web y, a diferencia de las aplicaciones de escritorio, no necesitan instaladores ni se amarran a un sistema operativo específico ni a las capacidades de memoria o procesador de la máquina o dispositivo en que se acceda.

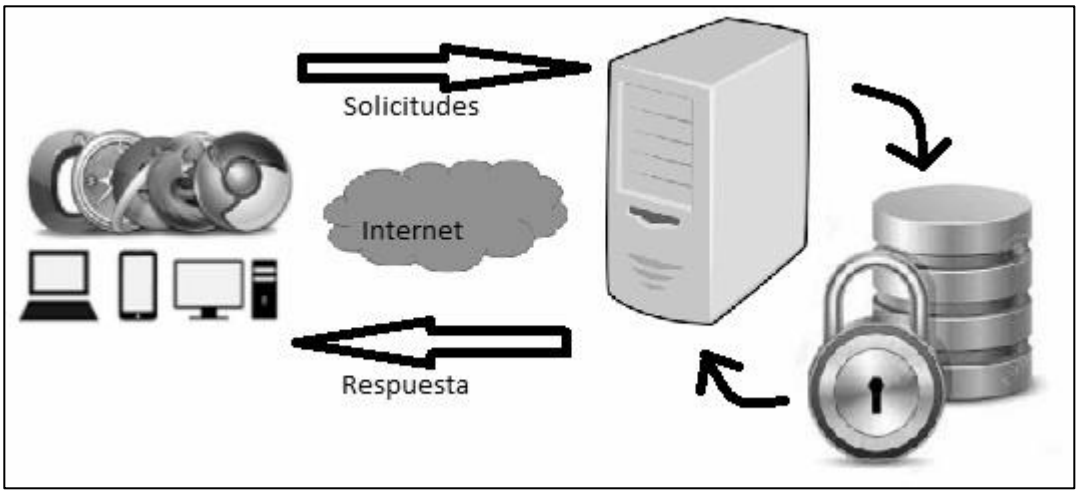
Generalmente, las aplicaciones web tienen un modelo de tres capas:

- Capa superior: el cliente, es el programa que interactúa con el usuario para hacer las peticiones al servidor, es un navegador de internet como Internet Explorer, Mozilla, Chrome, Opera, Safari, entre otros. Al cliente se le presenta una vista desarrollada en HTML (*hyper text markup language*, lenguaje de marcas de hipertexto) utilizando CSS para mejorar la presentación y algún código ejecutable como JavaScript o VBScript que lo hará más amigable.

- Capa intermedia: el servidor web es un programa que se mantiene en espera de las solicitudes del cliente, las atiende y envía una respuesta. Es el que procesa los datos y maneja los recursos multimedia que pudieran emplearse para ser descargados o ejecutados por parte del cliente. Para programar la aplicación pueden utilizarse diferentes lenguajes que corren o se interpretan del lado del servidor como php, java, ruby, entre otros. La comunicación cliente-servidor se realiza utilizando el protocolo HTTP (*hypertext transfer protocol*, protocolo de transferencia de hipertexto) o HTTPS, la versión segura de HTTP.

- Capa inferior: la persistencia de los datos, generalmente almacenados en bases de datos. Existen varios manejadores de bases de datos: SQL, MySQL, Oracle, entre otros. Esta última capa puede ser física con un servidor encargado únicamente de almacenar los datos o simplemente ser una capa conceptual y que los datos se almacenen en el mismo servidor web.

Figura 4. **Aplicaciones web**



Fuente: elaboración propia.

Las aplicaciones web funcionan de la siguiente manera:

- El usuario ingresa a un navegador web y especifica la dirección URL de la aplicación.

- El cliente establece una conexión con el servidor.

- El cliente hace la solicitud.

- El servidor ejecuta el código necesario para obtener los datos solicitados por el cliente y envía estos datos de vuelta al cliente, en su defecto, envía un mensaje de error.

- El cliente interpreta la respuesta del servidor y construye la página que presenta al usuario.

- Se cierra la conexión entre el cliente y el servidor.

- Se muestra la página al usuario.

2.1. Ventajas y desventajas de las aplicaciones web

2.1.1. Ventajas

- Actualizaciones más sencillas, solamente se actualiza la aplicación fuente en el servidor.

- Accesible desde cualquier punto conectado a la red.

- Centralización de los datos; con un buen esquema de seguridad es más fácil protegerlos.

- Independiente de la plataforma del dispositivo que utilice el usuario, libre de problemas de compatibilidad; para acceder a la aplicación solo se necesita un navegador web.

- Puede propiciar la colaboración entre usuarios.

- Alta disponibilidad, puede ofrecerse el servicio desde múltiples puntos para garantizar la disponibilidad.

2.1.2. Desventajas

- La disponibilidad de acceso depende de la red; si la red está saturada, se afecta el desempeño de la aplicación. Si la red se cae, simplemente no hay forma de acceder, depende totalmente del proveedor del servicio de red.

- Seguridad, uno de los problemas de mantener una aplicación en la nube es que no faltará el tipo de usuario que pretenda causar problemas como denegación de servicios, obtener una copia de los datos, entre otros. Por lo tanto, es conveniente invertir tiempo y dinero en mantener los datos seguros, que su acceso sea controlado.

2.2. Desarrollo de una aplicación web

En el desarrollo de aplicaciones web, como en el de cualquier otro tipo de aplicaciones, siempre es recomendable seguir cierto proceso organizado para asegurar que se cumplan con las especificaciones funcionales deseadas y que el producto tenga la calidad esperada.

2.2.1. Definición

Antes de iniciar con el desarrollo, debe definirse exactamente qué es lo que se desea construir, establecer de forma clara la funcionalidad esperada de la aplicación, ésta definición abarca:

- Definir el aporte que la aplicación tendrá a los usuarios. Qué se les va a ofrecer y qué necesidades estaría satisfaciendo.

- Definir el grupo al que va dirigido, esto conlleva un análisis previo del sector de mercado que se quiere abarcar.

- Definir y limitar la inversión del proyecto.

- Definir la tecnología y el tipo de servidor con el que se cuenta, incluso establecer el lenguaje de programación y manejador de base de datos con el que se desea trabajar.

2.2.2. Diseño

Roger Pressman, en su libro *software engineering a practitioner's approach*, define una pirámide de diseño para las aplicaciones web.

Figura 5. **Diseño de una aplicación web**

Fuente: elaboración propia.

2.2.2.1. Diseño de la interfaz

El objetivo es crear una interfaz consistente entre la funcionalidad y el contenido que guie al usuario en la interacción con la aplicación. El diseño de la interfaz define detalles como: la página de inicio, la página de bienvenida, el menú: vertical horizontal, el uso de imágenes para los enlaces o no.

2.2.2.2. Diseño estético

Representa la parte artística del diseño: cómo presentar visualmente el contenido de manera que el usuario lo encuentre atractivo.

2.2.2.3. Diseño del contenido

Se refiere a la estructura de la información, el diseño de los objetos que manejan esta información y la relación entre estos. Como la información se almacena y los procesos para su recuperación y la manera en que se entregará para presentarla al usuario.

2.2.2.4. Diseño de la arquitectura

Esta parte del diseño amarra el contenido, los usuarios y las políticas de navegación. Es la forma en que el contenido se estructura y cómo la aplicación maneja la interacción con el usuario, los mapas de navegación. Existen distintas formas: la arquitectura lineal, en grid o jerárquica.

2.2.2.5. Diseño de navegación

Identificar los componentes de la arquitectura, es decir, las páginas que se crearán para abarcar la funcionalidad, pueden diseñarse los caminos (*path*) que permitan el acceso a los mismos.

2.2.2.6. Diseño de componentes

Diseño y construcción de los programas que manejen los datos y generen las páginas que conforman la aplicación.

2.2.3. Implementación

Se refiere a la codificación o programación del código de la aplicación.

2.2.3.1. *Front-end*

Es la parte que abarca la interacción con el usuario, la parte visual, lo intuitivo, lo amigable de la aplicación, todo lo que se refiere al lado del cliente, los colores, el diseño, los botones, las imágenes, etc.

Para desarrollar el *front-end* se utiliza:

- HTML: lenguaje de marcado fundamental para crear y organizar el contenido para que pueda ser desplegado en el navegador.

- CSS: lenguaje de diseño gráfico para dar estilo y presentación en un documento estructurado.

- JavaScript: lenguaje de programación interpretado del lado del cliente, lo que significa que para ejecutarse no se necesita hacer otra petición al servidor; es utilizado para hacer las aplicaciones web más dinámicas, por ejemplo, al momento de hacer validaciones o agregar dinamismo a los menús.

2.2.3.2. *Back-end*

Es la parte de la aplicación que el usuario no ve, la encargada del procesamiento y almacenamiento de la información; se encarga de que el *front-end* realmente funcione. Implementa la lógica del negocio.

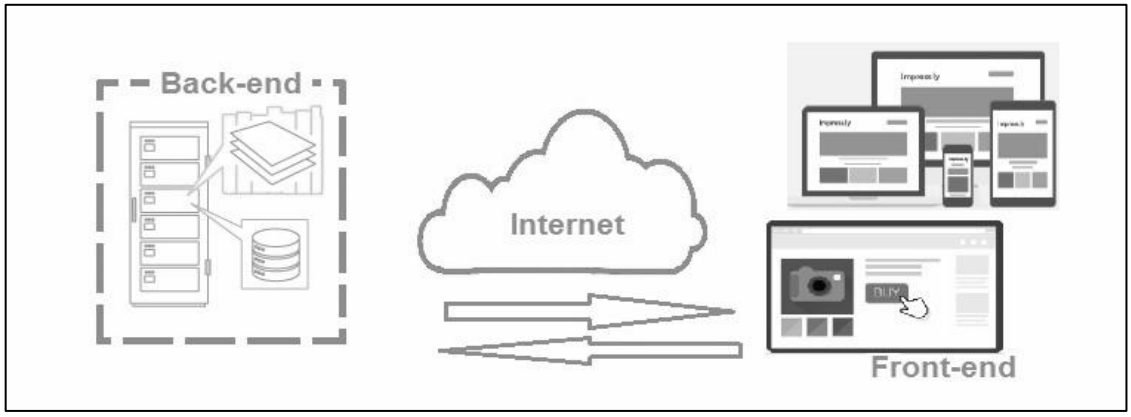
Para el *back-end*, es necesario que el desarrollador tenga conocimientos de:

- Programación en lenguajes web como Ruby, Phyton, PHP, JAVA, que son los más usados. Se puede optar por desarrollar desde cero la aplicación o puede apoyarse con la utilización de *frameworks*, de los cuales también existen varios para cada lenguaje que se desee emplear.

- Base de datos, no solo la habilidad de diseñar, sino la capacidad de utilizar los distintos manejadores que ofrece el mercado como SQL, MySQL, Oracle, entre otros.

- SEO (*search engine optimization*, optimización de motores de búsqueda) que se refiere al proceso de mejorar la visibilidad de un sitio o aplicación web en los resultados de los distintos buscadores como Google.

Figura 6. ***Back-end* / *Front-end***



Fuente: elaboración propia.

2.2.4. Pruebas

Como en cualquier aplicación, es necesario hacer pruebas de funcionalidad para validar así sus requerimientos. Las pruebas generales son:

- Pruebas unitarias: verifican el funcionamiento de cada módulo o cada funcionalidad de la aplicación.

- Pruebas de integración: una vez superadas las pruebas unitarias, es necesario comprobar que el conjunto funciona bien.

- Pruebas de regresión: aseguran que una vez resuelto un problema o error, no se haya comprometido la funcionalidad.

Pero cuando se trata de una aplicación web, otro set de pruebas se hace necesario dada la naturaleza del ambiente web y el uso de este tipo de aplicaciones. Entre otras, se pueden mencionar las siguientes:

- Pruebas de contenido: a nivel sintáctico y semántico, así como verificar que no haya problemas de traducción si se utilizan diferentes lenguajes.

- Pruebas de usabilidad: para evaluar el grado en que los usuarios interactúan con la aplicación.

- Pruebas de navegabilidad: para detectar la existencia de enlaces muertos o que realmente conduzcan a donde se espera que lo hagan.

- Pruebas de compatibilidad con los distintos navegadores: para asegurar que funcione en todos o al menos en la mayoría.

- Pruebas de seguridad o pruebas de penetración: para comprobar el nivel de seguridad de los datos almacenados y minimizar así las posibilidades de ataques maliciosos.

- Pruebas de estrés: para verificar la robustez y alta disponibilidad de la aplicación en condiciones extremas; se basa en enviar excesivas peticiones al servidor, así como el uso de la aplicación con limitaciones en el hardware del cliente.

2.2.5. Liberación

No es más que la publicación de la aplicación; si se habla de aplicaciones comerciales, la liberación puede incluir la promoción.

2.3. Perfil de los involucrados en el desarrollo web

En el inciso anterior se mencionan las fases que implica el desarrollo web; con el tiempo, un desarrollador obtiene su experiencia, pero es casi imposible que logre dominar todos los aspectos de este tipo de desarrollo en un 100 %; por lo tanto, se tiende a separar al equipo o distinguir en él algunas especialidades; esto, por supuesto, no implica que no puedan apoyarse unos a otros, aunque no sea su área. En 2011, Alain Alejo, un especialista en desarrollo de aplicaciones web, publicó un listado de los perfiles más comunes en un equipo de desarrollo web.

2.3.1. Diseñador web

Se encarga de armar la presentación visual de la aplicación, sin funcionalidad, puramente el diseño del aspecto final que tendrá la aplicación. Generalmente, sus herramientas serían:

- Photoshop
- Flash
- HTML y CSS

2.3.2. Maquetador

Traduce la plantilla entregada por el diseñador a lenguaje de marcado y estilos para que pueda ser interpretado por el navegador. Puede también agregar la funcionalidad del lado del cliente, para que la aplicación se vea como lo propuso el diseñador, pero ya con interactividad. Sus herramientas pueden ser:

- Photoshop
- HTML y CSS
- Javascript

2.3.3. Programador *front-end*

Convierte el diseño visual e interactivo en código *front-end*. Aunque su perfil es muy parecido al del diseñador web, tiene ciertas habilidades técnicas y de programación. Sus herramientas son:

- HTML/CSS
- Javascript, jQuery, JSON, AJAX
- PHP y MySQL

2.3.4. Programador *back-end*

Es el que desarrolla todo el código del lado del servidor. Debe estar involucrado en todo el proceso de desarrollo, incluyendo el diseño, pues es el encargado de diseñar la solución desde dentro; se encarga de los procesos de manejo de datos y de la implementación de la lógica de la aplicación. Sus herramientas son:

- HTML/CSS
- Javascript y jQuery
- Lenguajes de programación: PHP, Ruby, etc.
- Manejadores de bases de datos: MySQL, Oracle, etc.

2.3.5. Sysadmin

El administrador de sistemas, es el responsable de, una vez desarrollada la aplicación, mantener, monitorear, documentar y asegurar el correcto funcionamiento de la misma. Su papel es el de garantizar el rendimiento, uso de recursos y la seguridad de los servidores que administra. Generalmente su perfil incluye:

- Dominio de servidores web: FTP, SSH, Mail, DNS
- Conocimiento en redes

2.4. Ejemplos de aplicaciones web

2.4.1. Google Docs

Es un producto gratuito que ofrece Google. Se trata de un espacio en la red donde el usuario puede crear, almacenar y compartir documentos de texto, hojas electrónicas y formularios; aunque, recientemente, se agregaron también presentaciones y dibujos.

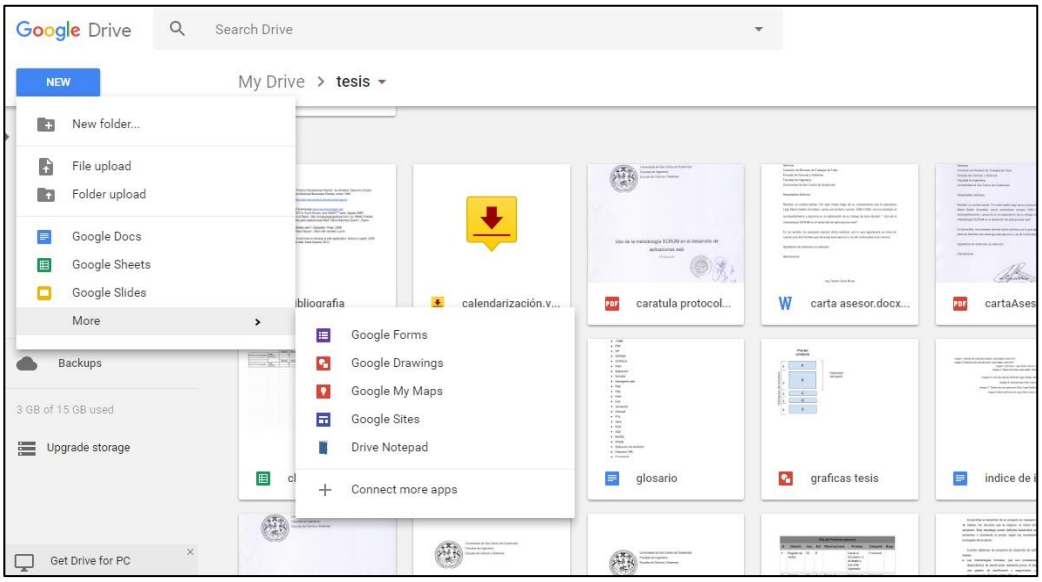
Como cualquier otra aplicación web, no requiere instalación de ningún tipo, solamente acceso a internet. Google agrega frecuentemente funcionalidades a sus aplicaciones, mismas que son transparentes al usuario y no afectan los documentos existentes.

Es posible la colaboración entre usuarios; aunque siempre existe un dueño del documento, este puede darle acceso y permisos de modificación a otros usuarios, o simplemente acceso de vista con opción a hacer comentarios.

Aunque Google ofrece con su Google Drive espacio de disco para almacenar los documentos, es posible también descargarlos a cualquier otro dispositivo y en distintos formatos.

Tiene compatibilidad con distintos editores de texto y hojas electrónicas de tal manera que es posible subir los documentos a la red y almacenarlos, compartirlos y/o editarlos en la nube. Otra de las bondades de la aplicación es que guarda automáticamente conforme se vayan agregando cambios, dando la opción a un historial completo de los documentos.

Figura 7. **Google Docs**



Fuente: elaboración propia.

2.4.2. Trello

Es una herramienta de colaboración gratuita para la administración de proyectos y actividades en la que es posible manejar tableros con distintas

listas de actividades; a cada actividad se le pueden agregar datos, como quién la lleva a cabo, y controlar desde el tablero el estado de cada una. Permite que cada miembro del equipo pueda actualizar en tiempo real el estado de sus actividades.

Trello muestra un tablero donde el usuario puede crear distintas listas; a cada lista puede agregarle un sinnúmero de actividades y a cada actividad puede pegarle una variedad de características: encargado, fecha de inicio y finalización, comentarios entre los colaboradores, etc.

Es una aplicación muy rápida y muy intuitiva de usar, permite el arrastrado de una actividad entre listas. Aunque su uso principal se ha enfocado en la administración de proyectos, manteniendo el control actualizado de cosas por hacer, las que se están haciendo y las que están terminadas; su uso puede ampliarse mucho más y adaptarse a las necesidades del usuario.

La colaboración entre los participantes es muy sencilla y las modificaciones se reflejan en tiempo real, pero si carece de un sistema de notificaciones de cambio, es decir, no envía notificaciones a los usuarios que no se encuentren conectados en el momento del cambio.

Figura 8. **Trello.com**

Fuente: elaboración propia.

2.4.3. Netflix

Es un servicio en línea de paga que ofrece al usuario un amplio catálogo de series, películas y documentales a las que, pagando una mensualidad, se puede acceder de forma ilimitada.

Una de las principales características de Netflix es la inteligencia que posee: es capaz de sugerir al usuario películas o series basado en las etiquetas de las que el usuario ha visto previamente.

Su catálogo es actualizado frecuentemente y permite que un usuario tenga acceso a la aplicación hasta en 5 dispositivos distintos y de forma concurrente, por la misma cuota mensual. No existe ningún contrato del servicio, el usuario

solicita el servicio dando los datos de la tarjeta de crédito o cualquier otra forma de pago en línea y el mismo puede cancelarse en cualquier momento.

Recientemente, ha impulsado su servicio *off-line*: al usuario la posibilidad de almacenar películas o series en su dispositivo de preferencia y luego poder reproducirlas con la aplicación en momentos sin conexión a internet.

Figura 9. **netflix.com**

Fuente: elaboración propia.

3. IMPLEMENTACIÓN DE SCRUM PARA EL DESARROLLO DE APLICACIONES WEB

3.1. Desarrollo de una aplicación web de hoteles

Supóngase que hay un equipo de desarrollo y que en este momento se tiene la solicitud para un nuevo proyecto. Entonces, el cliente se reúne con el director del equipo y le comenta su idea. Juntos trabajan en un listado de requerimientos que debe cumplir la aplicación. El director del equipo pasará a ser el dueño del producto, de ahora en adelante llamado DP, ya que dentro del equipo es el que sabe exactamente lo que el cliente solicita y cómo lo quiere; entiende la visión del cliente y la importancia que este le da a ciertos detalles y funcionalidades del proyecto; además, de poder comunicarse con el cliente en cualquier momento, este le confiere la potestad de tomar decisiones en cuanto al proyecto.

En la reunión, el cliente pide:

La aplicación web de búsqueda de hoteles, donde pueda o no crear una cuenta de usuario. Se hacen búsquedas por ciudad, número de estrellas del hotel y los precios de las habitaciones. Si se es usuario con cuenta en el sitio se puede entrar y comentar la experiencia en el hotel que se ha seleccionado: esta calificación irá de 1 a 5 estrellas; solamente los usuarios que tengan su cuenta creada. Habrá uno o varios administradores, encargados de ingresar los hoteles que desean aparecer en el sitio; este administrador podrá actualizar la información de los hoteles o eliminarlos del listado de hoteles disponibles.

De esta reunión, se genera, entonces, un listado de requerimientos que podría quedar de la siguiente forma:

Tabla III. **Requerimientos de la aplicación de hoteles**

id	Requerimiento	Descripción	Prio	Validación
1	*Login* de usuarios	Pequeño formulario de usuario y contraseña	200	Permite ingresar el usuario y contraseña y verifica que existe
2	Formulario de nuevo usuario	Formulario que pide ciertos datos para crear una cuenta de usuario	275	Presenta el formulario para nuevo usuario y guarda los datos ingresados
3	Formulario de administración de hoteles	Formulario para ingresar un nuevo hotel al sitio	250	Se llena el formulario y permite visualizar el hotel en el sitio
4	Búsqueda de hoteles	Filtro por precio, ciudad, número de estrellas	210	Se ingresan parámetros de búsqueda y lista los hoteles que los cumplan
5	Un solo hotel	Muestra los datos de un hotel específico, si se está logueado, se puede calificarlo	100	Se hace clic en el nombre del hotel y muestra su información
6	Inicio	Página de bienvenida a la aplicación	300	Se ingresa la URL del sitio y muestra un listado de los top 5 de los hoteles, según la calificación de los usuarios

Fuente: elaboración propia.

Ahora bien, del cuadro anterior, se preguntarán por qué la prioridad no va en orden de 1,2,3; pues bien, la explicación es sencilla: si en algún momento el DP decide que existe otro requerimiento que es más prioritario que el 4 pero menos prioritario que el 3, se tendría que modificar la prioridad de todos para poder hacer espacio a este nuevo requerimiento; sin embargo, al tener las prioridades con números como en el ejemplo, al nuevo requerimiento

simplemente se pone de prioridad 225 se resuelve el problema sin mover ninguna otra prioridad.

Ya con esta lista acordada, puede entonces el DP reunirse con el equipo y planificar el proyecto.

Características del equipo

- Está formado por el scrum master, SM de ahora en adelante, y cuatro miembros más, que desarrollan: M1, M2, M3 y M4 de ahora en adelante.

- Están acostumbrados a trabajar con la metodología scrum, por lo que ya confían en que sus cálculos de tiempo de trabajo son acertados en un 90 % de los casos.

- Los puntos de historia los miden en horas dedicadas al proyecto, tienen un máximo de 8 horas diarias de trabajo.

- Suelen establecer la duración de los *sprint* basados en lo que les tardará generar incrementos al producto y no con una duración establecida para todos.

- Las reuniones diarias suelen hacerlas a primera hora del día.

- En su forma de trabajo, le llaman producto terminado al incremento que ya está testeado, esta labor la hace siempre M2; por lo tanto, cuando todos terminan su trabajo, este pasa a esperar ser probado por M2 y solamente M2 puede moverlo a terminado.

Sobre el trabajo del proyecto

- El equipo utilizará correos electrónicos para la comunicación interna.

- Utilizarán trello para llevar el control de avance en las actividades, en este caso, el tablero de tareas incluirá las siguientes columnas:

    - Pila del producto: tendrá el listado de todas las tareas del proyecto, ordenadas por prioridad.

    - Por hacer: será el listado de tareas incluidas en el *sprint*.

    - Haciendo: incluye las tareas que actualmente está realizando el equipo, dado que son 4 miembros del equipo, esta columna no puede tener más de 4 tareas al mismo tiempo.

    - Testeando: son las tareas que el equipo terminó y que M2 tiene en proceso de revisión.

    - Terminado: son el listado de tareas que ya M2 revisó y dio por terminadas totalmente.

Tabla IV. **Tablero de tareas a utilizar en el desarrollo de la aplicación de hoteles**

Fuente: elaboración propia.

- Primera reunión, planificación del primer *sprint*

Se reúne el equipo completo y el DP explica de qué se trata el nuevo proyecto y comparte el cuadro que se ha generado de la reunión con el cliente. Ya que todo el equipo entiende la idea del proyecto, la primera tarea es descomponer cada requerimiento en tareas específicas para cumplirlo, por ejemplo:

Tabla V. **Ejemplo de cómo se detallan las tareas de cada requerimiento, desarrollo de la aplicación de hoteles**

Requerimiento	Tareas
Búsqueda de hoteles	1. Diseño de la base de datos (implícita desde el inicio) 2. Trabajo en la base de datos, generación de los queries 3. Diseño del formulario 4. Diseño de la página donde se mostrarán los resultados 5. Programación del formulario 6. Programación de la página de resultados
Inicio	1. Diseño de la página 2. Programación de la página

Fuente: elaboración propia.

De la manera con cada uno de los requerimientos. Ya con las tareas desagregadas, cada miembro del equipo de desarrollo pondera cada tarea con puntos de historia (el tiempo que se tomarían en desarrollarla) y luego hacen un promedio y definen los puntos para cada tarea. Es importante recalcar que si alguna tarea toma más de 16 horas (dos días de trabajo) es necesario descomponerla en subtareas que puedan desarrollarse dentro de ese límite. Así mismo, definen el orden en el que deberán cumplirse, por si alguna tarea

depende de que otra tarea esté terminada para poder iniciarse. Deciden también quién será el encargado de ejecutar cada tarea.

Para el primer *sprint*, se cubrirán los requerimientos de la página de inicio (id 7) y el formulario de nuevo usuario (id 2). Por lo que el cuadro de control queda de la siguiente manera:

Tabla VI. **Ejemplo de planificación de tareas en el desarrollo de la aplicación de hoteles**

Req	idTarea	Tarea	Puntos	Encargado	Depende de
7	1	Diseño de la página	8	M1	0
7	2	Programación de la página	16	M4	1
7	3	Test de la página	2	M2	2
2	4	Diseño de la base de datos	4	M3	0
2	5	Diseño del formulario	6	M1	0
2	6	Programación del formulario	16	M3	5
2	7	Test del formulario	3	M2	6

Fuente: elaboración propia.

De acuerdo al incremento que se ha planificado para el primer *sprint*, la duración de este será de 4 días, por lo que al día 5, ya no será reunión diaria del *sprint* sino se dará la revisión.

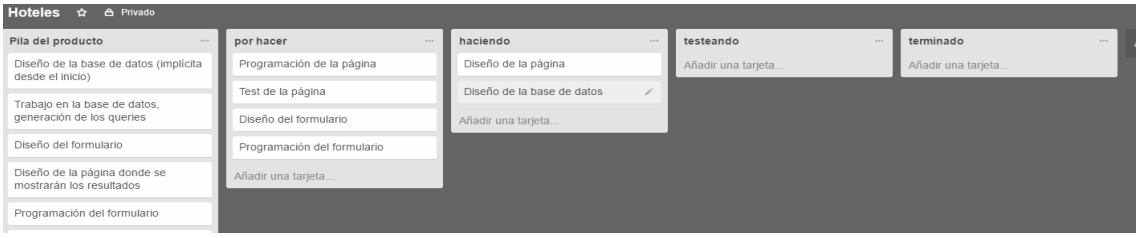
En el cuadro anterior se ve que M1 trabajará dos días en los diseños que le corresponden, y que no es hasta que los diseños están terminados que no puede avanzarse con la programación; por lo tanto, el día uno M1 trabajará en el diseño de la página de inicio y M3 puede iniciar el trabajo con la base de

datos. Mientras, tanto demás en el equipo pueden dedicarse a otro proyecto o algunas otras tareas no relacionadas con este proyecto.

M1 necesita obtener del DP información sobre el diseño, como si se prefiere una u otra paleta de colores específica o si tiene alguna imagen que quiera utilizar.

Al finalizar la reunión de planificación, el tablero queda así:

Figura 10. **Ejemplo de avances en el tablero de tareas en el desarrollo de la aplicación de hoteles, primera reunión diaria**

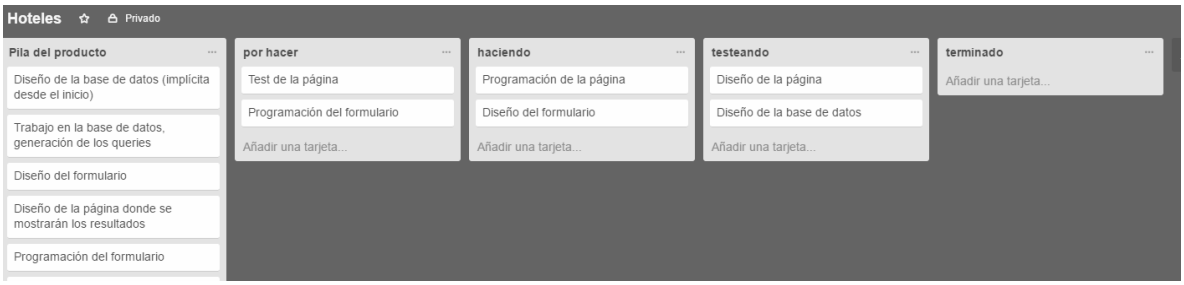


Fuente: elaboración propia.

En la siguiente reunión diaria, cada uno de los miembros del equipo cuenta qué hizo desde la reunión anterior, qué tiene planificado hacer hasta la siguiente reunión, si ha tenido algún inconveniente o necesita alguna información para seguir con su trabajo.

Al finalizar el tablero se verá así:

Figura 11. **Ejemplo de avances en el tablero de tareas en el desarrollo de la aplicación de hoteles, segunda reunión**



Fuente: elaboración propia.

Y el proceso se repite en cada reunión diaria, por los 4 días que dura el *sprint*.

En el día 5 realizan la revisión del *sprint* y, a diferencia de las reuniones diarias anteriores, en esta revisión si está presente el DP, el SM y los 4 desarrolladores, los que se han reunido diariamente.

En la revisión del *sprint*, el equipo presenta el incremento que ha realizado durante el *sprint*. Es importante mencionar que para la página de inicio se muestran hoteles ficticios, pues hasta este punto no se tienen datos de hoteles ni calificaciones de usuarios. El DP felicita al equipo por el trabajo realizado, pero les indica que no le gustan los colores que se han elegido, que en vez de tonalidades verdes preferiría verlo en tonos azules, a pesar de que fue él el que indicó al inicio la paleta de colores a utilizar. Esto no se considera como un problema, simplemente se agrega a la pila del siguiente *sprint* la tarea de cambiar la paleta de colores a utilizar.

Además, indica que la carga de hoteles para la aplicación ya no se realizará utilizando un formulario, como lo indicó al inicio, sino que ha conseguido que de distintas ciudades le envíen un listado de hoteles con la información de cada uno; estos listados serán archivos de extensión xls (de MS Excel), pero los datos pueden venir hasta en 3 formatos distintos, dependiendo las ciudades, por lo que la forma de cargarlos al sitio será subiendo el archivo y que se haga un parseo de los datos para guardarlos en la base de datos. De una vez se presentan los formatos.

Este nuevo requerimiento cambia la pila del producto, por lo que habrá que renegociar el alcance y la duración del proyecto y, si el DP acepta los términos de la negociación, hay que reorganizar las tareas a realizar.

Entonces, el requerimiento inicial con id 3 pasa de:

Tabla VII. **Ejemplo de cambios en los requerimientos, aplicación de hoteles**

id	Requerimiento	Descripción	Prio	Validación
3	Formulario de administración de hoteles	Formulario para ingresar un nuevo hotel al sitio	250	Se completa el formulario y ya se puede visualizar el hotel en el sitio
3	Carga de hoteles por medio de archivo .xls	Se sube un archivo con los datos de varios hoteles y deberá cargarse a la base de datos	250	Se selecciona el archivo con los datos, se presiona el botón de subida y están guardados los hoteles

Fuente: elaboración propia.

En el *sprint* 2, el equipo abarcará los requerimientos 3 y 4 que son los que, por prioridad del DP, son los que siguen en el desarrollo. Pero como se tiene la

solicitud del DP del cambio en la paleta de colores utilizada, su cuadro de tareas para este segundo *sprint* queda de la siguiente manera:

Tabla VIII. **Ejemplo del detalle de tareas para el segundo *sprint*, aplicación de hoteles**

Req	idTarea	Tarea	Puntos	Encargado	Depende de
7	1	Cambio de la paleta de colores de verde a azul	4	M1	0
3	2	Diseño de la base de datos	2	M3	0
3	3	Programación del parser para el formato 1	6	M4	0
3	4	Test formato 1	2	M2	3
3	5	Programación del parser para el formato 2	4	M3	0
3	6	Test formato 2	2	M2	5
3	7	Programación del parser para el formato 3	4	M4	0
3	8	Test formato 3	2	M2	7
3	9	Diseño de la pantalla de subida de archivos	3	M1	0
3	10	Programación de la pantalla de subida de los archivos	4	M3	9
3	11	Test de subida de archivos	1	M2	10
4	12	Diseño de la pantalla de búsqueda y entrega de resultados	5	M1	0
4	13	Programación de la búsqueda de hoteles	8	M4	12

Continuación de la tabla VIII.

4	14	Programación de la pantalla de entrega de resultados	16	M3	13
4	15	Test de la búsqueda y presentación de resultados	3	M2	14

Fuente: elaboración propia.

Según la ponderación de los puntos de historia para cada actividad del segundo *sprint*, se define que este también durará 4 días, pero, a diferencia del primer *sprint*, en este sí se involucran todos los desarrolladores en el proyecto durante más tiempo que en el anterior.

Todo avanza con normalidad en el segundo *sprint*, hasta que, en la reunión del segundo día, M4 indica que el desarrollo del parser para el primer formato (actividad con id 3 de la tabla) le ha tomado más tiempo del planificado y que no lo terminó en el primer día, como se hubiera esperado, que calcula que le tomará al menos 4 horas más para terminarlo, según el avance de hasta ahora.

Esto podría significar el retraso de casi un día de trabajo en la ejecución del *sprint*, pero, en lugar de eso, M1 se ofrece para colaborar con el desarrollo del parser para el formato 3 (actividad con id 7 de la tabla); pero M1 estima que no le tomará 4 horas como lo habían planificado sino le tomará 8, lo cual no atrasa el *sprint*, pues M1 quedaba libre después del segundo día al terminar los diseños requeridos.

3.1.1.1. Análisis del ejemplo

El ejemplo anterior ilustra los roles que intervienen en el proceso de scrum así como las distintas actividades que se realizan al seguir esta metodología; si se observa, no define las herramientas a utilizar para el desarrollo ni estándares de servidores o de bases de datos; scrum es solamente un lineamiento para organizar el trabajo de forma tal que el cliente o su representante (DP) puedan ir observando incrementos en la funcionalidad de la aplicación que se solicita en periodos cortos de tiempo.

También, se puede observar la libertad que permite al momento que surgen cambios en los requerimientos; y permite que el equipo se adapte con mucha facilidad y agilidad a los mismos. También, se trata de demostrar, en la última parte, la multifuncionalidad del equipo, y que aunque M1 se dedica mayormente al diseño gráfico; es también capaz de colaborar en la programación sin que esto represente comprometer la calidad del proyecto.

3.2. Módulo de asignación de cursos

En una universidad que ya cuenta con un sistema de registro de estudiantes inscritos con la información del semestre que el estudiante cursa y un módulo de cursos disponibles por semestre donde se tienen los datos de horarios y los catedráticos que los imparten; se necesita agregar un módulo de estudiantes donde cada estudiante pueda loguearse al sistema y, además, ver su información personal, pueda asignarse cursos para el semestre e ir revisando sus notas a lo largo del curso; esto plantea a la necesidad del módulo donde el catedrático se loguea para ingresar notas de sus cursos.

En este segundo caso de estudio, cambia un poco el escenario, pues se ha pedido apoyo a la escuela de Ingeniería en Ciencias y Sistemas de dicha universidad para desarrollo por lo que el dueño del producto, DP en adelante, es el coordinador del área de desarrollo de sistemas. Cuentan con el apoyo de un catedrático que, por su experiencia, será el scrum master, SM en adelante. El equipo de desarrollo está formado por 3 estudiantes del último año de la carrera de Ingeniería en ciencias y sistemas, en adelante E1, E2 y E3, que solamente pueden dedicarle medio día (4 horas diarias) al proyecto, pues tienen otras obligaciones en seguimiento a sus cursos.

Las primeras condiciones del módulo son:

- Debe mantener el diseño, en cuanto a logos, distribución de contenido y paleta de colores de toda la plataforma que ya existe y se utiliza.
- Existe un plazo de 15 días iniciado el semestre para la asignación de los cursos.

- El estudiante no puede asignarse cursos que tengan horario traslapado ni puede asignarse más de 5 cursos por semestre.

- Cada curso tiene un listado de cursos que deben aprobarse antes; por lo que el estudiante no podrá asignarse cursos sin cumplir con los prerrequisitos.

El equipo de estudiantes designado para este desarrollo, a pesar de sus capacidades técnicas y experiencia en el desarrollo de aplicaciones, no tiene mucha experiencia con scrum, pero se les propone trabajar en *sprints* de 10 días (2 semanas, pues el equipo solamente trabajará de lunes a viernes), esto representa 40 horas de trabajo por persona en cada *sprint*.

El cuadro de requerimientos por parte del DP es el siguiente:

Tabla IX. **Lista de requerimientos, asignación de cursos en línea**

id	Requerimiento	Descripción	Prio	Validación
1	Logueo de estudiantes	Ingreso de usuario y contraseña del estudiante	250	El usuario ingresa sus datos y el sistema le presenta una pantalla con sus datos personales
2	Asignación de cursos	El estudiante escoge los cursos que desea llevar en el semestre y se los asigna	200	El estudiante selecciona los cursos que llevará y al hacer clic en asignar, aparecerán los cursos ya asignados y el estudiante aparecerá en los listados oficiales del curso.
3	Logueo de catedráticos	Ingreso de usuario y contraseña del catedrático	150	El usuario ingresa sus datos y el sistema le presenta una pantalla con los cursos que impartirá en el semestre
4	Ingreso de notas	El catedrático selecciona el curso e ingresa las notas de los estudiantes	100	El catedrático ingresa las notas y que pueden ser consultadas luego por los estudiantes

Fuente: elaboración propia.

Como en el primer caso de estudio, la prioridad que se le da a cada requerimiento se califica con números no correlativos por si en algún momento se decide que existe otro requerimiento que es más prioritario que el 4 pero menos prioritario que el 3 y para no mover las prioridades de todos; solamente se encontraría un número en medio de los dos que ya se tienen en el requerimiento 3 y 4, lo cual, en proyectos que son muy grandes, evitará confusiones en la reorganización de prioridades.

En la reunión de planificación del *sprint*, el equipo comienza con descomponer los requerimientos en pequeñas tareas y valorándolas en términos de horas de trabajo, a lo que se le llama puntos de historia. Al equipo le tomó dos días terminar la planificación, pues los desarrolladores deben atender sus cursos, motivo por el cual no pudieron extenderse más de las cuatro horas que se ha acordado que se dedicarán al proyecto.

Cuando esto sucede, y la reunión de planificación se extiende más allá de lo que se ha planificado, lo mejor es quedarse hasta terminar, con pequeñas pausas a manera de que el equipo no lo sienta más pesado. Ahora bien, si no es posible extenderse, como en este caso de estudio, se recomienda programar otra reunión lo antes posible.

Puede ocurrir que, dependiendo del proyecto y los requerimientos que se solicitan, surjan tareas por realizar que no pertenecen precisamente a una historia de usuario (requerimiento), pero que son necesarias dentro del proceso; a este tipo de tareas se les conoce como *spikes*, generalmente son de tipo investigación y pueden ser técnicos para evaluar la viabilidad de un requerimiento o funcionales para entenderlos mejor. En este caso en particular, se ha detectado la necesidad de documentarse en el funcionamiento de la aplicación web de la universidad, pues se necesita agregarle los módulos solicitados; por lo tanto el listado de tareas a realizar queda de la siguiente forma:

Tabla X. **Desglose de tareas para cada requerimiento, asignación de cursos en línea**

idReq	idTarea	Tarea	Puntos
	1	Leer y documentarse de cómo está desarrollado el sistema actual para poder saber cómo amarrar el nuevo módulo.	36
	2	Diseño de la base de datos.	8
	3	Pruebas de integración con el sistema de la universidad.	16
1	4	Diseño de las pantallas de logueo (será el mismo diseño para las dos).	2
1	5	Desarrollo del *login* de usuarios.	4
1	6	Diseño de la página de inicio de los estudiantes.	2
1	7	Desarrollo de la página de inicio de los estudiantes.	8
1	8	Pruebas del *login* y página de inicio de estudiantes.	4
3	9	Desarrollo del *login* de catedráticos.	4
3	10	Diseño de la página de inicio de los catedráticos.	2
3	11	Desarrollo de la página de inicio de los catedráticos.	8
3	12	Pruebas del *login* y página de inicio de catedráticos.	4
2	13	Listar cursos del semestre que cursa el estudiante.	4
2	14	Validar los horarios de los cursos que el estudiante se desee asignar.	8
2	15	Validar los prerrequisitos de los cursos que el estudiante se desee asignar.	8
2	16	Función de asignación de cursos.	8
2	17	Desarrollo de la pantalla de asignación de cursos.	4
4	18	Listar los estudiantes que estén asignados a un curso.	4
4	19	Obtener los cursos que imparte el catedrático.	4
4	20	Desarrollo de la pantalla de ingreso de notas de un curso.	12

Fuente: elaboración propia.

Se entiende, entonces, que cada integrante del equipo dedicará 3 días en la investigación del funcionamiento de la plataforma de la universidad; luego, otros 2 días en realizar pruebas de integración con la misma.

Considerando entonces las tareas *spike* que surgieron y el tiempo de duración establecido para los *sprints*, se decide que solamente las primeras 15 tareas se incluyen en el primer *sprint*, con la asignación de responsables de la siguiente forma:

Tabla XI. **Asignación de tareas a los miembros del equipo, asignación de cursos en línea**

idReq	idTarea	Tarea	Puntos	Encargado
	1	Leer y documentarse de cómo está desarrollado el sistema actual para poder saber cómo amarrar el nuevo módulo.	36	E1, E2, E3
	2	Diseño de la base de datos.	8	E2
	3	Pruebas de integración con los otros módulos.	16	E1, E3
1	4	Diseño de las pantallas de logueo (será el mismo diseño para las dos).	2	E3
1	5	Desarrollo del *login* de usuarios.	4	E1
1	6	Diseño de la página de inicio de los estudiantes.	2	E3
1	7	Desarrollo de la página de inicio de los estudiantes.	8	E1
1	8	Pruebas del *login* y página de inicio de estudiantes.	4	E3
3	9	Desarrollo del *login* de catedráticos.	4	E2

Continuación de la tabla XI.

3	10	Diseño de la página de inicio de los catedráticos.	2	E3
3	11	Desarrollo de la página de inicio de los catedráticos.	8	E2
3	12	Pruebas del *login* y página de inicio de catedráticos.	4	E3
2	13	Listar cursos del semestre que cursa el estudiante.	4	E3
2	14	Validar los horarios de los cursos que el estudiante se desee asignar.	8	E2
2	15	Validar los prerrequisitos de los cursos que el estudiante se desee asignar.	8	E1

Fuente: elaboración propia.

La universidad ha provisto al equipo de un espacio donde pueden realizar el trabajo: Un salón con el equipo necesario y un área con una pequeña mesa para reuniones, donde se ha dado la planificación y donde se realizarán las reuniones diarias que han quedado programadas para las 7:00 horas de lunes a viernes. El equipo preparó una de las paredes del área de reuniones y acomodó el tablero de tareas con cartulinas y papeles con cinta adhesiva para las tareas, solamente utilizarán cuatro columnas:

- Pila del producto: con el listado de todas las tareas a realizar en el proyecto completo.
- Por hacer: con la pila del *sprint*.
- Haciendo: con las tareas en las que actualmente se dedican.
- Finalizado: con las tareas ya finalizadas.

Cabe aclarar que se ha decidido que cada miembro del equipo es el responsable de probar su trabajo de tal modo que cuando se mueve a la columna de finalizado es porque está seguro de que funciona como debería funcionar. Es responsabilidad de cada miembro mover los papelitos de las tareas que tiene a su cargo para que de esta manera el tablero esté actualizado todo el tiempo.

Adicional al tablero, el SM ha preparado un diagrama para ir controlando el avance del trabajo en relación al tiempo que ha pasado un *burndown* que ha colocado junto al tablero de actividades.

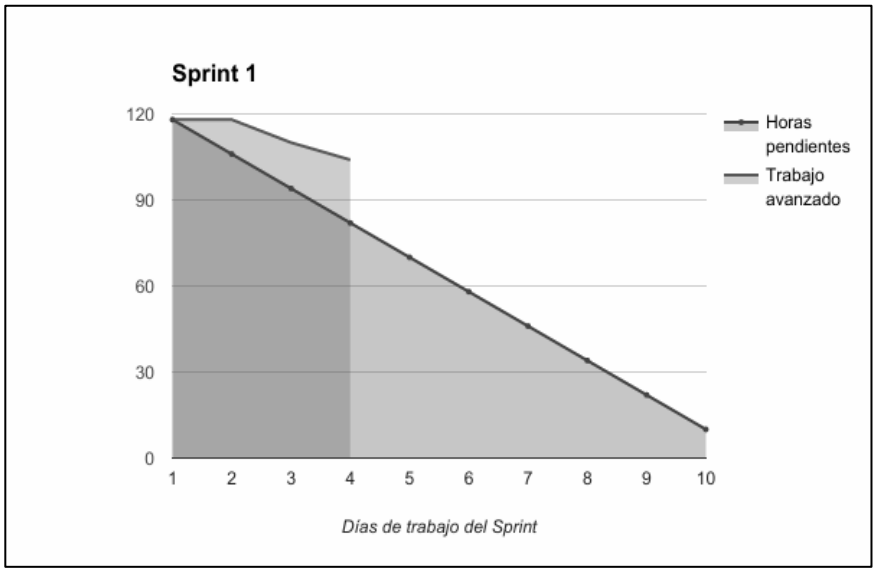
Al terminar la reunión de planificación, el SM empieza por solicitar a donde corresponde la documentación técnica de la plataforma de la universidad para poder compartirla con el equipo y puedan entenderla y, de esta manera, poder trabajar con ella.

En la primera reunión diaria, el SM le informa al equipo que no ha recibido la información técnica solicitada que significa que no se puede avanzar en la planificación de actividades en el orden acordado. En estos casos, el equipo debe acordar avanzar en las actividades que sí pueden hacerlo, por lo que los tres desarrolladores trabajan en el diseño de la base de datos del módulo esperando que para la siguiente reunión ya se cuente con la información necesaria para su análisis.

La misma situación, para la segunda reunión y para la tercera, sigue sin recibir la información y ya se tiene el diseño aprobado de la base de datos, como en la primera reunión; se debe tratar de reorganizar para avanzar en lo que se pueda por evitar retrasos que afecten la finalización del *sprint*. El equipo, entonces, comienza con los diseños de las pantallas para tratar de no atrasar más el avance.

Entonces, ya han pasado tres días (12 horas por miembro del equipo) y el trabajo que han avanzado es mínimo ya que solamente han realizado el trabajo de la base de datos (8 horas) y han avanzado las 6 horas planificadas para el diseño de todas las pantallas. La situación es crítica y se refleja en el *burndown*:

Figura 12. ***Burndown* a la cuarta reunión diaria, asignación de cursos**



Fuente: elaboración propia.

En estos casos, el diagrama de *burndown* cobra mayor importancia, pues evidencia el retraso y el equipo se da cuenta que, de seguir en las mismas condiciones, no cumplirán con el avance planificado en el *sprint*.

Afortunadamente, ya para la cuarta reunión diaria, el SM distribuye la información técnica de la aplicación, misma que recibió al finalizar la tarde del día anterior. Para este punto ya llevan casi 20 horas de retraso, por lo que empiezan a dudar que logren terminar completamente lo programado.

En la reunión del 8vo día, el equipo ha avanzado en conocer la plataforma y ha realizado las pruebas de integración. Pero solamente les quedan 3 días de trabajo que equivalen a 36 horas hombre de trabajo. En casos de retrasos tan significativos en el *sprint*, se debe refinar la pila nuevamente y, de ser necesario, eliminar historias (actividades) del mismo, ya que la duración del *sprint*, una vez decidida, no es variable. El equipo, entonces, abarcará solamente hasta la tarea con id 12, es decir, solamente desarrollarán las páginas del *login* y de inicio tanto de estudiantes como de catedráticos, y presentarán hasta allí el incremento del *sprint*, dejando fuera las siguientes tareas:

Tabla XII. **Tareas pendientes del *sprint* uno, asignación de cursos en línea**

2	13	Listar cursos del semestre que cursa el estudiante	4	E3
2	14	Validar los horarios de los cursos que el estudiante se desee asignar	8	E2
2	15	Validar los prerrequisitos de los cursos que el estudiante se desee asignar	8	E1

Fuente: elaboración propia.

En la revisión del primer *sprint*, el equipo presenta el incremento del proyecto y explica los motivos de los atrasos, mismos que, aunque fueron por motivos totalmente ajenos, igual impactan en el proceso de desarrollo.

Esa misma tarde, después de clases, el SM se reúne con los tres miembros del equipo de desarrollo para platicar de lo bueno y lo malo del *sprint,*

en la cafetería. Es lo que se conoce como retrospectiva. En la reunión surgen comentarios como:

- "Que mal que no nos entregaron la información a tiempo, tal vez si la hubiéramos pedido antes de iniciar, desde que se nos habló del proyecto…"
- "Lo bueno es que nos apoyamos todos en lo que se podía hacer, trabajamos juntos en los diseños tanto de la base de datos como de las pantallas…"

Y aunque nadie culpa del atraso a ningún compañero, si están conscientes que deben ser un poco más proactivos en el futuro.

El equipo se reúne para la planificación del segundo *sprint*, se incluyen primero las tareas que quedaron pendientes del *sprint* anterior y se agregan las demás que quepan, según lo establecido en la duración del *sprint* que prácticamente incluye todo el proyecto, a este proceso se le conoce como refinamiento de la pila. A continuación, se presenta el listado de las actividades y los responsables de cada una:

Tabla XIII. **Asignación de tareas a los miembros del equipo en el sprint dos, asignación de cursos en línea**

2	13	Listar cursos del semestre que cursa el estudiante.	4	E2
2	14	Validar los horarios de los cursos que el estudiante se desee asignar.	8	E1
2	15	Validar los prerrequisitos de los cursos que el estudiante se desee asignar.	8	E3
2	16	Función de asignación de cursos.	8	E2

Continuación de la tabla XIII.

2	17	Desarrollo de la pantalla de asignación de cursos.	8	E3
4	18	Listar los estudiantes que estén asignados a un curso.	4	E1
4	19	Obtener los cursos que imparte el catedrático.	4	E2
4	20	Desarrollo de la pantalla de ingreso de notas de un curso.	12	E1
2	21	Pruebas de integración con la plataforma.	36	E1, E2, E3

Fuente: elaboración propia.

El trabajo del equipo se desarrolla de forma normal sin mayores problemas a partir de este punto.

3.2.1.1. Análisis del ejemplo

En este segundo caso de estudio se ejemplifica un atraso en el desarrollo de un proyecto que, ya sea por razones propias o ajenas al equipo como es en este caso, igual se debe lidiar con el atraso.

Lo importante es resaltar que la metodología utilizada correctamente permite que el equipo pueda prever el atraso y planificar cómo lidiará con este. El manejo de la situación dependerá de los motivos del atraso, lo importante es que el atraso pueda verse con tiempo suficiente para lidiar con este.

Supóngase que no todas las actividades en la pila del producto hubieran dependido de la información técnica de la plataforma sino solamente las del primer *sprint*, se hubiera podido manejar el retraso en recibir dicha información cambiando la pila del *sprint* y reorganizar toda la pila del proyecto de forma que se pudiera avanzar con algo en lo que se esperaba la información.

Ahora bien, si el atraso se debe a la lentitud del equipo o que alguno de los miembros tuviese razones para ausentarse del proyecto, igual puede refinarse la pila o hacer cambios de asignación de las tareas, a manera de compensar para tratar de no afectar el ritmo del trabajo. Incluso se pudiera evaluar el agregar más personas al equipo.

El ejemplo, también, muestra la importancia de que el equipo sea multidisciplinario, es decir, los tres integrantes son capaces de diseñar, programar y, además, hacer sus propias pruebas, lo que puede ayudar a la agilidad del trabajo.

Se muestra también lo útil que puede ser la reunión para hacer retrospectiva del *sprint*, pues en un ambiente más relajado y agradable para el equipo pueden evaluar lo bueno, lo malo y las cosas que pudieran mejorarse en el proceso. Lo importante es que el SM dirija la reunión de tal forma que no se convierta en un juicio tratando de encontrar culpables de los problemas, sino que sea un momento de crecimiento y aprendizaje para todos.

3.3. Sistema de gestión documental

En una entidad de gobierno, surge la necesidad de gestionar de forma más sistematizada la documentación que se recibe en las 3 sedes por lo que deciden contratar desarrolladores externos para realizar el proyecto. La entidad ya cuenta con una estructura de red que permite la interconexión de las máquinas de todos sus empleados a un servidor central por medio del cual ya manejan el correo y alguna que otra aplicación para realizar sus labores. Es por eso que piden que la aplicación para la gestión documental sea basada en la web.

El equipo de desarrolladores del proyecto es un grupo de 2 desarrolladores, D1 y D2 y un coordinador que generalmente actúa como scrum master, SM. Tienen poco más de 3 años trabajando juntos, lo que indica que han logrado combinar sus capacidades de tal forma que les ha permitido desarrollar más de 8 proyectos juntos. Utilizan scrum desde hace poco más de un año desarrollando en este tiempo 4 proyectos de éxito. Acostumbran hacer *sprints* de 1 semana de duración (5 días) con jornadas laborales de 7 horas diarias.

De parte de la entidad gubernamental, el director del área de tecnología es la persona que ha recabado los requerimientos de los usuarios y que, en este caso, actuará como dueño del proyecto, DP, pues, del lado del cliente, será la persona encargada de la toma de decisiones. Ya se ha reunido varias veces con las personas involucradas en el proceso de gestión documental y ha preparado una lista de requerimientos que se detalla a continuación:

- Se desea un sistema que le permita a la institución llevar el control de todos los documentos que llegan a cualquiera de las sedes. Que lleve un control de usuarios por sede.

- El sistema debe permitir registrar la correspondencia que se recibe en la institución, identificando también tanto al remitente como a la persona que lleva el documento a la institución. Este registro se hace en el área de recepción de documentos.

- La persona que recibe el documento, además de registrar los datos anteriores, debe escanear el documento y asignar dentro del sistema quién debe recibirlo.

- Al estar ingresado, el sistema debe indicarle a la persona destinataria que tiene un documento nuevo y debe permitirle verlo y descargarlo a su computadora, anotando la fecha y hora en que lo recibe.

- Un mismo documento recibido puede ser entregado a varias personas dentro de la institución.

- Si un documento llega a una unidad, pero por cuestiones de trabajo debe compartirse con otra, la persona que lo recibe inicialmente será el dueño del documento, pero podrá compartirlo para que otros usuarios tengan también acceso.

- Si por error el documento es asignado a una persona, esta podrá devolverlo al área de recepción y allí se encargarán de reasignarlo a donde corresponde.

- Adicionalmente, en el sistema se debe registrar quién recibe el documento físico para su archivo, registrando la fecha y hora de entrega.

- El sistema debe permitir búsquedas de documentos por fecha de recepción, por remitente o por destinatario.

- El sistema debe ser capaz de guardar un registro de acciones de los usuarios, de tal forma que pueda ser trazable el recorrido de un documento desde el momento que ingresa, guardando siempre qué usuario realiza cada acción.

Luego de la primera reunión del SM con el DP, acuerdan el desarrollo de la aplicación para un tiempo máximo de 2 meses, con un cuadro de historias de usuario de la siguiente forma:

Tabla XIV. **Historias de usuario para el sistema de gestión documental**

id	Historia	prio	Validación
1	El usuario de tipo receptor podrá registrar documentos que llegan a cualquiera de las sedes de la institución y adjuntar el documento escaneado.	500	Se llena el formulario de nuevo ingreso y se puede ver un listado de los documentos ingresados por día.
2	El usuario de tipo destinatario podrá ver los documentos que han ingresado para un usuario específico.	450	El usuario ingresa al sistema y se le muestra una alerta de los documentos nuevos y un listado de todos los que ha recibido.
3	Todos los tipos de usuario podrán hacer búsquedas de documentos.	400	Se ingresan los parámetros de búsqueda y lista los coincidentes con el registro de la fecha de entrada al sistema.
4	El usuario de tipo destinatario podrá regresar los documentos que no le corresponden y que se le han asignado por error.	350	Con un botón el documento regresa al módulo de registro con bandera de que iba mal asignado y se podrá reasignar.
5	El usuario destinatario podrá descargar el documento.	300	El usuario puede descargar el documento digital a su computador.
6	El usuario de tipo destinatario podrá recibir un documento físico.	250	Registrar con un clic cuando ya se ha recibido el documento físico y el sistema debe indicar que dicho documento ha sido entregado.
7	El usuario de tipo destinatario podrá archivar los documentos que recibe.	200	El usuario puede archivarlo y el documento desaparece del listado de inicio.
8	El usuario de tipo destinatario podrá compartir documentos con otros destinatarios.	150	El usuario selecciona un documento y hace clic en compartir, selecciona con quién desea compartirlo y todos esos usuarios tendrán acceso al documento, pero solo para verlo, no para archivarlo.

Fuente: elaboración propia.

Como en los casos anteriores, la prioridad que se le da a cada historia se califica con números no correlativos por si en algún momento se decide que existe otro requerimiento que es más prioritario que el 4 pero menos prioritario que el 3, y para no mover las prioridades de todos, solamente se encontraría un número en medio de los dos que ya se tienen en el requerimiento 3 y 4, lo cual, en proyectos que son muy grandes, evitará confusiones en la reorganización de prioridades.

Ya con la pila del producto preparada por el DP, es momento de la planificación del *sprint*, se reúnen entonces los dos desarrolladores, el SM y el DP para ponderar actividades por hacer y decidir el incremento del *sprint* uno.

Luego de varias horas de reunión, acuerdan que la reunión diaria tendrá lugar todos los días a las 9:00 horas, el listado de tareas a realizar ya ponderadas y asignadas a un responsable, queda de la siguiente forma:

Tabla XV. **Pila del producto del sistema de gestión documental**

Hist.	corr	Tarea	Pond	Resp
	1	Diseño de la base de datos del producto..	7	
	2	Diseño de la aplicación.	21	
	3	Programación del login.	7	
1	4	Programación del formulario de ingreso de documentos.	10	
1	5	Pruebas de ingreso de documentos.	3	
2	6	Programación de la página de inicio del destinatario, que muestra los documentos que se han recibido.	7	
3	7	Programación de los formularios de búsqueda (por fechas, por nombre del documento, por destinatario, por remitente).	7	

Continuación de la tabla XV.

3	8	Programación de las funciones de búsqueda y presentación de resultados.	14	
3	9	Programación de la página de resultados de búsquedas con paginación.	7	
3	10	Pruebas de búsquedas.	3	
4	11	Función de regresar documentos.	5	
4	12	Función de reasignar documentos.	5	
5	13	Función de descarga de documentos.	5	
6	14	Función de marcar como recibido el documento físico.	5	
7	15	Función de archivar documentos.	5	
4,5,6,7	16	Pruebas de funcionalidad usuario tipo destinatario.	14	
3	17	Programación de búsqueda y presentación de resultados en los documentos archivados.	7	
8	18	Función de compartir documentos.	5	
8	19	Pruebas de compartir documentos.	2	
1	20	Notificaciones de documentos nuevos.	4	
4	21	Notificaciones de documentos que han sido devueltos para reasignación.	4	
1,4	22	Pruebas de notificaciones.	4	

Fuente: elaboración propia.

Como se ha establecido que el equipo se dedicará al desarrollo 7 horas diarias y que el *sprint* durará una semana (5 días hábiles), se define que se cuenta con 35 horas semanales por cada desarrollador; ahora bien, en la planificación del *sprint*, D2 ha comentado que le han autorizado una licencia de tres meses para sus estudios, por lo que solamente se podrá dedicar 4 horas diarias al trabajo, lo que hace un total de 20 horas, más las 35 que le dedicará D1, son 55 horas hombre disponibles para el cada *sprint*. Considerando lo

anterior, se seleccionan las siguientes actividades para generar el incremento en el primer *sprint*:

Tabla XVI. **Pila del primer *sprint* del sistema de gestión documental**

Hist.	corr	Tarea	Pond	Resp
	1	Diseño de la base de datos del producto.	7	D2
	2	Diseño de la aplicación.	21	D1
	3	Programación del *login.*	7	D2
1	4	Programación del formulario de ingreso de documentos.	10	D1
1	5	Pruebas de ingreso de documentos.	3	D2

Fuente: elaboración propia.

Ahora bien, sé que, aunque la suma de las horas de las actividades asignadas a cada desarrollador es menor al número de horas que cada uno de ellos puede dedicarse al proyecto (D1 suma 31 de 35 mientras D2 suma 17 de 20 horas semanales), la actividad 6 no se ha agregado al *sprint*. La razón de no hacerlo es que la actividad 6 tiene una ponderación de 7 horas, por lo que no se le puede asignar a ninguno de los dos desarrolladores porque no les daría tiempo de terminarla (D1 solamente tiene 4 horas libres y D2 tiene 3); y en scrum, una tarea medio terminada no está terminada, por lo que es preferible no incluirla para asegurar que todas las tareas que se incluyen en el *sprint* se completarán al 100 %.

También, se puede observar que, a diferencia de los casos de estudio anteriores, en este caso se ha establecido el diseño tanto de la base de datos como de la aplicación como una sola gran tarea esto solamente obedece a la forma de trabajo del equipo que, en este caso, prefiere comenzar teniendo todo

el diseño desde el inicio para luego no perder tiempo en diseñar por módulos la aplicación como lo pueden preferir otros equipos de trabajo, no hay reglas para esto.

Se preguntarán por qué el *login* y los diseños no corresponden a ninguna historia la explicación es porque en verdad, aunque son tareas muy necesarias para el proyecto, no pertenecen a una historia específica se asume que para las historias ya está contemplado que utilizan una base de datos, se tendrá un diseño y dado que la aplicación conlleva control y tipos de usuario, es necesario tener un punto de acceso donde cada usuario se identifique para ingresar.

El *sprint* transcurre con relativa normalidad, el equipo se reúne diariamente para ir revisando los avances, pero no surge ninguna situación fuera de lo normal.

No es sino hasta la revisión de este primer *sprint* que el equipo detecta un error. El incremento funciona muy bien, los usuarios pueden loguearse, ingresan documentos y los mismos aparecen en un listado muy sencillo solamente para mostrar que el ingreso funciona. Lo que ninguno de los desarrolladores consideró, por el más inocente de los descuidos, fue el detalle de que debían existir distintos tipos de usuario: el tipo encargado de ingresar los documentos y el tipo de usuario que es destinatario de los documentos, cada uno con funciones muy distintas dentro del sistema.

En este caso, el incremento funciona, pero quedan pendientes tareas que debieron considerarse desde el principio. Por lo que no incurren en ningún atraso como tal pero si se ven en la necesidad de refinar lo que se incluirá en el siguiente *sprint*.

El DP propone algunos cambios en el diseño de la aplicación, pero queda satisfecho con el incremento que se le ha presentado.

Luego de la revisión del *sprint*, el SM y los desarrolladores se reúnen para hacer una pequeña retrospectiva lo que ha sucedido durante el *sprint*. Cada uno de los desarrolladores se expresa en relación a cómo han ido pasando las cosas durante el *sprint*. D2 les comenta, por ejemplo, que considera que, dado que solamente está trabajando en el proyecto durante 4 horas diarias, la reunión a las 9:00 horas le corta su mañana y siente que le quita más tiempo, por lo que les propone moverla para las 8:00 horas, para arrancar el día con esto y luego poder dedicarse de corrido a sus tareas.

En la reunión de planificación para el segundo *sprint,* el SM hace oficial la nueva hora de las reuniones diarias pues todo el equipo estuvo de acuerdo con moverla. El DP les comenta también que ha estado pensando y que necesita que el sistema guarde registró de quién ingresa cada documento, así como de las fechas y usuarios que regresan los documentos para que sean reasignados. Además, necesitan un usuario como supervisor para que esté monitoreando el ir y venir de los documentos para establecer que tantos errores cometen los usuarios; esta nueva historia tiene una prioridad de 325 para el DP por lo que debe insertarse entre las historias 4 y 5 del listado original. Queda lo insertado así:

Tabla XVII. **Actividades agregadas a la pila del producto del sistema de gestión documental**

Hist.	corr	Tarea	Pond	Resp
4.5	13	Programación de la página de inicio del supervisor.	4	
4.5	14	Programación de funciones de supervisión, cuadros de cuántos documentos ingresa cada usuario, cuántos son regresados por error en la asignación, por rangos de fecha y filtros de usuario.	7	

Fuente: elaboración propia.

De igual forma, la nueva historia no se considera para abordarla en el segundo *sprint*, del cual la pila queda de la siguiente forma:

Tabla XVIII. **Pila del segundo *sprint* del sistema de gestión documental**

Hist.	corr	Tarea	Pond	Resp
	*	Agregar tipos de usuario y las validaciones de ingresos y permisos.	4	D2
2	6	Programación de la página de inicio del destinatario que muestra los documentos que se han recibido.	7	D2
3	7	Programación de los formularios de búsqueda (por fechas, por nombre del documento, por destinatario, por remitente).	7	D1
3	8	Programación de las funciones de búsqueda y presentación de resultados.	14	D1
3	9	Programación de la página de resultados de búsquedas, con paginación.	7	D1
3	10	Pruebas de búsquedas.	3	D1
4	11	Función de regresar documentos.	5	D2
4	12	Función de reasignar documentos.	5	D1

Fuente: elaboración propia.

Si se dan cuenta, sumando las horas de ponderación de cada tarea, D1 tiene una hora más en tareas asignadas de las que tiene disponibles en la

semana, pero al ser un trabajador muy proactivo se compromete a terminar el trabajo a tiempo. Aunque no es lo normal con esta metodología, pues arriesga el hecho de terminar el incremento completo, deciden aprobarlo.

Para el día 1, se reúnen a primera hora, D1 inicia con la programación de los formularios de búsqueda mientras D1 agrega los controles de tipos de usuarios y los accesos que cada tipo tiene a las opciones del sistema.

En el día 2, igual, todo normal en la reunión de la mañana, D2 ha terminado el trabajo de los tipos de usuario y comienza con la actividad 6: la página de inicio de los destinatarios. D1 comienza la programación de las funciones de búsqueda.

En la reunión del día 3, D2 notifica que debe ausentarse los días 4 y 5 por razones personales, por lo que dejará la tarea 11 sin trabajar. D1 le explica al equipo que va un poco complicado con sus tiempos también y que no está seguro de poder terminar lo que se le asignó. El SM puede ver en el *burndown* que definitivamente no completarán el incremento planificado con la tendencia que llevan, por lo que lo mejor en este caso es refinar la pila y dejar fuera las actividades 11 y 12. Con lo que se aseguran presentar un incremento hasta las búsquedas de documentos, es menos de lo acordado, pero al menos será funcional y potencialmente listo para salir a producción.

Así termina el *sprint*; el día de la revisión, el DP se muestra molesto por los inconvenientes en el desarrollo del sistema pero termina por aceptar lo sucedido, pues el SM le comenta que es preferible no abarcar algunas tareas en el *sprint* a comprometer la calidad del producto, lo cual pasaría si deciden trabajar más rápido solo por cumplir plazos. Además, le recuerda que el compromiso era terminar en 2 meses pero que, si continuaban con el ritmo de trabajo, podrían terminar mucho antes.

3.3.1. Análisis del ejemplo

En el caso mostrado, se ejemplifica cómo las condiciones del equipo pueden verse afectadas por distintas situaciones, en el caso de D1 que aparte de su licencia para estudios tuvo algunos otros problemas que afectaron su rendimiento en el trabajo que evidencian que los problemas y atrasos no siempre vienen del lado del cliente y sus nuevos requerimientos, a veces el equipo se ve comprometido también. Afortunadamente scrum, en su condición de ser una metodología ágil, permite la rápida reacción y adaptación a estas situaciones; siempre y cuando se cumplan los procesos, se cumpla con el compromiso que debe tener cada participante, no solamente el equipo de desarrollo sino también el dueño del producto quien debe mantenerse cerca y en estrecha relación con el scrum manager y los desarrolladores, debe integrarse al equipo.

Es importante resaltar la importancia de las reuniones diarias, son la base de la flexibilidad que ofrece la metodología, lo que permite que el equipo reaccione o mejor aún, prevea los problemas y pueda manejarlos con cierta antelación. De allí que deben realizarse en horario conveniente para todos los involucrados, asegurando así su presencia.

El ejemplo hace énfasis en el hecho de que un producto medio terminado no está realmente terminado, y así deberá considerarse siempre. Habla también del compromiso que los desarrolladores deben mantener con la calidad del trabajo que realizan, tanto en lo que se ve en diseños agradables y profesionales, como en lo que el cliente no vé el código. Generar baja calidad de código por minimizar los tiempos de desarrollo solamente genera una debilidad en el producto final, por lo tanto, es preferible refinar las pilas del *sprint* antes que comprometer la calidad del trabajo.

4. ENCUESTA A ESTUDIANTES Y EGRESADOS DE INGENIERÍA EN CIENCIAS Y SISTEMAS DE LA USAC

4.1. Diseño de la encuesta

El objetivo de realizar una encuesta es conocer qué tanto las personas que se dedican al desarrollo web conocen la metodología scrum y su aplicación en este tipo de desarrollos. Saber si es parte de los temas abordados en los cursos que se imparten en la Escuela de Ciencias y Sistemas de la Universidad de San Carlos de Guatemala o si la persona ha tenido que aprenderla en otros ambientes como el laboral o algún otro.

Además de conocer la metodología, interesa saber si ha trabajado con ella o si está consciente de los beneficios que puede obtener aplicándola. Y más aún, si sabiendo sus beneficios, estaría dispuesto a utilizarla.

La encuesta consta de una serie de 10 preguntas con respuestas cerradas, es decir, de selección, para facilitar el análisis de los resultados. Las preguntas y sus opciones de respuesta son:

- ¿Cuál es su relación actual con la Escuela de Ciencias y Sistemas? (obligatoria)
    - Estudiante
    - Egresado
    - Graduado

Aunque muchas personas utilizan indistintamente los términos egresado y graduado, no significan lo mismo.

- Graduado: es la persona que culminó sus estudios y es acreditado con un documento que certifica que culminó sus estudios: el título.
- Egresado: es la persona que culminó sus estudios pero que, por alguna razón ajena a la universidad, no se ha graduado, es decir, no tiene un título.

- ¿Se dedica usted al desarrollo de *software*? (obligatoria)
  - Sí
  - No

- ¿Qué tipo de metodología prefiere seguir al desarrollar un producto de *software*?
  - Ágiles
  - Formales
  - No aplica

- ¿Conoce la metodología scrum? (obligatoria)
  - No la conozco
  - He oído/leído algo
  - Sí la conozco y la he implementado

- ¿Cómo conoció la metodología scrum? (obligatoria)
  - En la universidad
  - En el trabajo

- En libros/Internet
- Por otras personas
- No he oído de scrum

- ¿Cree que utilizar scrum puede ayudar a mejorar el proceso de desarrollo de *software*? (obligatoria)
  - Sí
  - No
  - No sé, no la he utilizado

- ¿Cuál ha sido el mayor reto que ha enfrentado al implementar scrum? (puede escoger uno o más)
  - Es difícil manejar la relación con el dueño del producto
  - Mucha dificultad para hacer tiempo para el scrum diario
  - Es muy complicado calcular qué historias incluir en el *sprint*
  - Nunca la he utilizado

- Siendo 1: muy fácil y 5: muy difícil, ¿qué tan difícil considera que es adoptar la metodología scrum para el desarrollo?
  - 1
  - 2
  - 3
  - 4
  - 5

- ¿Estaría dispuesto a cambiar su actual metodología de trabajo por una que le permita ir generando de forma ágil e incremental, pequeñas partes funcionales, dándole la posibilidad de verificar los requerimientos y la calidad de su producto?
    - Sí
    - No

- ¿Estaría interesado en conocer más acerca de scrum?
    - Sí
    - No

4.2. Metodología

La encuesta se realizó utilizando la aplicación Google Forms y ha circulado a través de redes sociales de los grupos de estudiantes de la Escuela de Ciencias y Sistemas de la USAC, con el apoyo del movimiento COECYS (Congreso de Estudiantes de Ciencias y Sistemas).

Ha estado disponible durante tres semanas y tuvo respuesta de 80 personas.

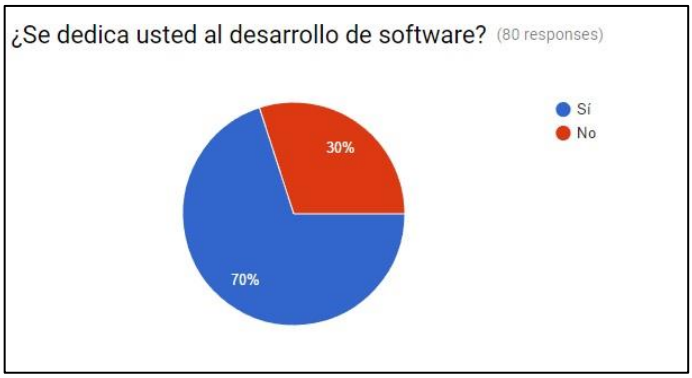
4.3. Resultados

Figura 13. **Encuesta 1: conocer el nivel académico de los encuestados**

Fuente: *Metodología de desarrollo scrum*. https://docs.google.com/forms/d/1zw4u_dqwRduodGzgpeZbpZuNf4rFSgN9zSnkiM0xRBE/edit. Consulta: 18 de enero de 2017.

La primera pregunta tiene como objetivo conocer el nivel académico de los encuestados. De los 80 encuestados: 51 (63,7 %) son estudiantes y el resto ha concluido sus estudios.

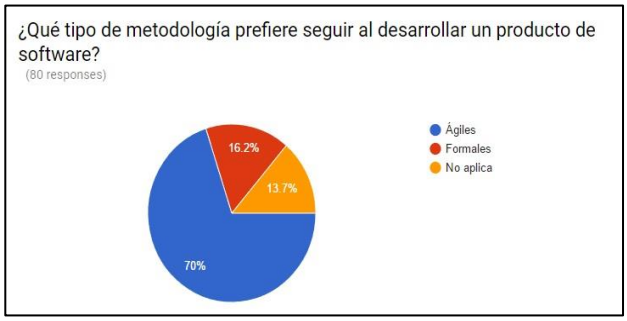
Figura 14. **Encuesta 2: ¿se dedica al desarrollo de *software*?**



Fuente: *Metodología de desarrollo scrum*. https://docs.google.com/forms/d/1zw4u_dqwRduodGzgpeZbpZuNf4rFSgN9zSnkiM0xRBE/edit. Consulta: 18 de enero de 2017.

De los 80 encuestados, un 70 % se dedica al desarrollo de software: que un buen grupo para evaluar la metodología scrum en el desarrollo de aplicaciones.

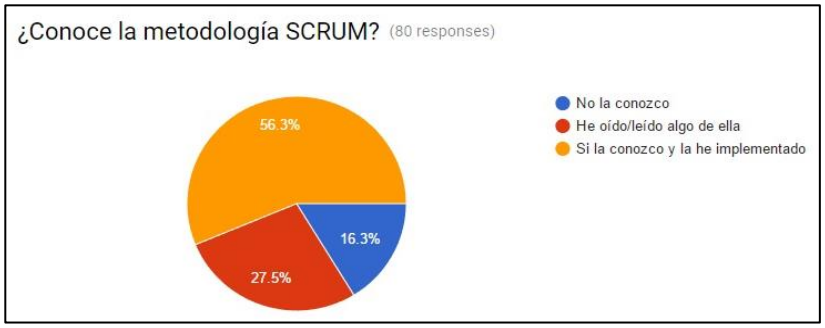
Figura 15. **Encuesta 3: ¿qué tipo de metodología prefieren seguir?**



Fuente: *Metodología de desarrollo scrum*. https://docs.google.com/forms/d/1zw4u_dqwRduodGzgpeZbpZuNf4rFSgN9zSnkiM0xRBE/edit. Consulta: 18 de enero de 2017.

El 70 % de los encuestados prefiere las metodologías ágiles para el desarrollo de aplicaciones. scrum es definida como tal.

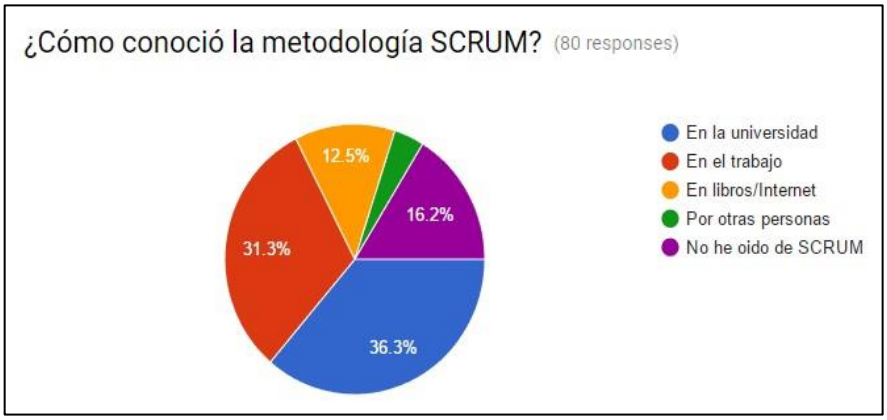
Figura 16. **Encuesta 4: ¿conoce la metodología scrum?**



Fuente: *Metodología de desarrollo scrum*. https://docs.google.com/forms/d/1zw4u_dqwRduodGzgpeZbpZuNf4rFSgN9zSnkiM0xRBE/edit. Consulta: 18 de enero de 2017.

El 56,3 % de los encuestados conoce y ha implementado scrum, solamente el 16,3 % nunca ha oído ni conoce de la metodología.

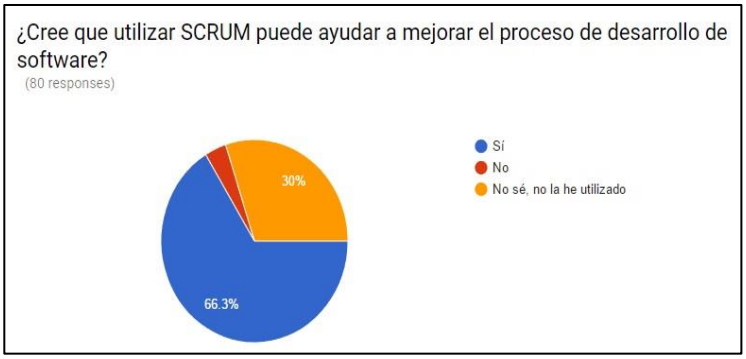
Figura 17. **Encuesta 5: ¿cómo conoció la metodología scrum?**



Fuente: *Metodología de desarrollo scrum*. https://docs.google.com/forms/d/1zw4u_dqwRduodGzgpeZbpZuNf4rFSgN9zSnkiM0xRBE/edit. Consulta: 18 de enero de 2017.

La mayoría de los encuestados ha conocido scrum en la universidad, seguido de los que la han conocido por su trabajo, 36,3 % y 31,3 % respectivamente. Lo que indica que es un tema que sí abordan en los cursos de la carrera.

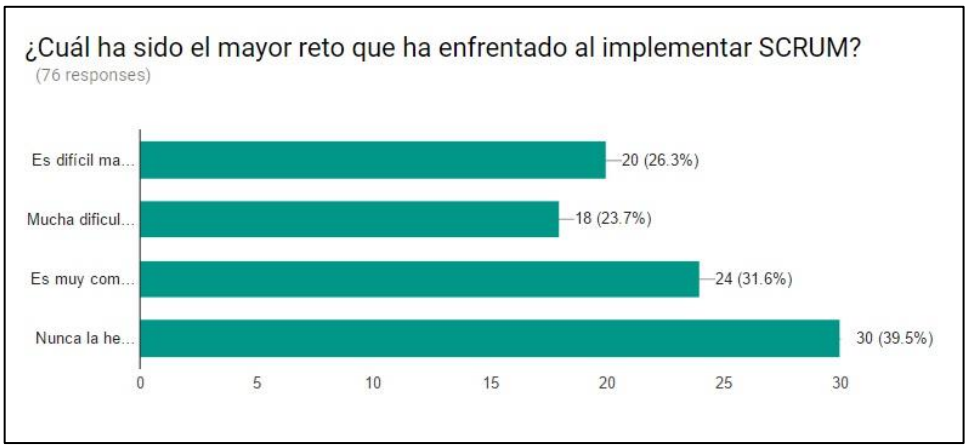
Figura 18. **Encuesta 6: personas que sí han utilizado scrum para desarrollar *software***



Fuente: *Metodología de desarrollo scrum*. https://docs.google.com/forms/d/1zw4u_dqwRduodGzgpeZbpZuNf4rFSgN9zSnkiM0xRBE/edit. Consulta: 18 de enero de 2017.

De las personas que sí han utilizado scrum para desarrollar *software*, el 66,3 %, dos tercios de los encuestados, consideran que la metodología sí les ayuda a mejorar sus procesos de desarrollo; solamente, 3 personas creen que esto no sucede.

Figura 19. **Encuesta 7: mayor reto que se enfrenta con la metodología**

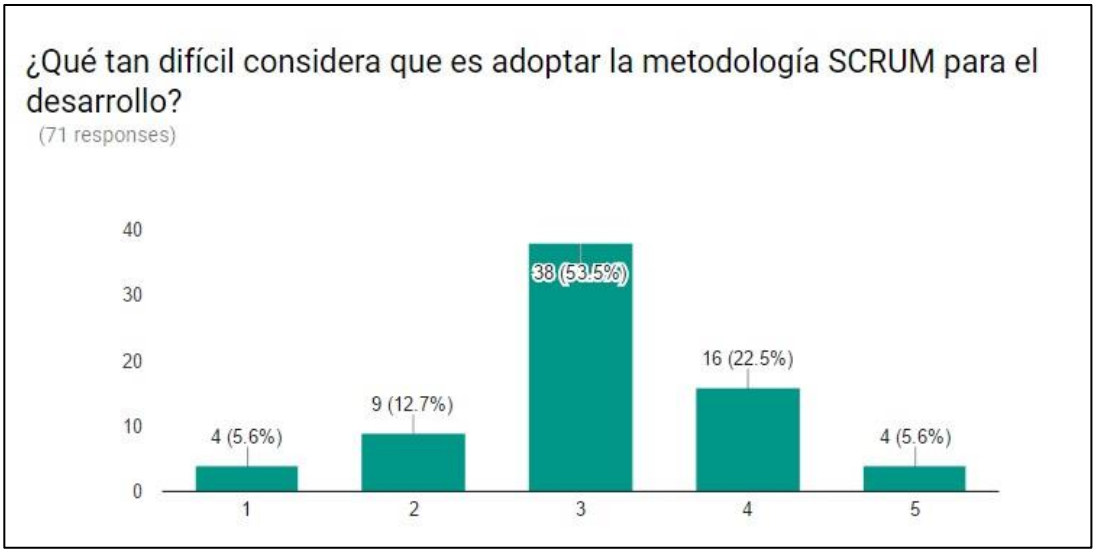


Fuente: *Metodología de desarrollo scrum*. https://docs.google.com/forms/d/1zw4u_dqwRduodGzgpeZbpZuNf4rFSgN9zSnkiM0xRBE/edit. Consulta: 18 de enero de 2017.

El mayor reto que se enfrenta con la metodología es decidir las historias de usuario que deben incluirse en cada *sprint*, según el 31,6 % de los encuestados. Seguido con la dificultad de manejar la relación con el dueño del producto (26,3 %) y la dificultad que implican las reuniones diarias (23,7 %).

Esto da una idea de que en scrum el mayor reto es la ponderación de cada historia de usuario en cuanto al tiempo que se le dedicará, lo cual complica la decisión de cuántas se incluyen en un *sprint* de una duración definida.

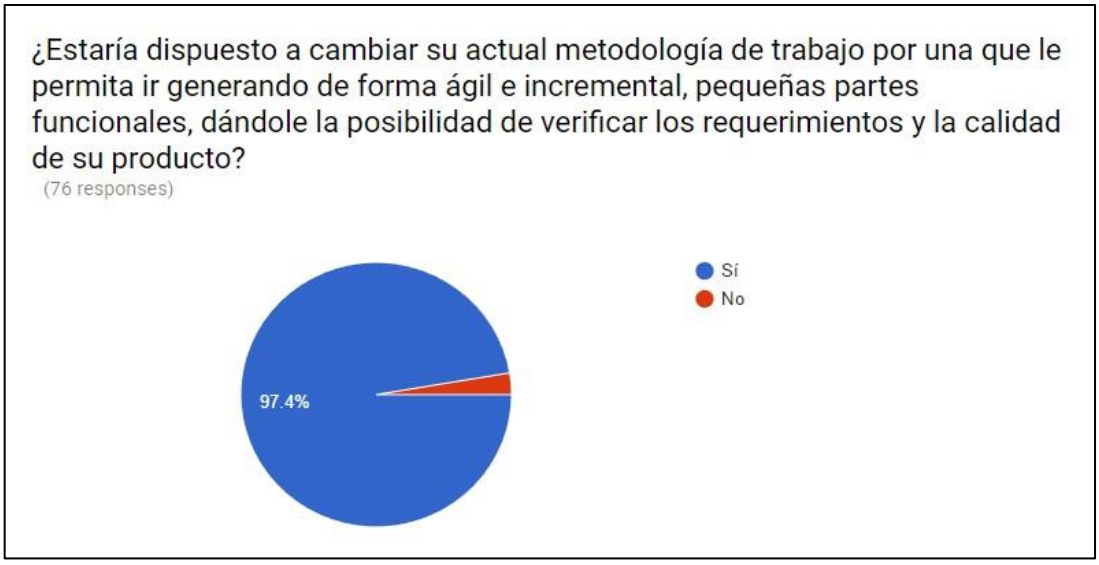
Figura 20. **Encuesta 8: los encuestados evalúan scrum como una metodología**



Fuente: *Metodología de desarrollo scrum*. https://docs.google.com/forms/d/1zw4u_dqwRduodGzgpeZbpZuNf4rFSgN9zSnkiM0xRBE/edit. Consulta: 18 de enero de 2017.

El 53,5 % de los encuestados evalúa scrum como una metodología con mediana dificultad de aprendizaje e implementación. Solamente 4 de los encuestados la califica como una metodología muy difícil de adoptar.

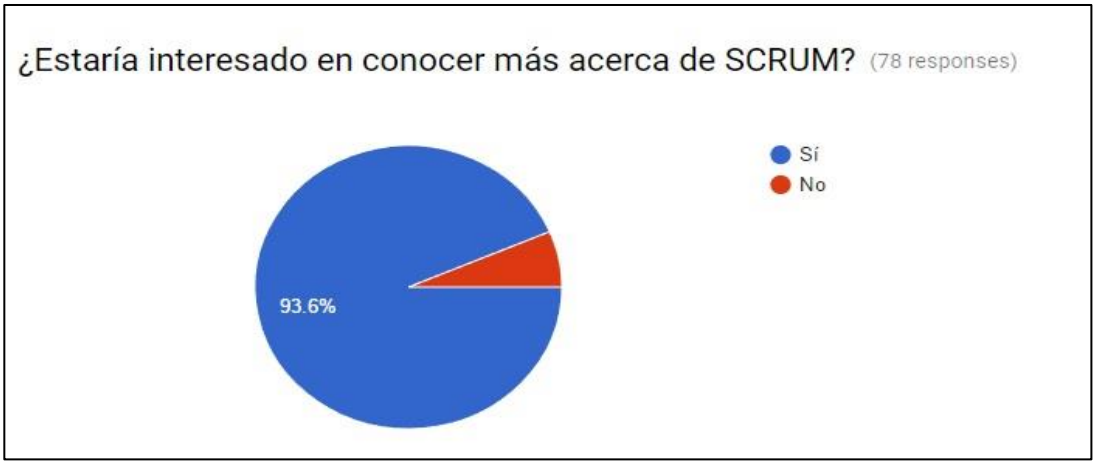
Figura 21. **Encuesta 9: ¿están dispuestos a cambiar la metodología que utilizan?**



Fuente: *Metodología de desarrollo scrum*. https://docs.google.com/forms/d/1zw4u_dqwRduodGzgpeZbpZuNf4rFSgN9zSnkiM0xRBE/edit. Consulta: 18 de enero de 2017.

De los encuestados, el 97,4 % está dispuesto a cambiar la metodología que utilizan para el desarrollo por una metodología que les ofrezca los beneficios que ofrece acrum.

Figura 22. **Encuesta 10: interesados en conocer más la metodología scrum**



Fuente: *Metodología de desarrollo scrum*. https://docs.google.com/forms/d/1zw4u_dqwRduodGzgpeZbpZuNf4rFSgN9zSnkiM0xRBE/edit. Consulta: 18 de enero de 2017.

El 93,6 % de los encuestados está interesado en conocer más la metodología scrum.

4.4. Análisis de resultados

Del total de encuestados, solamente el 16,3 % no conocen ni han oído hablar de la metodología scrum, el 27,5 % solamente la conocen y el 53,6 % la ha utilizado. Este parámetro representa una base para tomar con seguridad las respuestas de las siguientes preguntas en la encuesta.

Entre los puntos más importantes a resaltar está el hecho de que dos terceras partes: 66,3 % de los encuestados considera que utilizar acrum para el desarrollo de aplicaciones si puede ayudar a mejorar los procesos.

La mayoría de los encuestados, el 53,5 % la considera una metodología no muy difícil de adoptar; la mayor dificultad la capacidad de definir las historias que se incluyen en cada *sprint*, con un 31,6 %, seguido de la dificultad en mantener la comunicación con el dueño del producto, con un 26,3 %.

Cabe resaltar que se ha incluido la metodología scrum en los contenidos de los cursos impartidos en la USAC, ya que el 36,3 % de los encuestados menciona que allí conocieron la metodología; la conocieron en el trabajo, el 31,3 %; solamente un 12,5 % de los encuestados tuvo la motivación personal de investigar el tema en libros o Internet.

Casi la totalidad de los encuestados, el 97,3 % estaría dispuesto a cambiar la metodología de desarrollo que utiliza actualmente por una metodología que le brinde las ventajas que ofrece scrum; y el 93,6 % está interesado en conocer más a fondo la metodología.

CONCLUSIONES

1. Scrum, más que una metodología de desarrollo como tal, es un proceso organizacional para gestionar un proyecto de desarrollo que da directrices de cómo organizar el trabajo y cómo controlar los procesos de tal forma que el equipo sea capaz de adaptarse a los cambios o prever complicaciones en etapas tempranas para poder tomar las decisiones necesarias para solventarlas.

2. Aunque el ciclo de vida de scrum es simple y bien definido, adaptable a casi cualquier equipo de trabajo, la complejidad de su uso radica en que no hay formas correctas o incorrectas de llevar a cabo el proceso de desarrollo con SCRUM; cada equipo puede ir refinando el uso de la metodología y adaptándola a sus condiciones específicas, lo cual puede llevar algún tiempo. Esto no significa que la metodología no funcione en equipos nuevos en el tema, es solo que la metodología funciona mejor a medida que el equipo tome experiencia en su uso.

3. El desarrollo de aplicaciones web se beneficia con el uso de scrum en la medida en que permite ir obteniendo incrementos funcionales en períodos de tiempo relativamente cortos; permite que ponga en producción una primera versión por aumentar su funcionalidad en versiones posteriores, según los incrementos que se generen en el proceso.

4. En muchos casos de desarrollo de aplicaciones web es necesario preparar o promocionar su lanzamiento, por lo que se requiere de cierta

certeza en el tiempo que tomará el desarrollo hasta que esté lista para salir a producción; la capacidad de medir estos tiempos con exactitud es una ventaja en los equipos con experiencia en la utilización de scrum, pues la metodología obliga a hacer este cálculo y luego a evaluarlo de tal forma que cada vez sea más exacto.

RECOMENDACIONES

1. Dado que el cálculo del tiempo que toma el desarrollo de cada historia de usuario es una de las mayores dificultades que un equipo enfrenta al implementar scrum, es recomendable que utilicen tiempos de holgura, sobre todo al principio, mientras el equipo va tomando experiencia y midiendo su propio ritmo de trabajo para que con el tiempo los cálculos sean más fáciles y certeros.

2. Para garantizar los beneficios de la utilización de scrum es necesario que el equipo de desarrollo esté realmente comprometido con la metodología es decir aunque se considere obviar las reuniones diarias no afectará al desarrollo, no deben saltarse ninguna, pues está demostrado que ayuda a la identificación temprana de posibles problemas.

3. Para facilitar la utilización de scrum, es importante que el equipo de desarrollo le explique la metodología al cliente con la finalidad de que él mismo se comprometa a ser parte del proceso o nombre a quien lo represente, otorgándole la potestad de tomar decisiones sin tener que retrasar al equipo.

BIBLIOGRAFÍA

1. BECK Kent; BEEDLE Mike; VAN BENNEKUM Arie; COCKBURN Alistair, CUNNINGHAM Ward; FOWLER Martin; GRENNING James; HIGHSMITH Jim; HUNT Andrew; JEFFRIES Ron; KERN Jon; MARICK Brian; MARTIN Rober Cecil; MELLOR Steve; SCHWABER Ken, SUTHERLAND Jeff y THOMAS Dave. *Manifiesto por el desarrollo ágil de software*. [En línea]. <file:///C:/Users/sebas/Downloads/Manifesto%20of%20Agile%20Software%20Development.pdf>. [Consulta: 18 de enero de 2017].

2. BERNHARDT, Manuel. *Reactive web applications. Publicaciones Manning*. [En línea]. <https://www.manning.com/books/reactive-web-applications/>. [Consulta: 18 de enero de 2017].

3. KNIBERG, Henrik. *Scrum y XP desde las trincheras: cómo hacemos Scrum.* Estados Unidos: C4Media Inc., 2007. 122 p.

4. LUJÁN MORA, Sergio. *Programación de aplicaciones web: historia, principios básicos y clientes web.* España: Club Universitario. 354 p.

5. LUPETTI, Antonio. *The woork papers No. 1: structured process you must know to develop a web application.* [En línea]. <https://css-tricks.com/app-from-scratch-1-design/>. [Consulta: 18 de enero de 2017].

6. MATEU, Carles. *Desarrollo de aplicaciones web.* Barcelona, España: Eureca media, 2014. 378 p.

7. MENZINSKY, Alexander; LÓPEZ, Gertrudis y PALACIO, Juan. *Scrum manager, guía de información.* [En línea]. <http://www. scrummanager.net/files/scrum_manager.pdf/>. [Consulta: 18 de enero de 2017].

8. PALACIO, Juan. *Flexibilidad con scrum.* [En línea]. <http://www.scrum manager.net/files/flexibilidad_con_scrum.pdf/>. [Consulta: 18 de enero de 2017].

9. _____. *Gestión de proyectos. scrum manager.* [En línea]. <http://www.scrummanager.net/files/sm_proyecto.pdf/>. [Consulta: 18 de enero de 2017].

10. PRESSMAN, Roger S. *Ingeniería del software, un enfoque práctico*. 7a ed. México: McGraw Hill, 2010. 805 p.

11. SÁNCHEZ-ZUAÍN, Silvia; DURÁN, Elena. *Taxonomía de requisitos para aplicaciones web.* Chile: Universidad de Santiago El Estero, 2016. 10 p.

12. SCRUMstudy. *A guide to the SCRUM BODY OF KNOWLEDGE (SBOK GUIDE).* 3a ed. Arizona, USA: Phoenix, 2016. 342 p.

13. TAKADA, Mikito. *Single page applications in depth.* [En línea]. <http://singlepageappbook.com/>. [Consulta: 18 de enero de 2017].

14. TAKEUCHI, Hirotaka y NONAKA, Ikujiro. *The new new product development game.* [En línea]. <https://hbr.org/1986/01/the-new-new-product-development-game>. [Consulta: 18 de enero de 2017].

15. THÜER, Sebastián. *¿Qué debe saber un diseñador web? Técnicos, analistas, programadores y creativos.* [En línea]. <https://thuer. com.ar / analytics / 2009 / que - debe - saber - un-disenador-web>. [Consulta: 18 de enero de 2017].

16. WAKE, Bill. *INVEST in good stories, and SMART tasks.* [En línea]. <http:// xp 123.com / articles / invest-in-good-stories-and-smart-tasks / >. [Consulta: 18 de enero de 2017].