Descripción del producto a desarrollar

Escenario

Vistos los distintos tipos de metodologías que se contemplan, entramos a describir el entorno del cliente, el modo de trabajo, los equipos de trabajo, la estructura organizativa y el objetivo de producto para el cliente.

Se trata de una mediana empresa en la que el departamento de Sistemas de la Información está compuesto por dos áreas separadas de trabajo, Hardware y Software. En este caso concreto el departamento está formado por 10 miembros.

La gestión de proyectos se realiza con metodologías ágiles adaptadas. Esto quiere decir que no siempre tenemos equipos multidisciplinares como pueden exigir algunas metodologías inicialmente. No siempre te puedes permitir el *pair programing* debido a las dimensiones del departamento. Los integrantes de cada equipo pueden variar en función del proyecto/s en curso. En determinadas fases de un proyecto se puede echar mano de otro compañero, ya sea del mismo área o no, es decir hay miembros del equipo que se incorporan para unas tareas concretas y no durante todo el proyecto.

Solo el departamento IT utilizará la aplicación para la gestión de proyectos que pretendemos desarrollar. El resto de departamentos o no necesita esta gestión o ya tiene sus propias herramientas.

Teniendo en cuenta este tipo de limitaciones y otras muchas que iremos detallando, el principal objetivo del cliente es poder medir. El propio departamento necesita conocer los costes de tomar unas u otras decisiones, permitiendo analizar los resultados y mejorar en la toma de decisiones. Además otros departamentos como el de administración necesitan conocer los costes económicos de un desarrollo interno, soporte técnico, *deuda técnica*[4], etc. Es decir, tenemos el mismo personal para la ejecución de proyectos, para resolver dudas técnicas, resolución de incidencias, o bugs provenientes de otros desarrollos.

Actualmente no hay una gestión controlada de estas interrupciones y ante estas situaciones es necesario establecer prioridades y proceder, el equipo es autosuficiente para tomar las decisiones que considere oportunas. Como consecuencia los proyectos se pueden ver afectados en alguno de **los tres conceptos del triángulo de hierro.**

Por parte del departamento de IT y siguiendo con los principios ágiles, los integrantes del equipo no necesitan conocer cuánto tiempo han empleado en realizar un tarea, sprint, iteración, un proyecto...Lo necesario para ellos es conocer cuánto falta para llegar al objetivo.

El cliente no impone ninguna restricción del tipo de proyecto a acometer, si tiene que ser una aplicación propietaria, una aplicaciones base que podamos adoptar o un servicio *SaaS* (Software as a Service).

Alcance

En una primera aproximación se detectan los objetivos de negocio que detallaremos a continuación. Estos vienen determinados por ramas bien diferenciadas. En primer lugar los objetivos del propio departamento de IT, que necesita de una herramienta para gestionar proyectos con Scrum y en segundo lugar para departamentos externos que lleven el control de costes y análisis de proyectos internos.

La aplicación gestiona proyectos, usuarios y equipos. Los usuarios con un perfil personal se agrupan en equipos con un rol determinado. A su vez un equipo tiene asignado un proyecto. Y los proyectos contienen toda la información relacionada: actas, historias de usuarios, contratos, planificación, coste real, etc. Aunque no es un requisito propio de la metodología, es muy útil la posibilidad de tener la información asociada al proyecto.

Siguiendo la definición de Scrum, el cliente forma parte del equipo, pero por ahora no se sabe si este tiene que poder utilizar la aplicación o no.

Una vez procesados los objetivos de negocio, tenemos las tareas agrupadas por historias de usuario. Es necesario que estas tareas sean asignadas a los miembros del equipo. Para que se pueda asignar una tarea tiene que estar completa, con coste estimado de realización. Los miembros del departamento IT tienen una restricción de productividad: Les impide tener varias tareas en ejecución. A medida que las tareas estén completas, a los usuarios se les van asignando otras en ejecución. Además todo esto debe permitir recoger la duración real del proyecto.

En numerosas ocasiones los usuarios interrumpen la ejecución de una tarea para realizar otra más urgente. Es el caso de bugs informáticos, nuevas funcionalidades en proyectos anteriores, que tienen mayor prioridad que el proyecto en curso y es necesario cambiar de tarea.

Los errores o tareas clasificadas con prioridad “urgente” se gestionan del siguiente modo. Cuando se decide qué tenemos que hacer, la tarea se asigna a uno de los empleados del Dpto. y este valorará su alcance, pudiendo llegar a ser un proyecto nuevo o una versión de otro anterior. Si se trata de un error o algo de menor alcance se decide quién la ejecuta y se procede. Puede ser necesaria la intervención de varios compañeros.

Como ya hemos comentado anteriormente, debemos conocer el avance y coste de cada proyecto, incluso de los que están en ejecución, teniendo en cuenta que para el equipo Scrum lo necesario será conocer qué falta por hacer.

Por ahora tampoco se contempla que la documentación se gestione de forma interna con herramientas de colaboración. Puede ser necesario adjuntar archivos a los proyectos e incluso solventar los problemas de versión de archivos de un modo transparente. Un ejemplo de documentación a añadir pueden ser los manuales de usuario.

Para mostrar la información de control se debe tener la posibilidad de exportar en distintos formatos que nos permitan trabajar con los datos. A través de los informes también podremos conocer el origen, los costes y toda la información relacionada con los trabajos ejecutados por el equipo.

Respecto a las métricas que deberá permitir, partimos de dos requisitos, uno viene dado por la gestión de proyectos ágiles, permitir estimar el coste de los objetivos. Y el segundo requisito, dado por el cliente, es que debe permitir conocer las horas invertidas en cada tarea y/o el coste final real de cada tarea. Aunque se tiene claro que el cálculo de los costes es un requisito prioritario, por ahora no existe una decisión unilateral de qué métrica utilizar.

Podríamos utilizar el tiempo en horas pero no para todo el mundo significa lo mismo una hora de su tiempo. Podemos asignar un coste monetario a una hora de trabajo. O concretar una hora de trabajo y un usuario específico. Pero las tareas luego no tienen el mismo valor, los empleados tienen un coste variable...Entramos en un escenario que no tenemos claro y tomaremos una decisión durante la marcha del proyecto. Por ahora tomamos la decisión de que controlamos únicamente el tiempo.

Product Backlog

A continuación detallamos la primera versión de nuestro *Product Backlog*. Conforme vayamos completando las historias de usuarios este podrá sufrir modificaciones en cada iteración. Se han divido en dos tipos de historias de usuarios, las historias que aportan funcionalidad y valor al usuario y las historias no funcionales pero que de algún modo aportan valor y hemos considerado oportuno que aparezcan.

Identificador Objetivo	Descripción	Coste estimado	Condiciones de satisfacción	Valor aportado
A1	Dar alta proyecto	300	Quiero crear el proyecto a traves de un formulario.	1000
			Al crear el proyecto me debe permitir asignar miembros del equipo	
A2	Eliminar/Finalizar proyectos	300	Podemos eliminar un proyecto que no tiene asignados miembros del equipo ni historias de usuario asociadas.	200
			Si el proyecto ya tiene informacion asociada no se elimina cuando es dado de baja	
			Si el proyecto tiene las tareas asociadas en estado finalizado se cerrará.	
A3	Alta Historia de Usuario	200	Deberá ser asignada a un proyecto.	1000
			Debe contener un título corto identificador, una breve descripción, el valor aportado para el cliente y el coste estimado.	
A4	Modificar Historia de Usuario	200	Se podrán modificar los datos de alta salvo el identificador.	500
			No se podrá modificar los datos de alta si la historia se encuentra en ejecución.	
A5	Acceso cliente, Historias Usuario	300	Al acceder a una historia de usuario puede modificar el valor aportado	500
			Conocer el estado de las historias, por empezar, en ejecución o finalizada.	
A6	Creación de tareas	400	Para la creación de una tarea debemos indicar un nombre, una descripción breve y un coste estimado	1000
			Deberá estar asignada a una historia de usuario.	
			Puedo asignar la tarea a un miembro del equipo	

***Product Backlog* Versión 1**

Identificador Objetivo	Descripción	Coste estimado	Condiciones de satisfacción	Valor aportado
A7	Procesar tareas	400	Puedo autoasignarme tareas que no estén asignadas a otro miembro del equipo.	1000
			Puedo rellenar el coste de real de la tareas asignadas.	
			Puedo modificar el estado de una tarea: en ejecución, pausa, terminada.	
A8	Acceso Director, Proyectos	300	Al acceder al proyecto se le da acceso a toda la información de historias de usuario y tareas que están relacionadas.	600
			Exportar informes con los datos actuales.	
A9	Representacion visual de los datos	1200	Puedo ver el Gráfico Burn-Down en base a los datos de cada proyecto.	500
			Puedo ver el Gráfico Burn-UP en base a los datos de cada proyecto.	
A10	Añadir documentacion Externa	200	Puedo decargar documentos adjuntos a un proyecto.	100
A11	Control de versiones para archivos	2000	Puedo conocer el historial de un documento que anteriormente he descargado.	200
			Puedo conocer el historial de un documento que anteriormente he adjuntado al proyecto.	
A12	Representación visual de los elementos Scrum (Kanban)	1300	Puedo consultar la informacion de proyectos y sus elementos de forma visual.	400
A13	Asignación de permisos entre roles de usuario	1500	El administrador de la plataforma puede asignar permisos para una funcionalidad concreta	100
A14	Workflow metodología ágil (Scrum)	1200		
A15	Reuniones (scrum diario)	1000		
A16	Sprint Backlog	300		

***Product Backlog* Versión 1**

Identificador Objetivo	Descripción	Coste estimado	Condiciones de satisfacción	Valor aportado
B1	Login de usuarios	400	Al acceder a una funcionalidad/ruta se deberá comprobar que un usuario existente en la BD se ha identificado con su usuario y password.	1100
B2	Crear usuarios	200	Al crear un usuario tiene que estar disponible su acceso	1100
			Si se trata de un miembro del equipo estará disponible para asignar historias de usuario	
			Si se trata de un miembro del equipo estará disponible para asignar tareas	
B3	Dar de baja usuarios	300	Al dar de baja un usuario deja de estar disponible su acceso	400
			Si es un miembro del equipo dejará de estar disponible para ser asignado con tareas u historias de usuario.	
B4	Definir y crear una interfaz grafica	200	Será aprobada por el cliente. En ningún caso será una interfaz real, simplemente un boceto de la misma para definir estructura y la organización del contenido.	200
B5	TDD (Test Driven Development)	>600	Su propio funcionamiento es la condición de aceptación	1100
B6	Tecnología y Herramientas	800	Las conclusiones finales, como resultado de los problemas o dificultades que hayamos tenido nos servirán para conocer el grado de satisfacción.	1100
Total		**13000**		

***Product Backlog* Versión 1**

Estudio de mercado

Antes de pasar a desarrollar el proyecto, se analizará el mercado para ver las soluciones existentes y ver si es necesario realizar un desarrollo propio, o adaptar alguna aplicación que ya exista en el mercado.

Actualmente el desarrollo del software avanza cada vez más rápido y es posible que si investigamos a fondo soluciones que ya tenemos en el mercado puedan resultar completamente funcionales para nuestro cliente y únicamente tengamos que gestionar su implantación después de verificar que los objetivos de nuestro cliente están cubiertos.

En esta primera fase de investigación de mercado detectamos las siguientes aplicaciones que están relacionadas con alguno de los objetivos del proyecto. A partir de la definición de nuestras historias de usuario, se procederá con el análisis de las mismas para revisar qué aspectos no son cubiertos y comparar con las prioridades asignadas por nuestro *Product Owner* en los objetivos del mismo. Si en algún caso no se cubren las historias de usuarios con mayor prioridad esta aplicación será descartada inmediatamente.

Detalle de las aplicaciones destacadas

Trello.com

Se podría definir como un tablero virtual, nos permite configurar varios tableros, añadir otros usuarios con los que compartir nuestros tableros y un historial de cambios así como notificaciones. En definitiva es una pizarra Kanban a la que podemos acceder vía web. Al ser únicamente tableros deja a un lado la parte de gestión de proyectos, historias de usuarios y su posterior análisis.

Jira

Bastante más complejo que el anterior ya que no se trata de un simple aplicación web. Cuenta con multitud de plugins que nos permiten añadir funcionalidad adicional en base a nuestros requisitos, podemos desarrollar nuestros propios plugins e incluso comercializarlos. La mayor dificultad de esta opción sería el adecuarnos a una forma concreta de hacer las cosas. Aunque sea personalizable siempre existe la posibilidad de que el método de trabajo no sea el que busca el cliente. Respecto al precio por ahora estaríamos hablando de 50$ mensuales permitiendo hasta 15 usuarios, totalmente escalable si crecemos en usuarios.

Esta opción nos obligaría a formarnos en dicha herramienta o contratar alguien con experiencia, además de formarnos como usuarios para poder implantar la solución en nuestro cliente.

Groupcamp.es

Se presenta como una aplicación web para el trabajo en equipo añadiendo la gestión del tiempo y los presupuestos. Permite conectarse con otras aplicaciones como Office, Google, email interno. La falta de plugins o acceso a un API, nos impediría modificar la aplicación y/o añadir nuevas funcionalidades específicas. Además aunque la aplicación permite la gestión de proyectos y una amplio abanico de funcionalidades relacionadas no trata una metodología concreta como es nuestro caso.

Detalladas las tres aplicaciones que más nos han interesado durante el análisis de mercado y en un principio nos podrían ayudar, tras un segundo análisis más exhaustivo nos encontramos una aplicación que no gestiona proyectos, otra aplicación que gestiona proyectos sin seguir una metodología y una tercera aplicación, plataforma con plugins y api, muy completa la cual no conocemos y sería necesaria formación y/o ayuda externa y que además podría añadir complejidad con opciones que nuestro cliente no tiene intención alguna de utilizar por ahora.

Aspectos éticos y de seguridad

A la hora de llevar a cabo el proyecto deberemos tener presente una serie de aspectos legales, éticos y de seguridad que pueden influir en los requisitos del mismo.

Desde el **punto de vista Ético**, la aplicación trata relaciones laborales del grupo de desarrollo con otros miembros de la empresa del mismo rango o superior.

Aunque la propia empresa no persiga la finalidad de controlar a los empleados, estos pueden pensar lo contrario. Pueden llegar a dudar de la privacidad ante otros compañeros, etc. Además para la empresa puede resultar muy tentador comparar las métricas de uno u otro empleado. Esto influye directamente en una de las historias de usuario proyectadas (A7) según la cual solo debemos ver nuestro control de tiempos y un coste total del resto de usuarios.

Si durante la ejecución del proyecto surgen otras cuestiones relacionadas con los aspectos éticos deberemos tenerlas en cuenta para garantizar a las partes implicadas que se cumplen estas condiciones y su privacidad está asegurada en la medida de lo posible.

Como toda aplicación informática deberá tener presente en todo momento la **seguridad**. Una vez esté concretado el tipo de aplicación (extranet, intranet, escritorio), se tendrán en cuenta las medidas de seguridad más adecuadas. Por ahora se conoce que la aplicación manejará datos de empleados, de proyectos internos de la empresa, de clientes y proveedores que

deberán confiar en la herramienta que les proporciona nuestro cliente. En este punto contamos con cierta incertidumbre, es decir, lo que para alguien se considera inseguro puede ser una garantía de seguridad para otro. Aclaramos este punto con el siguiente ejemplo. Una aplicación basada en software libre y mantenida por una amplia comunidad de usuarios nos da las garantías de que un fallo en la seguridad será notificado y solucionado con bastante premura. Pero a su vez esto hace que un posible atacante pueda aprovecharlo para conocer nuestras debilidades antes de ser subsanadas.

Otro aspecto de vital importancia serán los aspecto legales. Estamos realizando una página web con datos personales de clientes, proveedores y empleados, es decir deberemos cumplir la LOPD. En este caso debemos incluir una referencia de los datos personales que trata la página web (nombre y correo electrónico), explicando el uso que haremos de estos datos. La LOPD también nos obliga a cumplir una serie de políticas de seguridad y registrar nuestro fichero de datos personales en la AEPD, Agencia Española de Protección de Datos [7].

Una medida de seguridad a la hora de utilizar un mecanismo de identificación mediante usuario y contraseña será aplicar un mecanismo de cifrado con *salt* [5]y a su vez un algoritmo de cifrado concreto que por ahora no se considere inseguro en caso de que alguien acceda al almacenamiento de nuestras contraseñas. El uso de *salt criptográfico* añade una dificultad extra a la hora de intentar crackear un hash almacenado en la base de datos.

En cuanto a la seguridad física, se podrá dar el caso de que tengamos que elegir un proveedor u otro si los datos requieren de un hardware específico para garantizar mayor nivel de seguridad, contratar un hardware específico (Ej: firewalls), o incluso instalar la aplicación en un servidor propio del cliente de modo que éste tenga el control de los mismos. Por ahora estos requisitos no se conocen y se plantean durante la ejecución del proyecto.

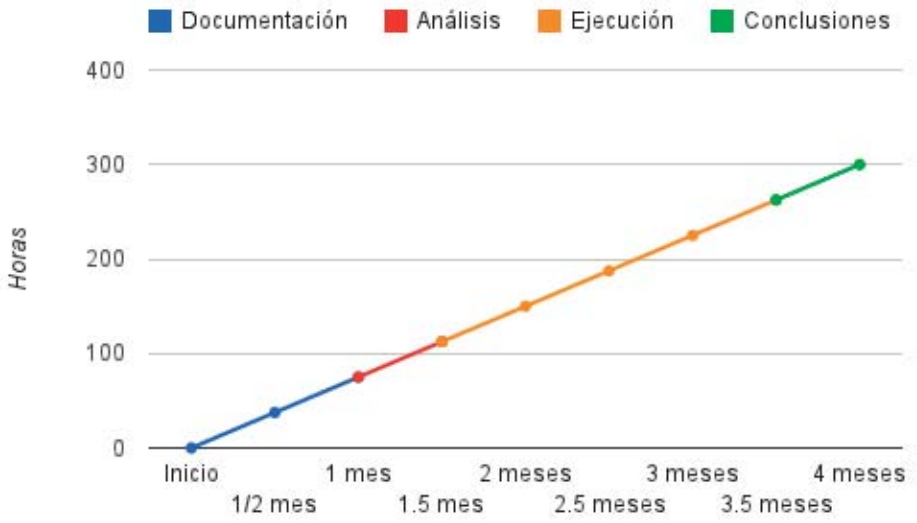
Planificación y metodología

En primer lugar detallar que la **metodología** a seguir se basa en las metodologías ágiles que hemos descrito en este capítulo, concretamente Scrum. Siguiendo el guión[6] predefinido para Trabajos Fin de Grado, describiremos la ejecución del proyecto con los conceptos definidos por la metodología que realmente nos resulten útiles buscando el objetivo principal de las metodologías ágiles, no utilizar una metodología rígida sino adaptarla a nuestras necesidades.

Para afrontar la ejecución del desarrollo se han identificado los objetivos de negocio en historias de usuario, asignado un coste aproximado, un valor de negocio y priorizando en función de estos valores. Para estimar el coste no se han utilizado técnicas específicas como las mencionadas en la descripción de la metodología, debido a que el equipo se compone de una única persona y las ventajas que nos ofrecen son fomentar la participación y evitar los malentendidos entre los miembros de un equipo. Para dar prioridad hemos tenido en cuenta que nuestros objetivos de negocio no son los mismos que para un cliente real. En vez de un beneficio económico, se busca un beneficio académico.

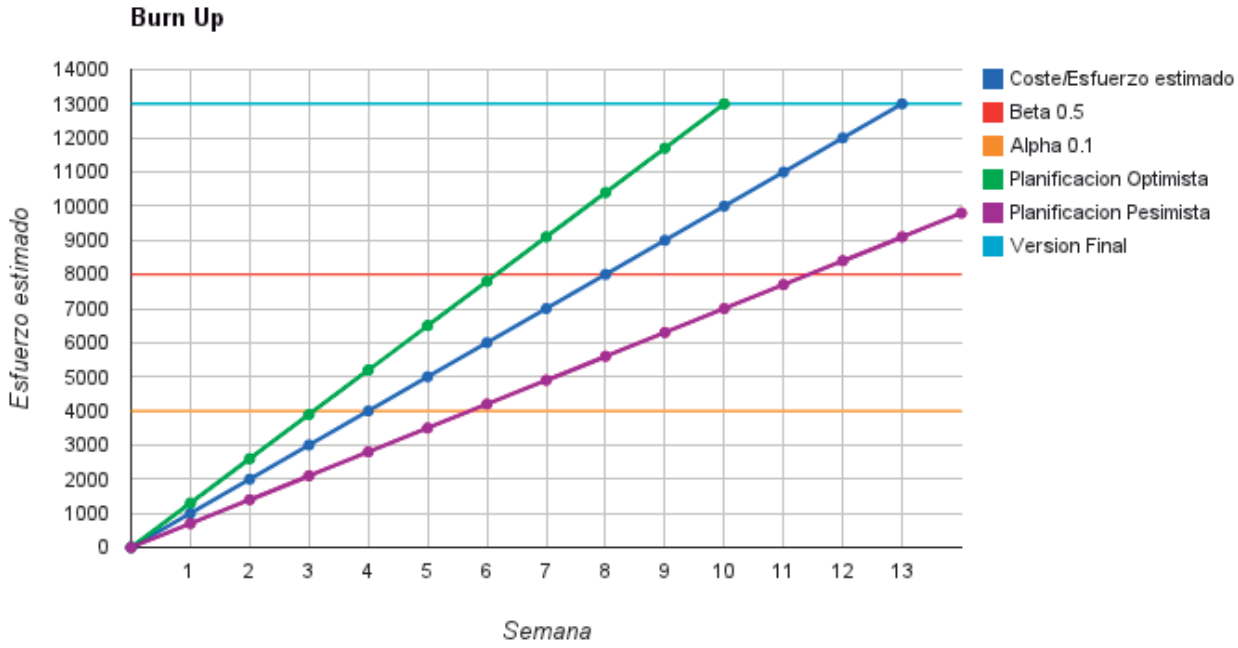
Otro punto a diferenciar entre Scrum y nuestra ejecución son las reuniones. Aunque se han celebrado reuniones después de cada iteración (reunión de revisión y de preparación) y se ha ido revisando avance del sprint durante su ejecución (reunión diaria), en ambos casos no se han seguido al pie de la letra las recomendaciones que implica Scrum; es decir, realizar la reunión de forma automática, sin tener en cuenta que es la primera tarea del día, juntar reuniones de revisión con reuniones de preparación, etc.

Respecto a la **planificación** hay que tener en cuenta que hay establecidas las siguientes limitaciones: 300 horas y entrega a mediados del mes de Junio. Con una dedicación de 19 horas a la semana en 4 meses cumpliríamos la horas estipuladas y dejaríamos un pequeño margen para posibles interrupciones. Dentro de la planificación de sprints o iteraciones tenemos en cuenta que disponemos de un tiempo limitado y para obtener un producto con una funcionalidad aceptable los sprints no deben superar la duración de 1 semana. A partir de la duración de 1 semana por sprint hemos asignado coste estimado en el *Product Backlog* y la velocidad media de desarrollo debería ser de 1000 puntos por semana.



En el gráfico anterior se representa la línea de tiempo planificada. Al comienzo del mismo está la fase de documentación, compuesta por la exposición del problema y documentación sobre el tema del proyecto que formará el primer mes. La segunda parte, Análisis, nos ocupa la primera mitad del segundo mes, que continúa con el desarrollo de la aplicación durante 2 meses. Y para finalizar dejamos 2 semanas como mínimo para la redacción de la memoria final, sacar conclusiones y finalizar el proyecto.

Ahora para reflejar la correspondiente planificación haciendo uso de las herramientas que nos proporciona Scrum vamos a utilizar el gráfico *Burn-up.*



El *Burn-up* anterior nos muestra la comparativa de una estimación ideal frente a una optimista y otra pesimista. Para realizar esta comparativa se ha aumentado y disminuido en 300 puntos la velocidad por semana estimada anteriormente. Además permite conocer en qué momento de una u otra estimación tendremos disponibles las distintas versiones. Para ello se estima que la primera versión utilizable será después de la 2ª iteración en los 4000 puntos (Alpha) y una segunda versión de prueba más estable a los 8000 puntos (Beta). Ahora la gráfica nos permite conocer cómo la versión "Beta" puede acabar retrasándose hasta las 11 o 12 semanas de duración. Una vez este gráfico empiece a dibujar la línea de la duración real nos iremos haciendo una idea de cuál es nuestra velocidad y estimar en cada momento cuál será nuestra fecha fin para cada versión.

Dada la planificación anterior podemos definir los sprints agrupando historias de usuario según su prioridad. La siguiente tabla nos muestra las historias de usuario priorizadas hasta la fecha y agrupadas por iteraciones hasta la iteración 2 que forman la versión Alpha. El resto de historias priorizadas o no serán analizas e incluidas en próximas revisiones del *Product Backlog.*

N Sprint	Identificador Objetivo	Descripción	Coste estimado	Valor aportado	Prioridad
Iteración 0	B6	Tecnología y Herramientas	800	1100	1
			800		
Iteración 1	B1	Login de usuarios	400	1100	2
	B5	TDD	600	1100	3
	B2	Crear usuarios	200	1100	4
	A1	Dar alta proyecto	300	1000	5
	A2	Eliminar/Finalizar proyectos	300	200	6
	B4	Definir y crear una interfaz gráfica	200	200	7
			2000		
Iteración 2	A3	Alta Historia de Usuario	200	1000	8
	A4	Modificar Historia de Usuario	200	500	9
	A6	Creación de tareas	400	1000	10
	A7	Procesar tareas	400	1000	11
			1200		
Versión Alpha			4000		
	A8	Acceso Director, Proyectos	300	600	12
	A5	Acceso cliente, Historias Usuario	300	500	13
	A9	Representación visual de los datos	1200	500	?

Destacar cómo en algunos casos no se llega a la planificación estimada y en otros la sobrepasamos. Aunque podríamos mezclar historias de usuario se ha preferido dejar las historias de usuario agrupadas siguiendo la prioridad asignada inicialmente. La iteración 1 tiene un coste estimado de 2000 puntos y aunque podemos separarla creando otra iteración se prefiere dejar estas historias en la misma iteración puesto que son la toma de contacto con el framework de acceso a datos y TDD.

Desarrollo

En esta sección vamos a detallar cómo hemos afrontado el proyecto, la descomposición de las historias de usuario en tareas, entrando en detalle de las tareas que se consideren oportunas y detallando la evoluciones, describiendo así la organización y metodología seguidas para el desarrollo.

Historias de Usuario

En el *Product Backlog* se han incluido las historias de usuario funcionales que enumeramos como resultado de los objetivos de negocio y en segundo lugar historias de usuario especiales que contienen requisitos no funcionales o que se dan por supuesto. Destacar que en el *Product Backlog* no se ha tenido en cuenta las primeras fase del TFG, la ejecución de la memoria, algunos aspectos de formación e investigación en metodologías ágiles ni la preparación de la defensa final.

Durante la creación del *Product Backlog* y descomposición de historias de usuario en tareas, nos hemos perdido una parte valiosa de la metodología que es poder ejecutar esta tareas en equipo, permitiendo contrastar opiniones y detectar posibles errores. Como base para comenzar estas tareas se han tenido en cuenta algunos consejos, malas prácticas, consejos...como por ejemplo:

- Escribir **qué** se debe hacer y no **cómo** se debe hacer.
- Las tareas deben seguir los principios INVEST[7].

Iteración 0

Esta primera iteración denominada “Iteración 0” recibe este nombre por el hecho de la historia de usuario (B6) que agrupa. Se ha dudado si incluirla en el *Product Backlog*. Finalmente después de varias reflexiones se cree conveniente que quede reflejado ya que se considera como un punto igual o más importante que el resto y el *Product Owner* o cliente debe ser consciente de ello.

Backlog	Tarea	Tipo	Descripción	Responsable	Estimación
B6	Tarea 1	Investigación	Brainstorming de tecnologías/herramientas	J.M	20%
B6	Tarea 2	Investigación	Brainstorming de requisitos técnicos	J.M	20%
B6	Tarea 3	Investigación	Documentarse sobre las ideas destacadas	J.M	30%
B6	Tarea 4	Investigación	Toma de decisiones	J.M	20%
B6	Tarea 5	Investigación	Control de versiones & Scrum	J.M.	10%

En primer lugar detallar cómo se ha rellenado la columna de estimación en este y el resto de *sprints backlogs*. En este caso y debido al entorno y variabilidad que podemos obtener en las horas dedicadas en un día se ha decidido incluir un porcentaje de la duración completa del Sprint. Dado el tipo de tareas que tratamos en esta iteración, la duración real de la misma no será muy valiosa para verificar las estimaciones realizadas.

Respecto a esta iteración cabe destacar la tarea 4, "Toma de decisiones". Partiendo de los requisitos del cliente, conocemos que deberá ser accesible desde varias organizaciones; por ello podemos concretar que vamos a desarrollar una aplicación web ya que facilita dicho requisito. Este tipo de aplicaciones se ven más expuestas a posibles ataques y la información puede estar expuesta, pero aplicando la medidas oportunas durante la fase de desarrollo se asume el posible riesgo.

Una vez conocido el tipo de aplicación las tecnologías a utilizar son más concretas aunque seguimos teniendo muchas posibilidades. En este punto y teniendo en cuenta el contexto y los plazos de ejecución, la experiencia del equipo será un punto a tener en cuenta a la hora de

elegir la tecnología. Por ello se cierran las opciones a programar con PHP valorando si seleccionar un framework conocido o no. Las últimas opciones a elegir son *Silex* [8] y *Codeigniter*[9]. El primero se trata de un microframework que destaca por su gran velocidad de desarrollo con multitud de librerías y su gran modularidad para seguir desarrollando aplicaciones más complejas. Además se basa en *Symphony*[10], otro framework *PHP* que ha se ha ganado un nombre en la comunidad de desarrolladores durante los últimos años; sin embargo su curva de aprendizaje es muy elevada. En este punto y teniendo en cuentas otros objetivos tomamos la decisión de utilizar *Codeigniter* con el fin de ahorrar tiempo para documentación e investigación de otros temas, ya que este framework es conocido por el equipo.

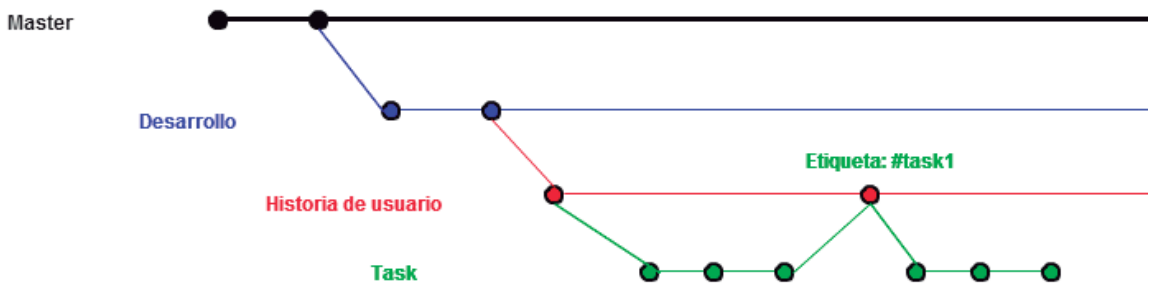
Una vez seleccionado el framework buscamos un *ORM*[11] que nos facilite la implementación en la capa de acceso a datos. En este caso elegimos *Datamapper*[12], no se conoce ningún *ORM* anteriormente por lo que la decisión está basada en la documentación encontrada del propio *ORM* ya que está pensado únicamente para utilizar con Codeigniter y su instalación es muy sencilla. Otros *ORM* como Doctrine más conocidos pueden resultar más complejos inicialmente.

En segundo lugar y para el almacenamiento de los datos que conectará con el *ORM* utilizamos *MySQL*[13], simplemente por ser el SGBD más conocido por el equipo y su facilidad de uso.

Para llevar a cabo la implementación de las distintas funcionalidades y teniendo en cuenta que estamos utilizando una metodología ágil vamos a profundizar en el desarrollo guiado por test o más conocido como *Test Driven Development* (TDD). Por ello en esta fase nos hemos adentrado en su filosofía y cuál es el método a seguir sin centrarnos en cómo implementarlo en PHP. Para ello se ha incluido una historia de usuario específica en la siguiente iteración.

Por último detallar que para llevar un control del desarrollo utilizaremos software de control de versiones, concretamente *GIT*. El resto de opciones son desconocidas y esta nos permite una instalación descentralizada y no requiere configuraciones complejas. Siguiendo el objetivo principal el flujo de trabajo con *GIT* vendrá dado por la metodología ágil. De este modo con cada historia de usuario crearemos un rama (*branch*) a partir de la cual iremos creando otras ramas por cada tarea dentro de la misma historia y que fusionamos a la rama principal de la historia una vez esté terminada la funcionalidad concreta de la tarea. Posteriormente podemos eliminar las ramas de las tareas, guardando una referencia a la rama eliminada con una etiqueta. Este método será seguido a su vez con la ramas que hacen referencia a las historias de usuario. Una vez que son aprobadas, pasan a la rama principal de desarrollo[14].

En el siguiente diagrama podemos ver el ciclo de creación de ramas junto con las etiquetas que se crean al fusionar una rama en otra y ser eliminadas. Ej: La rama *task1* se fusiona con la rama “historia de usuario” y creamos una etiqueta *#task1* antes de seguir con la siguiente tarea en la rama *task2*.



Iteración 1

En esta segunda iteración (considerada la primera desde el punto de vista de los requisitos del cliente) vamos a tratar las historias de usuario A1, A2, B1, B3 y B4. Estas historias se descomponen en el siguiente *Sprint Backlog.* Como vemos en el tabla, el número de tareas respecto a la iteración anterior ha aumentado considerablemente. Teniendo en cuenta el número de tareas y que existen temas desconocidos para el encargado de las mismas, ampliaremos la duración de esta iteración a 2 semanas (coincidiendo con el esfuerzo estimado y planificación del mismo).

Backlog	Tarea	Tipo	Descripción	Responsable	Estimación
B1	Tarea 1	Análisis y diseño	Modelo de datos	J.M	10%
B1	Tarea 2	Implementación	Crear clases MVC	J.M	5%
B2	Tarea 3	Implementación	Creación de formularios y métodos	J.M	5%
B5	Tarea 3	Documentación	TDD y PHP	J.M	20%
B5	Tarea 4	Implementación	Creación de test unitarios	J.M	10%
A1	Tarea 5	Implementación	Implementar MVC para proyectos	J.M	10%
B5	Tarea 6	Pruebas	Creación de test unitarios	J.M	15%
A2	Tarea 7	Implementación	Implementar funcionalidad	J.M	5%
A2	Tarea 8	Pruebas	Creación de test unitarios	J.M	10%
B4	Tarea 9	Implementación	Programar la estructura y diseño de las vistas	J.M	10%

Descompuestas las tareas de las historias que comprenden esta iteración vamos a detallar los aspectos más importantes a la hora de llevar a cabo la iteración.

Las historias B1 y B2 son la base para comprender el funcionamiento del *ORM* seleccionado. A través de un ejemplo que implementa un mecanismo de Login y creación de usuarios creamos

nuestros modelos y controladores. Al mencionar modelos y controladores estamos refiriéndonos al patrón MVC[15], a través del cual estructuramos la aplicación.

En la siguiente historia de usuario (B5) comenzamos con la búsqueda de la herramientas necesarias para llevar a cabo los test unitarios de forma más cómoda. Los test unitarios es la primera tarea que implica la técnica TDD. En este caso y debido que es nuestra primera aproximación a esta técnica, partimos de un desarrollo ya realizado sobre el que crearemos los test unitarios. Como framework principal para test unitarios en *PHP* tenemos *PHPUnit*; a la hora de abordar los tests sobre nuestras clases y métodos específicos para el framework elegido, utilizaremos otra librería externa, *my-cuinit*[16] que nos facilite la creación de test entre *PHPUnit* y *CodeIgniter*.

Obviando los pasos de instalación y configuración nos centraremos en un ejemplo de test unitario con el fin de clarificar su funcionamiento. En este ejemplo vamos a testear el método privado *encrypt* de la clase modelo *User.*

```
function _encrypt($field)
    {
        if (!empty($this->{$field}))
        {
            if (empty($this->salt))
            {
                $this->salt = md5(uniqid(rand(), true));
            }
            $this->{$field} = hash("sha512",$this->salt . $this->{$field});
        }
    }
}
```

Dado un nombre de campo (valor del campo *$field*) y un *salt* aleatorio guardará en la propiedad que se corresponde con el valor del campo el contenido de dicha propiedad concatenado con el *salt* y encriptado mediante el algoritmo *sha512.* Si el contenido de la propiedad es vacío no se hace nada y si no tenemos un *salt* generamos uno aleatorio y encriptamos.

Definido el funcionamiento del método podemos testear las siguientes condiciones:

- Si la propiedad a la que referencia el parámetro $field está vacía/empty (no tiene valor) la propiedad deberá seguir vacía.

```php
public function testEncryptEmptyPassword()
{
    $u = new User();
    $u->password = '';
    $u->_encrypt("password");
    $this->assertEmpty($u->password);
}
```

- Si la propiedad salt está vacía al ejecutar el método se creará un salt y almacenará el valor en la propiedad.

```php
public function testEncryptEmptySalt()
{
    $u = new User();
    $u->password = 'password';
    $u->salt = '';
    $u->_encrypt("password");
    $this->assertNotEmpty($u->salt);
}
```

- Si la propiedad salt tiene un valor este será el mismo después de ejecutar el método.

```php
public function testEncriptSaltEquals()
{
    $u = new User();
    $u->password = 'password';
    $u->salt = 'saltEncripted';
    $u->_encrypt("password");
    $this->assertEquals($u->salt, 'saltEncripted');
}
```

- Si la propiedad que pasamos como parámetro tiene valor, después de ejecutar el método no será el mismo.

```
public function testEncriptPasswordNotEquals()
{
    $u = new User();
    $u->password = 'password';
    $u->_encrypt("password");
    $this->assertNotEquals($u->password, 'password');
}
```

- Tras ejecutar el método el valor almacenado en la propiedad que hace referencia el valor del campo $field deberá ser igual a la concatenación del salt con el valor de la propiedad encriptados con el algoritmo sha512.

```
public function testEncriptPasswordSaltSHA512()
{
    $u = new User();
    $u->password = 'passwordSinEncriptar';
    $u->salt = 'saltEncripted';
    $u->_encrypt("password");
    $this->assertEquals($u->password, hash("sha512",'
        saltEncripted' . 'passwordSinEncriptar'));

}
```

En este punto hemos creado 5 test unitarios que verifican el correcto funcionamiento del método *encrypt*. La finalidad de los test unitarios es testear la única responsabilidad de un método, en este caso encriptar la cadena correspondiente cuando se cumplen las condiciones.

Un test está formado por tres partes, que se denominan con las siglas AAA: *Arrange, Act, Assert*, es decir, preparar, actuar y afirmar. Esto hace que en muchos test la fase de *Arrange* sea la misma y los frameworks nos proporcionan los mecanismos para no tener que reescribir el mismo estado por cada test. Un ejemplo es el método *setUp(),* de la clase *UserModelTest* que hereda de *CUInit_TestCase*. Nos permite ejecutar cualquier sentencia con anterioridad e independencia entre nuestros tests. Respecto a la parte de *Act*, se refiere a la ejecución de nuestro método a implementar. Y por ultimo en la parte de *assert*, a través de los métodos proporcionados por el framework aseguramos que el resultado de la fase de actuación es el esperado.

A la hora de testear un método debemos asegurarnos que estamos testeando una única responsabilidad. Por ello los métodos deberán tener una única responsabilidad y si hay más deberemos separar cada responsabilidad en un método nuevo. Hasta este punto hemos estado

testeando, **validación de estado**. Pero en determinados casos no es posible y es necesario realizar **validación de iteración**. Un ejemplo de validación de iteración es cuando nos encontramos con elementos que modifican el estado del sistema, como por ejemplo las clases que acceden a bases de datos.

Los test de validación tienen como objetivo evaluar la interacción entre las partes sin que se llegue a realizar el acceso a la base de datos. Es entonces cuando utilizamos objetos *mocks* que nos proporciona el framework. La filosofía de estos métodos es proporcionar una interfaz que define todos los métodos que implementa nuestra clase de acceso a BD, de modo que en la validación verificaremos la iteración de nuestros métodos con los distintos métodos que acceden a la BD pero llamando a los métodos que define la interfaz del objeto *mock* (es decir sin acceder realmente a la BD, simplemente verificando que se efectúa la llamada [10]).

Tras esta pequeña introducción a los test unitarios hemos visto un claro ejemplo de test unitario de estado ejecutado durante esta iteración junto con otros test de estado. Al continuar con los test unitarios de la clase *User*, nos encontramos con el método *Login* que pasa a utilizar métodos de acceso a BD y deberemos testar su iteración. La complejidad de los test de iteración, añadida al desconocimiento de los frameworks de test y la escasa documentación encontrada hace que la duración de la iteración se vea comprometida y debemos tomar una decisión. En este punto se decide finalizar la iteración con las siguientes tareas y poder entregar el resultado funcional y no testado que tenemos hasta el momento, perdiendo calidad pero asegurándonos cierto margen de entrega.

Finalizada la iteración en un pequeño ejercicio de análisis, revisión del sprint, se decide seguir adelante con la 2º iteración dando prioridad a las historias de usuario que componen la siguiente iteración frente a la historias de usuario B4.

Como ya hemos adelantado en párrafos anteriores la planificación de esta iteración, aun siendo estimada para dos semanas se ha alargado la duración de la misma en un 100% debido a la falta de conocimientos en TDD y la formación necesaria.

Iteración 2

El resultado de esta iteración planificada para una sola semana ya debería darnos como resultado una versión Alpha de la aplicación sobre la que el cliente ya puede ir viendo el futuro de la misma y si nos estamos desviando del objetivo final.

Backlog	Tarea	Tipo	Descripción	Responsable	Estimación
A3, A4, A6, A7	Tarea 1	Análisis y diseño	Modelo de datos	J.M	30%
A3	Tarea 2	Implementación	Crear clases, métodos MVC	J.M	20%
A4	Tarea 3	Implementación	Creación de formularios y métodos	J.M	10%
A6	Tarea 4	Implementación	Crear clases y métodos MVC Task	J.M	20%
A7	Tarea 5	Implementación	Crear clases y métodos MVC Timer	J.M	20%

En esta iteración empezamos con el análisis y diseño del modelo de datos que deberemos ampliar respecto al creado en la iteración anterior. Estas relaciones afectan también a nuestros objetos de tipo Modelo (clase de tipo modelo siguiendo el patrón MVC, implementado por los frameworks que hemos seleccionado). En ellas se reflejan todas las relaciones entre entidades en ambos sentidos y en este caso se han configurado para disponer de los objetos relacionados automáticamente. Es decir, al cargar los datos de un registro en un objeto, en la misma instrucción estamos cargando los datos de los registros relacionados en la propiedad correspondiente. Un caso real dentro de nuestra implementación son los objetos *User_story* y *Task*. Ambos están relacionados y esto se refleja en la propiedades al definir cada clase. Ahora al cargar una historia de usuario (User_story) en la misma sentencia estamos cargando los datos de las tareas (Task) relacionadas en la propiedad *task* del objeto *User_story*

Por ahora el modelo de datos es sencillo y el número de usuarios está controlado, de modo que no afectará al rendimiento de la aplicación de un modo drástico.

En la 2º tarea así como en otras tareas similares, que implementan la funcionalidad necesaria para enviar datos del usuario a la aplicación, se ha utilizado validación de datos en dos sentidos, es decir, en el lado del cliente y en el del servidor. En el lado del servidor utilizamos la validación de *HTML5* y en el servidor los métodos proporcionados por el *ORM*.

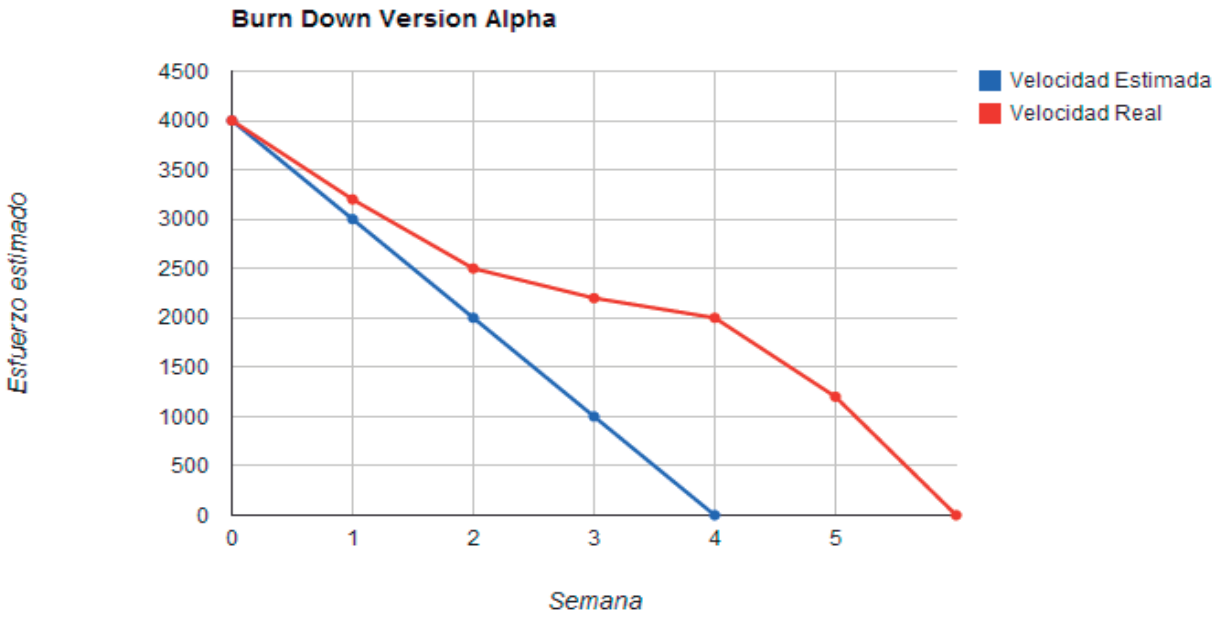
En la reunión de revisión del Sprint nos encontramos con una aplicación centrada en las historias de usuario, las tareas y el control de tiempos. Pero se ha perdido el foco de la

metodología, que la propia aplicación nos invite a seguir los pasos básicos que conlleva utilizar Scrum. Esto puede ser causa de una mala identificación y definición de las historias de usuario. Revisaremos los criterios de aceptación que darán lugar a un iteración intermedia; antes de proseguir con las siguientes historias de usuario debemos garantizar que están finalizadas las historias ejecutadas hasta el momento.

Criterios de validación:

- Los dos primeros requisitos de la historia A2 (eliminar/finalizar proyectos) se cumplen, pero en el segundo se puede ser más específico. O incluso añadir un nuevo requisito, al intentar eliminar/dar de baja un proyecto que no está finalizado o con datos deberá mostrar un mensaje de error al usuario. Además el tercer criterio no se cumple y tampoco se ha definido en qué consiste la finalización de un proyecto.
- El primer criterio de la historia A7 (procesar tareas) es erróneo. Aunque haya otros miembros asignados se pueden seguir asignando miembros. Además en el segundo criterio debería concretar qué datos son los que se reflejan.
- Falta un criterio de aceptación que verifique el resultado de cambiar el estado de una tarea.
- Los criterios de la tarea B4 (definir y crear una interfaz gráfica) no son válidos. Esto ha hecho que en la iteración 1 no se detecten errores y en este momento sí. Un criterio válido puede ser: La aplicación deberá permitir conocer la situación del usuario en todo momento dentro de la plataforma de forma inequívoca y por varios métodos.

Antes de comenzar con la iteración de revisión, se ha incluido un gráfico *Burn-down* para analizar el esfuerzo realizado durante la ejecución de las primeras iteraciones.



En el gráfico anterior podemos ver cómo la iteración 1 tiene un gran descenso de los puntos de esfuerzo ejecutados. Esto refleja el retraso que tiene la iteración durante la ejecución de las tareas 3, 4 y 6 de dicha iteración. Una vez eliminado el causante del retraso la velocidad se recupera e incluso supera la velocidad estimada.

Iteración 2B

En la reunión de preparación del Sprint se prepara el *Sprint Backlog* y se concreta el funcionamiento de las tareas a ejecutar.

Backlog	Tarea	Tipo	Descripción	Responsable	Estimación
B4	Tarea 1	Implementación	Aplicar las correcciones visuales en la estructura de las páginas.	J.M	15%
A2	Tarea 2	Análisis y diseño	Definir los estados de un proyecto	J.M	10%
A2, A7	Tarea 3	Diseño	Redefinir el modelo de datos	J.M	10%
A2	Tarea 4	Implementación	Implementar los estados de un proyecto	J.M	15%
A2	Tarea 5	Implementación	Implementar acciones automáticas según estado de un proyecto	J.M	25%
A7	Tarea 6	Implementación	Limitar permisos de modificación de tareas durante su ejecución	J.M	25%

Se han detallado las tareas resultantes de analizar los criterios de aceptación correspondiente a la iteración 1 y 2 aunque lo correcto hubiera sido revisar los criterios de la iteración 1 antes de comenzar con la iteración, si no son validos no se debería dar por terminada la iteración y en consecuencia no debemos empezar una iteración nueva.

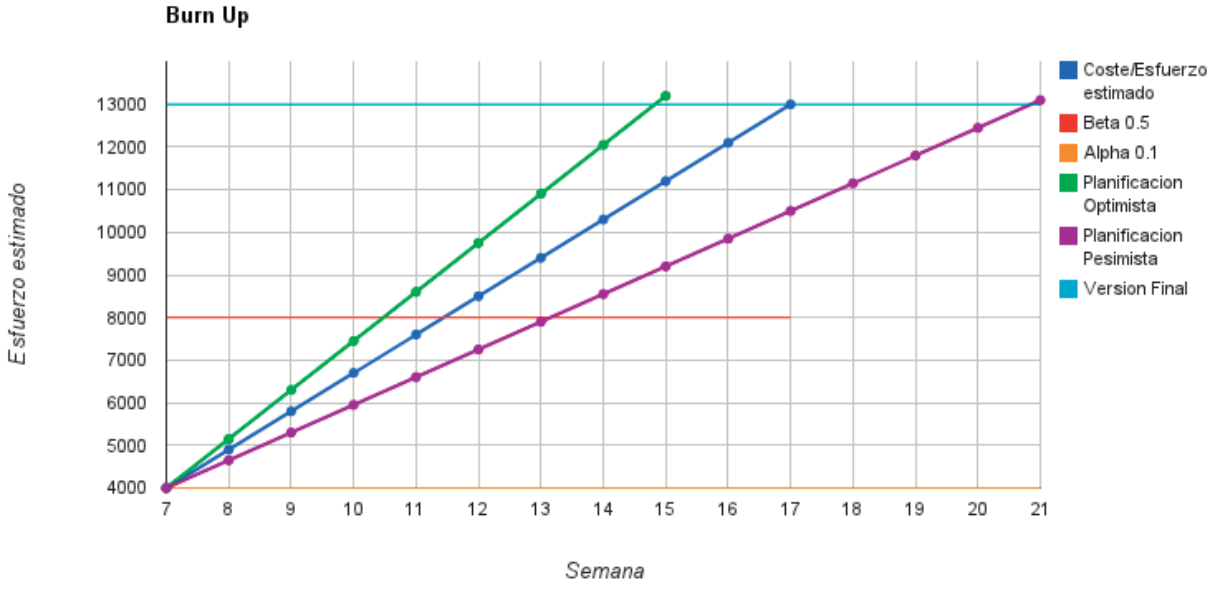
En este punto hemos alcanzado la situación actual del proyecto. Esta iteración continuará cuando terminamos la documentación del proyecto y el resultado será reflejado en la aplicación resultante al finalizar la iteración.

Revisión Product Backlog

Antes de proseguir con las iteraciones se ha realizado una actualización del *Product Backlog* con el fin de mejorar los objetivos de negocio y su ejecución.

Una vez eliminadas las historias de usuarios finalizadas se han reescrito algunas historias de usuario y sus criterios de validación. Con el objetivo de centrarnos únicamente en la siguiente iteración solo se han dado prioridad objetiva a las 4 siguientes historias. La siguiente iteración estará formada por las tareas A8 (acceso director, proyectos), A5 (acceso cliente, historias de usuario) y B3 (dar de baja usuarios) que juntas hacen un esfuerzo estimado de 900. El hecho de no llegar a los mil punto se debe que por ahora no tenemos una velocidad estable, en las anteriores iteraciones no se ha mantenido el ritmo y no se ha llegado a una velocidad de 1000 puntos semana.

En el siguiente *Burn-up* podemos ver reflejado el cambio de estimación, planificando a una velocidad media de 900 puntos por semana y con un margen de error de ± 250 puntos de esfuerzo. Debido a que nuestra velocidad real alcanzada ha tenido muchos altibajos como podemos observar en el gráfico *Burn-down*[17] no sirve base real para calcular la velocidad media que se acerca a los 700 puntos por semana. Superados los problemas que nos han frenados estas primeras iteraciones se espera aumentar la velocidad pero teniendo en cuenta que 1000 puntos por semana era una estimación optimista.



Este grafico Burn-up, actualiza la anterior planificación[18] realizada a 1000 puntos por semana. En esta planificación partimos de las 7ª semana de ejecución habiendo conseguido una ejecución de 4000 puntos de esfuerzo.

Replanificando observamos cómo la fecha idea para la versión beta se ha pasado hasta la mitad de la semana 11, mientras que al principio estaba planificada para la semana 8. Sin embargo cabe destacar que la iteración pesimista inicial coincide con la nueva planificación. Dicho esto podemos observar como aun tenemos otras dos líneas de planificación pesimista y optimista. Puesto que se trata de nuestros inicios en la metodología no tenemos suficiente información para poder disminuir el margen de error. En este caso se ha reducido en 50 puntos con el objetivo de acotar la planificación y ser más explícitos. Aunque tenemos en cuenta que el error anterior ha sido importante y tampoco podemos pasarlo por alto.

En la siguiente tabla observamos la actualización de historias de usuario y sus criterios de aceptación. Puesto que hemos priorizado únicamente las 4 siguientes historias, una vez esté finalizada la iteración 3, será necesaria otra actualización del *Product Backlog* y obligatoriamente priorizar las siguientes historias. De este modo seremos más conscientes de los posibles cambios y las necesidades antes de seguir con el desarrollo, y podemos tener en cuenta los posibles errores o aciertos que hemos ido teniendo a lo largo del proyecto.

Las historias de usuario no ejecutadas son el reflejo del **trabajo futuro** que puede tener este proyecto. Funcionalidades como la generación de gráficos automáticamente en base al trabajo será de gran aporte al proyecto, esto se refleja en el *Product Backlog*. Junto a esta se han quedado a la espera otras historias de usuario que aportan el valor de la metodología a la aplicación. Teniendo esto como principal meta, también hay que mencionar la necesidad de crear una aplicación de calidad y esto implicara documentarse lo necesario para realizar los test unitarios para testear el funcionamiento de lo que tenemos hasta el momento y continuar el desarrollo con TDD.

Identificador Objetivo	Descripcion	Coste estimado	Condiciones de satisfacción	Valor aportado	Observaciones	Prioridad
A5	Acceso cliente, Historias Usuario	300	Al acceder a una historia de usuario puede modificar el valor aportado debiendo modificar asi la prioridad de la misma.	500		12
			Conocer el estado de las historias, por empezar, en ejecución o finalizada.			
A8	Acceso Director, Proyectos	300	Al acceder al proyecto se le da acceso a toda la información de historias de usuario y tareas que están relacionadas.	600		11
			Exportar informes con los datos actuales.		Se concretarán los formatos al planificar la iteración.	
A9	Representacion visual de los datos	1200	Puedo ver el Gráfico Burn-Down en base a los datos de cada proyecto.	500		
			Puedo ver el Gráfico Burn-UP en base a los datos de cada proyecto.			
A10	Añadir documentación Externa	200	Puedo descargar documentos adjuntos a un proyecto.	100		
			Puedo añadir documentos adjuntos a un proyecto	100		
A11	Control de versiones para archivos	2000	Puedo conocer el historial de un documento, cuando y quien lo ha modificado.	200		
			Al subir un documento modificado por mi, el sistema me avisará si hay una notificación posterior a mi ultima descarga.			
A12	Representacion visual de los elementos Scrum (Kanban)	1300	Puedo consultar la información de proyectos y sus elementos de forma visual.	400		
A13	Asignación de permisos entre roles de usuario	1500	El administrador de la plataforma puede asignar permisos para una funcionalidad concreta	100		
A14	Como usuario de la aplicación quiero que la metodología se vea implícita en las acciones que me ofrece.	1200		300	La aplicación debe ayudar a seguir la metodologia. Se concrerán más detalles.	
A15	Como scrum master quiero poder guardar información relativa a las reuniones.	1000	Permitirá guardar la informacion referente a la reunion que da comienzo un sprint o iteración. Reunion de planificacion	300		
			Permitirá guardar la informacion referente a la reunion que revisa un sprint o iteración. Reunion de revisión.			
A16	Como miembro del equipo quiero poder consultar el sprint backlog	300	Un usuario puede consultar las historias de usuario que corresponden con la iteracion en curso del proyecto.	300		14
			Un usuario puede consultar las tareas que corresponden con la iteración en curso del proyecto.			
Otras historias						
B3	Como administrador de la aplicación quiero dar de baja usuarios	300	Al dar de baja un usuario deja de estar disponible su acceso	400		13
			Si es un miembro del equipo dejará de estar disponible para ser asignado con tareas u historias de usuario.			
Total		**9600**				

Product Backlog Versión 2

Conclusiones

En esta última sección de la memoria, vamos a detallar las conclusiones que hemos obtenido de la ejecución del TFG.

El objetivo marcado era crear una aplicación para la gestión ágil de proyectos dirigidos utilizando **Scrum**. Para ello nos hemos adentrado en el mundo de las tecnologías ágiles concretando el funcionamiento de Scrum que para entenderlo y poder reflejarlo mejor se ha utilizado como metodología para dirigir y ejecutar nuestro proyecto.

Es cierto la aplicación se ha quedado en una versión Alpha, que no refleja al 100% nuestro objetivo final por las dificultades encontradas durante la ejecución del proyecto. Por otro lado hemos obtenido una formación y visión de las metodologías ágiles entendiendo los valores que defiende y sus indicaciones, gracias a la puesta en marcha de las mismas en un proyecto real.

Estas dificultades han venido dadas por la necesidad de documentación previa en la metodologías y las áreas que se abren dentro de un proyecto ágil, entre ellas el desarrollo guiado por test, **TDD**. Entendidos los principios y las ventajas de esta "técnica" de desarrollo apenas hemos podido poner en marcha y ver las ventajas en nuestro propio proyecto. Los test que hemos podido poner en práctica nos garantizan que nuestra programación testada será de mayor calidad y en caso de existir un error podremos excluir esa parte como la causa del mismo.

El hecho de utilizar metodologías ágiles no implica la falta de un método, sino que simplemente se rigen por unos principios más flexibles y orientados a la situación real de algunos proyectos que están expuestos a un cambio constante y/que en su fase inicial no están definidos ni se conocen las posibles dificultades que podemos encontrarnos. Además de esto, en todo momento se tiene en cuenta la calidad de lo que hacemos y entiende que estamos realizando un trabajo artesanal que puede tener errores. Por ello incluye como técnica de desarrollo TDD.

Respecto a Scrum y los pasos que nos indica son la base para una ejecución ágil, si no tenemos experiencia en la dirección de proyectos son el mejor punto de partida que posteriormente podremos adaptar a nuestro entorno y experiencia. La herramientas que nos proporciona pueden resultar un poco abstractas y complejas. En particular la descomposición de los objetivos en historias de usuario totalmente independientes y con un lenguaje coloquial, compartido por el *Product Owner* y el equipo. La descomposición de las historias en tareas independientes nos ayuda a organizar el trabajo en equipo y aunque no ha sido aplicable en nuestro entorno, la descomposición en tareas ha resultado un aspecto bastante abstracto. Viniendo de las metodologías tradicionales en las que no se hace referencia a estos conceptos,

es complicado descomponer las funcionalidades en tareas totalmente independientes sin una experiencia previa, de modo que nos hemos basado en la documentación previa a partir de la cual tomar nuestras decisiones.

Un ejemplo puede ser la creación del modelo de datos. En nuestro *Product Backlog* no hay una historia de usuario que refleje la creación del modelo de datos. En un principio hemos llevado al extremo la independencia de las historias y en cada una se define el modelo de datos para las funcionalidades que indica. En la segunda iteración se decide crear un modelo de datos que refleje todas las historias de usuario de la iteración. Finamente después de esta experiencia empezaríamos un proyecto con una historia de usuario para el modelo de datos general. En ella deberíamos identificar las entidades y relacionarlas entre sí. Esto se complementaría con una tarea específica que identifique las propiedades de las entidades implicadas en la iteración que estamos ejecutando.

Otra característica de Scrum y que inicialmente hemos dejado de lado son las reuniones. Al no ser un proyecto en equipo, o el equipo estar compuesto por un solo miembro y un director tutor, que ha compartido el rol de Product Owner con el alumno, las reuniones sin el tutor no tenían lugar como tales. Por ello en la segunda iteración se concreta la reunión de revisión en la que sí se han revisado los criterios expuestos en el Product Backlog y posteriormente se han tomado las medidas oportunas para cumplir los criterios. Las revisiones de sprint sirven de utilidad para conocer el estado del proyecto día a día. En nuestro caso se reflejan aún más sus ventajas puesto que la dedicación en este proyecto no es 100% día a día y se hace más necesaria una revisión y planificación de lo que hemos hecho y lo que nos queda por hacer.

Para terminar destacar que la experiencia ha sido satisfactoria, sacando en claro muchos aspectos de las metodologías agiles y el porqué plantearse utilizarlas al inicio de un proyecto. En este caso debido a la naturaleza de nuestro proyecto, dirigir proyectos ágiles, y al desconocimiento de área que aborda, era de vital importancia conocer desde nuestro punto de vista la metodología. Esto ha propiciado que la planificación no sea lo más acertada y nos hayamos desviado, no llegando al esfuerzo estimado medio que inicialmente se planificó. Aun así, sí que estamos dentro del margen definido en aquella primera planificación, haciendo así que seamos conscientes de las complicaciones que pueden surgir durante la ejecución del proyecto. Esto nos da un claro ejemplo de lo importantes y útiles que resultan los elementos que nos ofrecen Scrum.