Trabajo de Fin de Grado

Anexo I

**Plan de proyecto Software.**

Texto

Descripción generada automáticamente

**Alumno**

Álvaro López Marcos

**Tutores**

André Filipe Sales Mendes

Gabriel Villarrubia González

Índice de contenidos

[Índice de Ilustraciones 2](#_Toc144916444)

[Índice de Tablas 2](#_Toc144916445)

[1. Introducción 3](#_Toc144916446)

[2. Estimación de Costes 3](#_Toc144916447)

[2.1. Complejidad Actores 3](#_Toc144916448)

[2.2. Complejidad Casos de Uso 4](#_Toc144916449)

[2.3. Complejidad Factores 5](#_Toc144916450)

[3. Planificación 7](#_Toc144916451)

[3.1. Metodología Ágil 7](#_Toc144916452)

[3.2. SCRUM 8](#_Toc144916453)

[3.3 Calendario de tareas 9](#_Toc144916454)

[Bibliografía 12](#_Toc144916455)

# Índice de Ilustraciones

[Ilustración 1. Agile 7](file:///C:\Users\alvie\Documents\GitHub\TrabajoFinDeGrado\Anexo%20I.docx#_Toc112860716)

[Ilustración 2. Scrum 8](file:///C:\Users\alvie\Documents\GitHub\TrabajoFinDeGrado\Anexo%20I.docx#_Toc112860717)

# Índice de Tablas

[Tabla 1. Razones Complejidad Actores 3](#_Toc144916458)

[Tabla 2. Pesos Actores 3](#_Toc144916459)

[Tabla 3. Razones Casos de Uso 4](#_Toc144916460)

[Tabla 4. Pesos Casos de Uso 4](#_Toc144916461)

[Tabla 5. Factores Complejidad Técnica 5](#_Toc144916462)

[Tabla 6. Factores Complejidad Entorno 5](#_Toc144916463)

# 1. Introducción

En este documento anexo a la memoria final del Trabajo Fin de Grado se detallan los requisitos funcionales, no funcionales y de información, casos de uso y sus diagramas y otros requisitos del sistema siguiendo el método de Durán y Bernárdez.

# 2. Estimación de Costes

Para una correcta estimación de los costes del proyecto a desarrollar debemos remitirnos a la estimación de esfuerzo. Existen múltiples métricas para realizar dicha estimación, en nuestro caso, utilizaremos una métrica de estimación de esfuerzo por tamaño, su nombre es “Use Case Points” (UCP), fue desarrollada por Gustav Karner en 1993 y es ideal para nuestro proyecto, ya que usamos herramientas del lenguaje unificado.

E cálculo de UCP se realiza teniendo en cuenta los siguientes pasos:

* Cálculo del peso no ajustado del actor (Unadjusted Actor Weight, UAW), basado en los tipos de interacción del usuario con el sistema (simple, medio, complejo).
* Cálculo del peso del caso de uso (Unadjusted Use Case Weight, UUCW), basado en el número de interacciones del caso de uso con el sistema (simple, medio, complejo).
* Cálculo de los puntos de caso de uso no ajustados (Unadjusted Use Case Point, UUCP), obtenido mediante la adición de los dos pesos anteriores.
* Cálculo de los puntos de caso de uso ajustados (Adjusted Use Case Point, AUCP), obtenido mediante la multiplicación de UUCP por el factor de complejidad técnica (Technical Complexity Factor, TCF) y el factor de complejidad del entorno (Environmental Complexity Factor, ECF).

### Complejidad Actores

Transaccion: conjunto de acciones que ocurren desde que el usuario inicia una petición hasta que el sistema queda pendiente de la siguiente petición.

|  |  |  |
| --- | --- | --- |
| **ACTORES** | **RAZÓN** | **PESO** |
| **Simple** | Sistema que se comunica con la aplicación mediante API | 1 |
| **Medio** | Sistema que se comunica con la aplicación mediante protocolo web | 2 |
| **Complejo** | Persona que se comunica con la aplicación mediante una interfaz | 3 |

Tabla . Razones Complejidad Actores

|  |  |  |  |
| --- | --- | --- | --- |
| **UAW** | **ACTOR** | **CATEGORÍA** | **PESO** |
| Usuario registrado | Complejo | 3 |
| Usuario no registrado | Complejo | 3 |
| **Total** | | | 9 |

Tabla . Pesos Actores

### Complejidad Casos de Uso

Transacción: conjunto de acciones que ocurren desde que el usuario inicia una petición hasta que el sistema queda pendiente de la siguiente petición.

|  |  |  |
| --- | --- | --- |
| **CASOS DE USO** | **RAZÓN** | **PESO** |
| **Simple** | El número de transacciones es igual o menor a 3 | 1 |
| **Medio** | El número de transacciones mayor que 3, pero igual o menor a 7 | 2 |
| **Complejo** | El número de transacciones es mayor que 7 | 3 |

Tabla . Razones Casos de Uso

|  |  |  |
| --- | --- | --- |
| **UUCW** | **CATEGORÍA** | **PESO** |
| UC-0001 | Simple | 1 |
| UC-0002 | Medio | 2 |
| UC-0003 | Medio | 2 |
| UC-0004 | Simple | 1 |
| UC-0005 | Simple | 1 |
| UC-0006 | Simple | 1 |
| UC-0007 | Medio | 2 |
| UC-0008 | Medio | 2 |
| UC-0009 | Simple | 1 |
| UC-0010 | Medio | 2 |
| UC-0011 | Simple | 1 |
| UC-0012 | Simple | 1 |
| UC-0013 | Medio | 2 |
| UC-0014 | Simple | 1 |
| UC-0015 | Medio | 2 |
| UC-0016 | Simple | 1 |
| UC-0017 | Medio | 2 |
| UC-0018 | Simple | 1 |
| UC-0019 | Medio | 2 |
| UC-0020 | Simple | 1 |
| UC-0021 | Simple | 1 |
| UC-0022 | Medio | 2 |
| UC-0023 | Simple | 1 |
| UC-0024 | Medio | 2 |
| **Total** | | 35 |

Tabla . Pesos Casos de Uso

Con el valor del peso no ajustado del actor y el peso del caso de uso, obtenemos los puntos de caso de uso no ajustado.

*UUCP = UAW + UUCW*

*UUCP = 35 +9 = 44*

### Complejidad Factores

#### 2.2.1. Factores Complejidad Técnica

*TCF = 0,6 + (0,01 \* TFactor)*

*TCF = 0,6 + (0,01 \* 14) = 0,74*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **NOMBRE** | **PESO** | **VALOR** | **FACTOR** |
| **T1** | Distribución | 2 | 1 | 2 |
| **T2** | Eficiencia | 2 | 1 | 2 |
| **T3** | Portabilidad | 1 | 1 | 1 |
| **T4** | Concurrencia | 1 | 2 | 2 |
| **T5** | Balance de carga | 2 | 1 | 2 |
| **T6** | Adaptación al cambio | 1 | 1 | 1 |
| **T7** | Seguridad | 1 | 2 | 2 |
| **T8** | Facilidad de uso | 1 | 2 | 2 |
| **TFactor** | | | | 14 |

Tabla . Factores Complejidad Técnica

#### 2.2.2. Factores Complejidad Entorno

*ECF = 1,4 + (-0,03 \* 20)*

*ECF = 1,4 + (-0,03 \* 20) =* *0,8*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **NOMBRE** | **PESO** | **VALOR** | **FACTOR** |
| **T1** | UML | 1 | 2 | 2 |
| **T2** | Análisis | 1 | 2 | 2 |
| **T3** | IDE | 2 | 2 | 4 |
| **T4** | Lenguajes | 1 | 2 | 2 |
| **T5** | Requisitos | 2 | 3 | 6 |
| **T6** | Frameworks | 2 | 2 | 4 |
| **EFactor** | | | | 20 |

Tabla . Factores Complejidad Entorno

*UCP = 44 \* 0,80\* 0,74*

*UCP = 26*

*TPH = 20 \* UCP*

*TPH = 20 \* 26= 520 horas/UCP*

# Planificación

A la hora de desarrollar software es muy importante estructurar el trabajo a lo largo del tiempo estipulado para la realización del proyecto, por lo que la planificación de las tareas cobra una gran importancia.

## 3.1. Metodología Ágil

La metodología de trabajo ágil nace en 2001 del Manifiesto Ágil. Una innovadora forma de trabajo flexible que se basa en 12 principios:

* Satisfacer a los clientes a través de la entrega temprana y continua.
* Los cambios siempre son bienvenidos, incluso al final del proyecto.
* Entrega de valor con frecuencia.
* Los empresarios y desarrolladores deben romper con los roles preestablecidos y trabajar de forma continua.
* Construir proyectos entorno a equipos competentes y motivados, no a la inversa.
* La eficacia de la comunicación está en el “cara a cara”.
* Mantener un ritmo de trabajo sostenible en el tiempo.
* La excelencia continua mejora la agilidad.
* La simplicidad es esencial.
* Equipos autoorganizados generan más valor.
* Reflexiona y ajusta regularmente tu forma de trabajar para aumentar la eficacia.

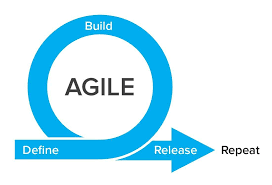
Dentro de todas las metodologías ágiles existentes la utilizada en este proyecto ha sido SCRUM.

Ilustración . Agile

## SCRUM

Scrum es una metodología de gestión de proyectos ágil que los equipos de trabajo usan para desarrollar, entregar y mantener productos complejos. Scrum está orientado a la entrega de valor continua al cliente. Es una metodología que ayuda a comenzar a pensar con una mentalidad propia de las metodologías ágiles y empezar a construir los principios ágiles en nuestra manera de trabajar.

Scrum se usa en múltiples ámbitos, recursos humanos, marketing, diseño… Pero donde más popular es esta metodología es en el desarrollo e ingeniería del software.

El software es un producto que no para de evolucionar. Los requisitos son cambiantes, las metas cambian, las situaciones dentro del equipo de trabajo cambian. Scrum abraza el cambio, lo acepta y trabaja con él.

Con scrum, un producto es construido en varias series de repeticiones llamadas sprints. Deshaciendo proyectos complejos en piezas mucho más pequeñas, con una complejidad reducida. Esto permite que los proyectos sean más manejables y que los equipos alcancen una versión de buena calidad antes y con mayor frecuencia, dándole al proyecto una mayor flexibilidad de cara a los cambios. Que scrum sea un marco de trabajo iterativo también ayuda a abordar problemas que frecuentemente algunos equipos experimentan en el flujo de trabajo en cascada. Iteraciones de corta duración reducen costes y aportan retroalimentación por parte de los usuarios dando la sensación al equipo de que se está logrando progresar.

Los roles dentro de un equipo Scrum son:

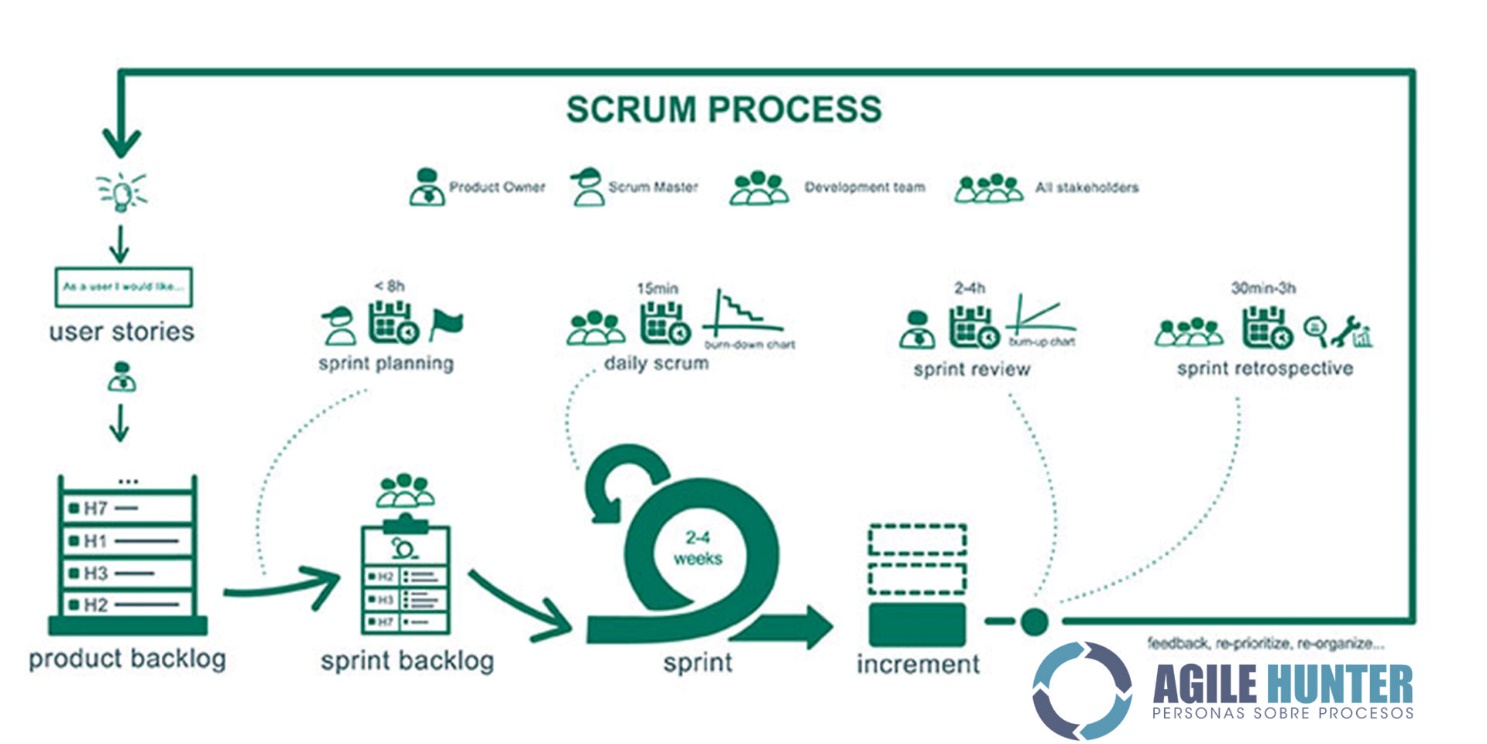
* **Product Owner:** Encargado de gestionar el trabajo buscando el incremento de valor del producto, teniendo en cuenta el valor del trabajo que el equipo realiza. Este rol sería adoptado por el alumno. Dado que conoce y fija los objetivos que dan valor al producto.
* **Scrum Master:**  Supervisa y gestiona el trabajo del equipo, aportando soluciones a los problemas que se puedan generar en el desarrollo. Los tutores del trabajo tomarán este rol.
* **Equipo de Desarrollo:** equipo que trabaja en el incremento de valor del producto y hace sus entregas. De nuevo el alumno adoptará este rol.

Ilustración . Scrum

## 3.3 Calendario de tareas

El proyecto se dividirá en 6 sprints, en primer lugar, se planifica la duración de las tareas, las prioridades del proyecto, estimación de costes y tiempo de trabajo. A esta fase le siguen dos de planificación, en las que desarrollaremos el diseño de nuestra aplicación, este diseño se desarrollará en los siguientes dos sprints y por último finalizaremos el proyecto con un Sprint de pruebas, en el que resolveremos las incidencias que encontremos a la hora de probar la funcionalidad de nuestra aplicación.

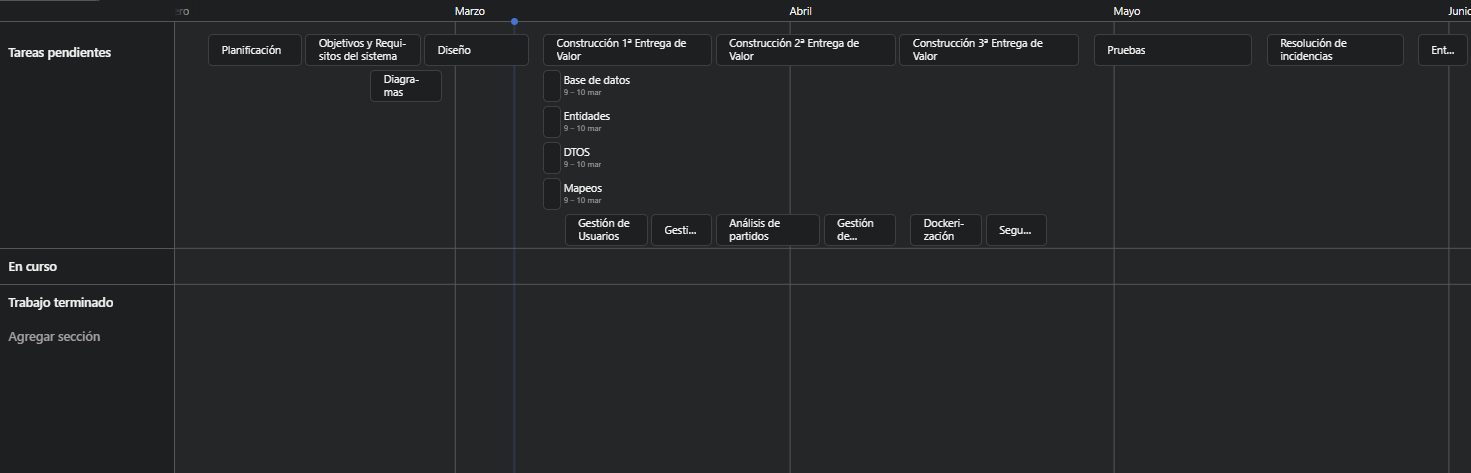


Ilustración . Estado del calendario al inicio del proyecto.