Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Gestión de proyectos en CRM Salesforce basada en metodología Agile

Autor: Ana Caballos Torroba
Tutor: David Canca Ortiz

Dpto. Organización Industrial y Gestión de Empresas I
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Gestión de proyectos en CRM Salesforce

Autor:
Ana Caballos Torroba

Tutor:
David Canca Ortiz
Profesor titular

Dpto. Organización Industrial y Gestión de Empresas I
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2020

Trabajo Fin de Grado: Gestión de proyectos en CRM Salesforce

Autor: Ana Caballos Torroba

Tutor: David Canca Ortiz

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2020

El Secretario del Tribunal

A mi familia y amigos por apoyarme y animarme en todo momento

A mis compañeros de trabajo por inspirarme y ayudarme en mi día a día

Finalmente, a mi tutor por la paciencia y dedicación en cada una de sus revisiones.

GRACIAS

Resumen

En la actualidad, es cada vez más común que las empresas recurran a algún software que les permita gestionar las relaciones con sus clientes, esto es lo que se conoce como CRM. Para ello surgen numerosas herramientas, como la plataforma Salesforce, en la que se va a centrar este documento, por ser una de las más demandadas en el mercado hoy en día.

Al ser una herramienta compleja y con numerosas posibilidades, las empresas suelen recurrir a equipos especializados que analicen las necesidades, configuren la plataforma y den una solución a éstas, de la manera más eficiente. Este documento se centrará en el papel de estos equipos, y de cómo debe ser la gestión de un proyecto de este tipo.

En concreto se analizará la gestión de un proyecto de Salesforce siguiendo una metodología ágil, basada en una mayor interacción entre los participantes y un análisis y revisión continuos de la solución que se va construyendo. Este análisis se hará tanto desde un punto de vista teórico, revisando las características y herramientas que se emplean en la gestión ágil, cómo desde un punto vista práctico, donde se revisará la aplicación de estas herramientas para el caso concreto de un proyecto en Salesforce.

Abstract

Nowadays, it is increasingly common for companies to use some software that allows them to manage relationships with their customers, this is what is known as CRM. For this reason, numerous tools have appeared, such as the Salesforce platform, on which this document will focus, because it is one of the most demanded in the market today.

Because it is a complex tool with many possibilities, companies often hire specialized teams to analyze the needs, configure the platform and provide a solution to them, in the most efficient way. This document will focus on the role of these teams, and how a project of this type should be managed.

Specifically, the management of a Salesforce project following an agile methodology will be analyzed. This agile methodology is based on a greater interaction between the participants and a continuous analysis and review of the solution that is being built. This analysis will be done both from a theoretical point of view, reviewing the characteristics and tools used in agile management, and from a practical point of view, where the application of these tools will be reviewed for the specific case of a project in Salesforce.

Índice

1 INTRODUCCIÓN

En la actualidad, con la proliferación de las nuevas tecnologías, cada vez más empresas emplean herramientas de software para la gestión de todos sus procesos, en especial, aquellos que tienen que ver con la relación y la gestión de sus clientes. Esto se traduce en un cambio progresivo en las expectativas de los clientes, que optan por aquellos proveedores que les ofrezcan servicios más competitivos y adaptados a sus necesidades. Con este objetivo han surgido herramientas como los CRM que centralizan todos los aspectos de esta gestión en un único lugar. Es por este motivo que, según un reciente estudio de Eurostat[1], actualmente en torno a un 20% de las empresas europeas utilizan este tipo de software [1].

Dentro del mundo de los CRM, una de las herramientas más utilizadas es Salesforce. Esto es porque además de ofrecer las ventajas de una plataforma en la nube, está en constante búsqueda de mejoras y actualizaciones, adquiriendo las últimas tecnologías que complementen las herramientas de las que disponen. Además, se presenta como un software seguro y estable con una gran cantidad de usuarios que lo avalan. Es por eso por lo que recientemente Salesforce ha vuelto a ser nombrado CRM número uno en el mercado por IDC (International Data Corporation) en su último estudio [2].

Salesforce es muy versátil y ofrece infinidad de opciones para implementar diferentes funcionalidades. Para poder encontrar la mejor solución para cada una de ellas, es necesario tener un conocimiento amplio de la herramienta, especialmente en grandes empresas con flujos muy complejos. Por este motivo, las empresas suelen externalizar este servicio y contratar otras empresas más especializadas que gestionen el software

A raíz de esto, surge un nuevo tipo de consultoría, la consultoría tecnológica especializada en Salesforce, a la que cada vez se suman más empresas. Un consultor Salesforce es una persona especializada en la herramienta que acompaña a las empresas en los proyectos para solucionar sus necesidades, desde la toma y definición de requerimientos y el análisis de la solución técnica, hasta la construcción e implantación final.

Estos proyectos por norma general se desarrollan bajo una metodología Agile, por lo que además de conocimientos en Salesforce, será necesario conocer esta metodología y las herramientas que utiliza.

Por este motivo, en este documento se analizarán los dos aspectos principales relacionados con este tipo de consultoría, uniendo los conocimientos sobre gestión Agile y el manejo de Salesforce.

1.1 Alcance y objetivos del documento

El objetivo principal de este documento es el de proporcionar una serie de nociones básicas sobre lo que implica la gestión de proyectos en Salesforce a aquellas personas que quieran introducirse en el mundo de la consultoría tecnológica de Salesforce o que simplemente quieran conocer un poco más sobre este tema.

Se presentarán los aspectos más relevantes que se necesitan para afrontar un proyecto de esta naturaleza. Se pretende proporcionar una visión de 360º de los aspectos más importantes, tanto en lo que al manejo y gestión de la herramienta se refiere, como a lo que implica un proyecto gestionado bajo una metodología Agile.

La motivación fundamental es la ausencia de bibliografía que resuma y una los dos mundos y que permita en un mismo documento conocer todo lo necesario para entender todos los aspectos involucrados desde el inicio al fin del proyecto, desde un punto de vista práctico.

Es importante entender que cuando se habla de practicidad no se refiere a la exposición de un caso de uso real, sino más bien analizar cada uno de los procesos implicados y qué se necesitan para poderlos llevar a cabo. Incluyendo las prácticas recomendadas tanto las recomendaciones generales que deben seguir todos los proyectos basados en Agile, como las buenas prácticas recomendadas desde la propia plataforma de Salesforce.

[1] Eurostat: Oficina Estadística de la Unión Europea

1.2 Estructura del documento

El documento consta de tres capítulos, los dos primeros se enfocan en el modelo de gestión ágil, en el primero se hace un análisis general y en el segundo se especifican las características de los marcos de trabajo más relevantes. El tercero es la aplicación de este modelo a la gestión de un proyecto en Salesforce desde un punto de vista práctico.

En el capítulo 2 se presentará en qué consiste el modelo de gestión Agile haciendo un recorrido por sus orígenes con el Manifiesto Agile y exponiendo los principios y valores en los que se basa. Se estudiará este modelo y las principales diferencias con respecto a los modelos tradicionales de gestión de proyectos, tanto en la estructura que sigue como en la gestión de equipos y personas, enumerando también sus principales ventajas e inconvenientes.

También en este capítulo se abordan las principales prácticas y herramientas que tienen en común todos los marcos de trabajo basados en esta filosofía, tanto en la etapa inicial de un proyecto, en la que se recogen las necesidades y se planifica su ejecución, como en las etapas posteriores de construcción y entrega para las que será necesario hacer un seguimiento.

En el capítulo 3 se hará hincapié en los principales marcos de trabajo Agile que se utilizan en la actualidad, que son Scrum, Kanban y Extreme Programming. Se explicará cada uno de ellos, sus características y las herramientas que emplean para poder implementar los principios y valores ágiles bajo una serie de prácticas.

La parte más práctica de este documento llega en el capítulo 4 donde se particulariza lo ya visto para un proyecto en Salesforce. En este capítulo se analizará qué metodología ágil es más recomendable y porqué, además se conocerán algunos aspectos propios de Salesforce que son fundamentales si se quiere conocer en profundidad lo que implica la gestión de proyectos en esta plataforma. Dentro de esta categoría incluimos algunas nociones técnicas básicas y la gestión de entornos.

2 GESTIÓN DE PROYECTOS ÁGIL

"Siempre es sabio mirar adelante, pero es difícil mirar más allá de lo que se puede ver"

- Winston Churchill -

La gestión ágil de proyectos o *Agile Project Management* agrupa un conjunto de metodologías y marcos de trabajo basadas en la filosofía **Agile**. Esta filosofía se fundamenta en una serie de principios y valores que están recogidas en el *Manifiesto Agile*, y que buscan un cambio en la mentalidad a la hora de gestionar y organizar los proyectos.

Este modelo de gestión está orientado principalmente al ámbito del software, aunque cada vez es más común su aplicación otras áreas. Su principal objetivo es dar a las empresas una mayor capacidad de adaptación ante las cambiantes exigencias del mercado, obteniendo así una mejora en el *time to market*[2]. Esto se consigue proponiendo un modelo más flexible y colaborativo, que permita revisar de forma continua las necesidades que se tienen en cada momento poniendo en el centro a las personas y sus capacidades como valor fundamental.

En este capítulo se explicará el origen y las bases en las que se fundamentan estas metodologías. A continuación, se estudiarán las características de estos modelos frente a los tradicionales y se analizarán sus ventajas e inconvenientes. Además, se conocerán las principales herramientas que se usan para llevar los proyectos desde su inicio, en la detección de una serie de necesidades, hasta la entrega del producto final.

2.1 Manifiesto Agile, origen y principios

El origen de las prácticas Agile se remonta a finales de la década de los 80 y principios de los 90, junto con la proliferación de las tecnologías informáticas en las empresas. Como ocurre en la actualidad, en esas fechas las empresas ya empezaban a encontrarse con la problemática de que los mercados avanzaban más rápido que los proyectos, especialmente cuando se trataba de proyectos de software. Por lo que en muchas ocasiones cuando se producía el lanzamiento de una solución, ésta ya estaba obsoleta y no satisfacía las necesidades de ese momento.

La frustración provocada por esta problemática hizo que empezaran a surgir alternativas a las metodologías tradicionales, con en plazos de entregas más cortos y una mayor interacción con los clientes. Entre ellas destacan la aparición de Scrum en 1986 [3] y Extreme Programming en 1999 [4].

Pero no fue hasta febrero del 2001, en la conferencia "*Light Weight Methods Conference*" en el Snowbird Resort en Utah, cuando se instauró el término "Agile", ya que hasta entonces era más común el término "light".

[2] *Time to market*: plazo de tiempo que transcurre desde su concepción hasta el lanzamiento de un producto

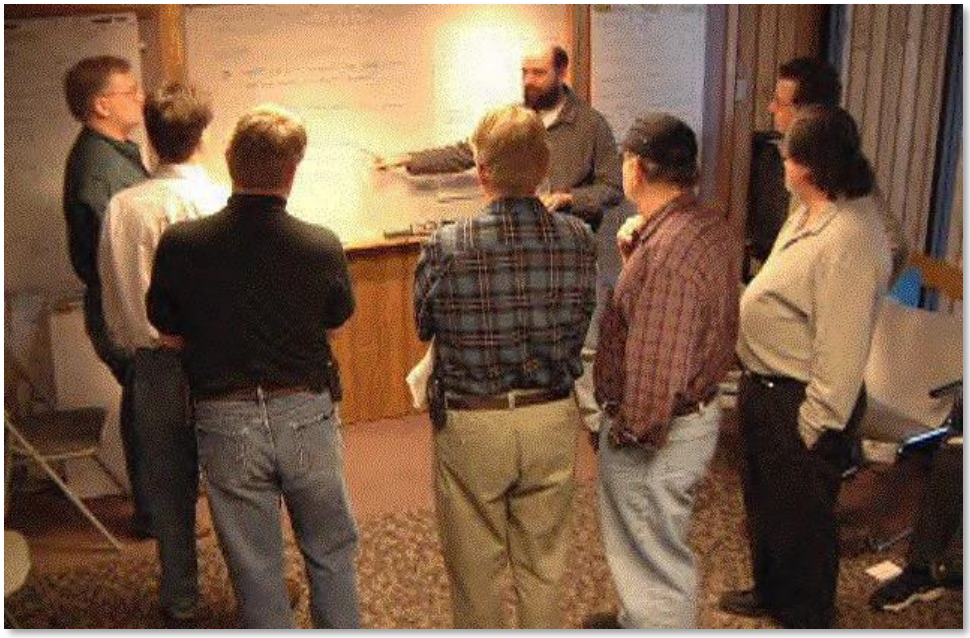

Figura 2.1 – "*Light Weight Methods Conference*", febrero 2001, Utah (Fuente: [5])

En esta reunión 17 especialistas, se reunieron para crear el "**Manifiesto Agile**", donde se firmaron 4 valores y 12 principios [6] que se enumeran a continuación.

Valores:

- Valorar a los **individuos e interacciones** sobre procesos y herramientas
- Valorar al **software en funcionamiento** sobre documentación extensiva
- Valorar la **colaboración con el cliente** sobre negociación contractual
- Valorar la **respuesta ante el cambio** sobre seguir un plan

Figura 2.2 – Valores Agile (Fuente: [7])

Principios:

1. Nuestra mayor prioridad es **satisfacer al cliente** mediante la entrega temprana y continua de software con valor.
2. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles **aprovechan el cambio** para proporcionar ventaja competitiva al cliente.
3. **Entregamos software** funcional frecuentemente, entre dos semanas y dos meses, con preferencia **al periodo de tiempo más corto posible**.
4. **Los responsables de negocio y los desarrolladores trabajamos juntos** de forma cotidiana durante todo el proyecto.
5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y **confiarles la ejecución del trabajo**.
6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la **conversación cara a cara**.
7. El **software en funcionamiento es la medida principal** de progreso.
8. Los procesos Ágiles promueven el **desarrollo sostenible**. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
9. La atención continua a la **excelencia técnica y al buen diseño** mejora la Agilidad.
10. La **simplicidad**, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen **de equipos autoorganizados**.
12. A intervalos regulares **el equipo reflexiona** sobre cómo ser más efectivo **para** a continuación **ajustar y perfeccionar su comportamiento** en consecuencia.

Figura 2.3 – Principios Agile (Fuente: [8])

En conclusión, el objetivo de estas metodologías debe ser poner el foco en el usuario, para así poder entregar un producto de calidad que se ajuste a las necesidades de éste en cada momento. Para ello se utiliza el propio software como herramienta sobre la que ir implantando mejoras y cambios para ir definiendo la solución final.

Desde ese momento se han ido desarrollando y mejorando este tipo de metodologías permitiendo la adaptación a las necesidades de cada momento. Pudiendo aplicarse, no únicamente a proyectos de software, sino a muchos otros ámbitos donde también se necesita flexibilidad y capacidades de readaptación a la hora de implementar soluciones.

2.2 Modelo Agile frente al tradicional

En universo Agile, hablar de gestión de proyectos puede ser bastante confuso, ya que esta filosofía rompe con los conceptos relativos a esta cuestión que se han manejado en los modelos tradicionales también conocidos como modelos *Waterfall*. Por ello, primero se expondrá qué es y qué significa gestionar un proyecto y luego se explicará que implica esto desde el punto de vista ágil frente al tradicional.

Un proyecto se puede definir como un conjunto de actividades que se deben realizar para llegar a un objetivo concreto dentro de un plazo determinado. Actualmente la mayoría de los procesos de cambio en las empresas, especialmente los que a software se refieren, se llevan a cabo mediante proyectos, por lo tanto, es importante tener claros ciertos conceptos antes de comenzar uno.

Según la RAE[3] gestionar significa "Llevar adelante una iniciativa o un proyecto". Para poder llevar a cabo esta tarea, es necesario responder a las cuestiones de qué, cómo, cuándo, por cuánto y por quién se van a realizar las acciones necesarias para completarlo.

Esto aplica tanto a los modelos de gestión tradicionales como a los ágiles, sin embargo, mientras que en la gestión tradicional esta tarea suele estar centralizada en una persona o en un grupo de personas, la gestión de un proyecto Agile se reparte entre todos los participantes, en mayor o menor medida, en función de sus capacidades y del papel que desempeñen.

Por lo tanto, al estar centrado este documento en la gestión de un proyecto desde una perspectiva ágil, no contiene un único punto de vista que explique cómo gestionar un proyecto de forma unidireccional, sino que se abordarán de forma integral todas las tareas necesarias para que éste salga adelante.

2.2.1 Modelo de gestión

Normalmente el ciclo de vida de un proyecto se estructura en torno a distintos procesos que van desde el inicio del proyecto cuando sólo es una idea, hasta su entrega final a usuario. Estos procesos se verán con más detalle en apartados posteriores, pero se pueden resumir en los 3 siguientes:

- Análisis y planificación: inicio del proyecto en el que se define el alcance, los costes y los plazos de entrega.
- Desarrollo: construcción de la solución en el que se debe hacer un control del avance que se está realizando
- Aprobación y cierre: proceso en el que el usuario prueba la solución y verifica que se ajusta a lo definido en el inicio, para poder ser entregado

Aunque estos procesos aplican de forma general a la mayoría de los proyectos, la forma de afrontarlos es la que determinará la diferencia entre las metodologías ágiles y las tradicionales.

2.2.1.1 Metodología Waterfall

Esta metodología también se conoce como **modelo de desarrollo en cascada** debido a que el desarrollo de del proyecto ocurre de manera secuencial. Es decir, el inicio de una tarea estará condicionado a la finalización de otra, de tal forma que no se podrá comenzar una etapa si no se ha finalizado completamente la anterior.

Además, otra característica principal de los modelos Waterfall es que la funcionalidad que se va a entregar se define completamente al inicio del proyecto y no se vuelve a revisar hasta el momento final de la entrega. Por lo tanto, en este modelo las etapas iniciales de análisis y planificación cobran una importancia vital, ya que el éxito del proyecto depende en gran medida del desarrollo de estas. Por ello, este modelo de gestión suele funcionar muy bien en los casos en los que las necesidades del proyecto se conocen con exactitud al inicio del proyecto y no varían a lo largo del tiempo.

[3] RAE: Real Academia Española de la lengua

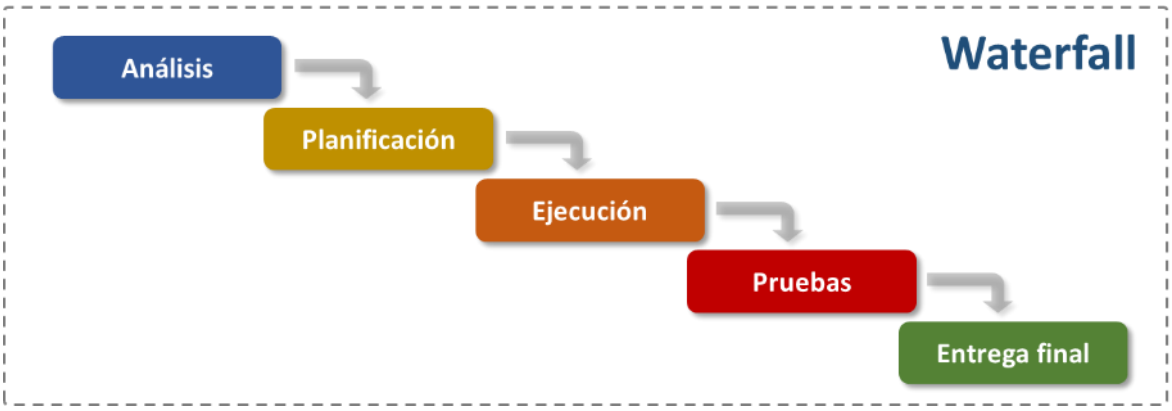

Figura 2.4 – Metodología Waterfall (Fuente: elaboración propia)

Para cada una de las fases será necesario generar una serie de documentos que permitan controlar el correcto desarrollo de las tareas necesarias, como son el Plan de proyectos, el documento de Diseño Técnico (DT), el documento de Diseño Funcional (DF) o el Plan de Pruebas, entre otros.

2.2.1.2 Metodología Agile

A diferencia de lo que ocurre en el modelo tradicional, las diferentes etapas no se desarrollan de forma secuencial, sino que se adopta un modelo **iterativo** e **incremental**, en el que se reevalúa el alcance y los objetivos conforme va avanzando el proyecto.

Para ello, se divide el alcance inicial en paquetes más pequeños y se priorizan según su importancia, para luego abordar su construcción en base a esta priorización e ir completando la solución final. Al final de cada iteración se evaluará el trabajo realizado para tomarlo como referencia en tareas futuras y corregir o mejorar los aspectos que sean necesarios

Esto permite que se pueda entregar valor en un plazo de tiempo menor ya que no es necesario esperar a que todos los procesos hayan finalizado para entregar el proyecto de forma conjunta. Además, el hecho de que sea iterativo e incremental permite que la solución final pueda ir adaptándose a las nuevas necesidades conforme avanza el proyecto si fuera necesario. En el siguiente esquema puede verse mediante una ilustración la diferencia entre ambos modelos.

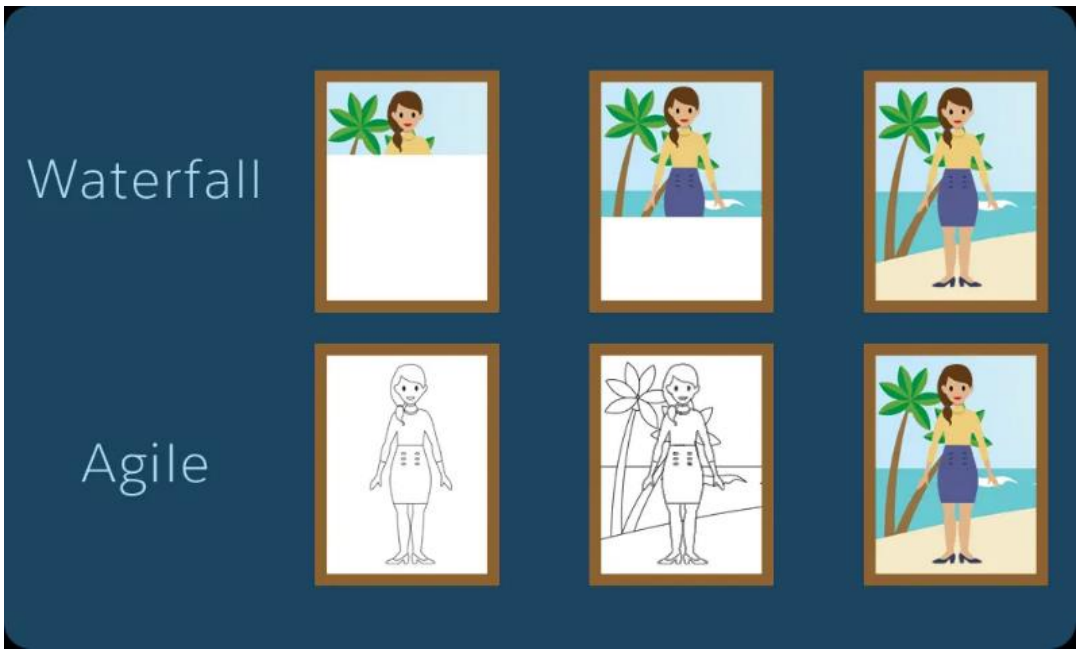

Figura 2.5 – Modelo de entrega Agile frente al tradicional (Fuente: [9])

Otra diferencia entre ambas metodologías se encuentra en la etapa de pruebas. En Waterfall las pruebas sólo se realizan al final del proyecto tras la etapa de construcción, sin embargo, en Agile se deben ir realizando pruebas a lo largo de toda la vida del proyecto conforme se termina la construcción de cada requerimiento. Este tema se desarrollará con más profundidad en un apartado específico.

Figura 2.6 – Ciclo Agile (Fuente: elaboración propia)

Además, mientras que en Waterfall la interacción con el cliente sólo ocurre en las etapas inicial y final del proyecto, en Agile se mantiene un modelo colaborativo durante toda la duración del proyecto. Obteniendo así un producto final que se va adaptando a las necesidades del usuario, aunque estas no sean las mismas al inicio del proyecto que en el momento de la entrega.

Es importante no confundir este modelo con "mini cascadas" sino que se abordan cada uno de los requerimientos por separado permitiendo flexibilizar e incluso paralelizar ciertas tareas que pueden seguir un curso de vida independiente. En la imagen siguiente se muestra un ejemplo de cómo podría ser un modelo de entrega ágil frente a uno tradicional.

Figura 2.7 – Ejemplo Agile vs Waterfall (Fuente: elaboración propia)

2.2.2 Equipo

Otro de los puntos clave y diferenciales en el modelo Agile es el equipo, cuya estructura rompe completamente con la que se venía utilizando en los modelos tradicionales. En los proyectos de tipo Waterfall existe una estructura completamente jerarquizada con la figura del jefe de proyecto a la cabeza y el equipo de desarrollo por debajo de éste. Sin embargo, en agile desaparece esta figura y se difumina la jerarquía, dando paso a un grupo de usuarios organizado en función del valor que pueda aportar cada uno.

En el modelo tradicional los integrantes se dividen en grupos muy diferenciados en función del papel que desempeñan con una interacción mínima entre ellos, como pueden ser los equipos de desarrollo, los de pruebas o los consultores funcionales. En Agile los equipos deben estar conectados y tener una comunicación continua y algunas personas pueden desempeñar diversos papeles

En cuanto a la gestión, mientras que en los proyectos tradicionales todas las decisiones, la gestión y control se centralizan en el jefe de proyectos, en Agile se reparten estas tareas entre los diferentes participantes del

proyecto en función de su naturaleza y las capacidades de cada uno. De esta forma, un equipo puede hacer el papel de jefe de proyecto, por ejemplo, determinando el coste de implantar los diferentes requerimientos, mientras que otro puede dedicarse a analizar la prioridad y el alcance de las tareas que se van a acometer. Esto mejora la productividad ya que disminuye las probabilidades de que se produzca un cuello de botella si el jefe de proyecto tiene alguna complicación.

Otra característica muy común en los equipos ágiles es que suelen estar autoorganizados, aunque puede existir una figura líder que facilita la comunicación y organización del resto del equipo. Cada proyecto tendrá sus particularidades y los roles no son un valor fijo, sino que dependerá de las necesidades de cada situación. Por tanto, aunque en un momento determinado una persona lleve a cabo ciertas tareas, en el siguiente proyecto puede cumplir un papel completamente diferente, permitiendo obtener el máximo partido en cada situación.

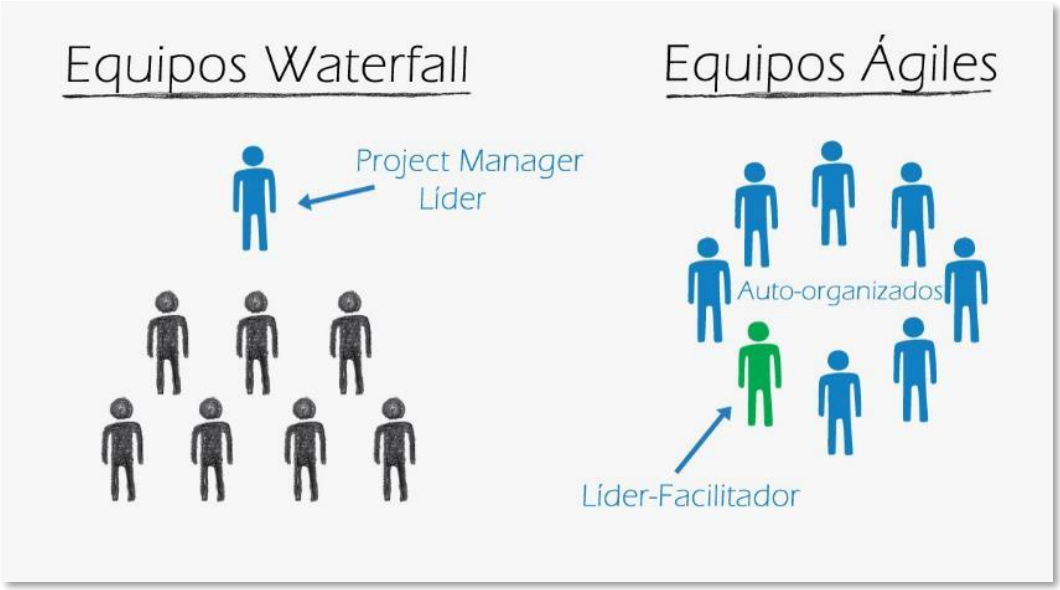

Figura 2.8 – Equipos Agile vs Waterfall (Fuente: [10])

2.2.3 Ventajas e inconvenientes

A continuación, se muestra una tabla resumen con la comparativa entre estos dos modelos con las características presentadas en los apartados anteriores:

Funcionalidad	Waterfall	Agile
Planificación	Toda la planificación relativa al proyecto se conoce de antemano	El proyecto se planifica conforme va avanzando
Detalle funcional	Las funcionalidades que se van a implementar se conocen al detalle antes de iniciar el proyecto	La funcionalidad que se va a implementar puede ir variando a lo largo del proyecto y su detalle no se conoce hasta justo antes de comenzar su desarrollo
Flexibilidad	No permite modificaciones una vez que el proyecto ha comenzado	Permite adaptarse a los cambios
Modelo	Modelo secuencial: no se puede comenzar una etapa si no ha finalizado completamente la etapa anterior. El producto final no se entrega hasta que no haya finalizado el proyecto	Modelo iterativo e incremental: A lo largo del proyecto se van produciendo entregas parciales de lo que conformará el producto final
Interacción con el cliente	La interacción con el cliente se produce antes de comenzar el desarrollo y cuando ya está construido	Se mantiene un modelo colaborativo entre todos los participantes durante todo el proyecto
Coste	El coste se puede determinar con facilidad desde el inicio	El coste puede ir variando conforme avanza el proyecto en función de las

		necesidades que se necesiten implementar
Gestión del equipo	Jerarquía estructurada en torno a un jefe de proyecto	Equipo autoorganizado
Organización del equipo	Equipos muy diferenciados con una mínima interacción entre ellos	Un mismo equipo con una comunicación continua para realizar las diferentes tareas

Tabla 2.1 – Comparativa Agile vs Waterfall (Fuente: elaboración propia)

En este apartado se enumerarán las principales ventajas e inconvenientes de adoptar el modelo ágil frente a los modelos tradicionales.

Ventajas

- **Mejora en la experiencia del cliente:** el cliente suele estar más satisfecho al estar más involucrado en todo el proceso, ganando visibilidad sobre el estado y el avance del proyecto.
- **Optimización de los recursos disponibles:** al ser un equipo autoorganizado, el reparto de tareas se hace de forma más eficiente ya que es el propio equipo quien lo hace, evitando los bloqueos que se producen en ocasiones por dependencias de una jerarquía. Además, también se fomenta la iniciativa y creatividad del equipo, mejorando a su vez el nivel de implicación y la motivación.
- **Mejora la flexibilidad y la rapidez:** al hacer entregas más pequeñas y evaluar de forma constante el alcance del proyecto se consigue mayor flexibilidad a la hora de realizar cambios ya que se pueden implementar más rápidamente cuando se detectan.
- **Anticipación en la detección los riesgos:** al trabajar con hitos más cortos, los riesgos se pueden detectar antes evitando que se acumulen hasta la entrega final.
- **Mejora en la calidad del producto entregado:** este modelo permite la reevaluación constante del producto a entregar para que se ajuste a las necesidades reales del usuario en cada momento. Esta es una de las ventajas más importantes, ya que en los modelos tradicionales puede ocurrir que el producto diseñado al inicio del proyecto quede obsoleto en el momento de su entrega

Inconvenientes

- **Incertidumbre:** esto ocurre principalmente en proyectos de larga duración, donde se conoce el punto de partida, pero en ocasiones la falta de límites en el proyecto y los cambios constantes pueden llevar a grandes desvíos frente a la planificación inicial.
- **Bloqueos por falta de disponibilidad:** aunque la participación constante del cliente favorece una mejora en la calidad del producto, este tipo de proyectos requieren de un alto nivel de disponibilidad. Por lo que cuando esto no es posible, el proyecto puede verse afectado e incluso bloqueado por no poder avanzar.
- **Modelos de solución insuficientes:** aunque es cierto que se gana mucha flexibilidad en este tipo de proyectos, a veces, cuando se definen nuevas necesidades puede ocurrir que se descubra que el modelo de solución ya implementado es insuficiente y hay que corregirlo añadiendo "parches" que cubran el nuevo requerimiento. Sin embargo, esto no ocurre con los modelos tradicionales en los que se decide la solución óptima desde el inicio del proyecto, ya que se conocen todos los requerimientos que se van a abordar previamente.
- **Documentación insuficiente:** las metodologías ágiles no presentan una alternativa clara para la recopilación de la documentación relativa al proyecto
- **Dificultad en la adopción de un nuevo modelo:** en muchas ocasiones las empresas no están preparadas para adaptarse a los cambios necesarios para la transición al nuevo modelo, lo que se traduce en una mala gestión de los proyectos.

Por lo tanto, aunque las ventajas que aporta este modelo son muy beneficiosas para la gestión de cierto tipo de proyectos, no siempre es la mejor solución. Por ello, es conveniente estudiar en cada caso que modelo se adapta mejor a las necesidades de cada proyecto antes de implantarlo.

2.3 Planteamiento del proyecto

Cómo se ha visto en los apartados anteriores, todo proyecto comienza con una etapa de análisis y planificación. Es en esta etapa cuando se responde a las preguntas de qué, cómo, cuándo, por cuánto y por quién se desarrollará el proyecto.

Aunque es cierto que en Agile la generación de documentación no es un tema prioritario, es recomendable recoger todos estos conceptos en un documento que sirva como guía a lo largo de todo el proyecto. Esto es lo que se conoce tradicionalmente como *Plan de proyecto* e indicará tanto el objetivo principal como los diferentes hitos que se deban cumplir.

En primer lugar se empezará por la toma de requerimientos, en la que se recogerán cada una de las necesidades que deben abordarse a lo largo del proyecto, para luego organizarlas y priorizarlas según su nivel de urgencia e importancia.

A continuación, se analizarán estos requerimientos en busca de una solución, valorando el coste de abordar cada uno de ellos y se planificará su ejecución. En este caso, dependiendo de la metodología Agile que se decida adoptar, la forma de abordar la planificación podrá variar.

Una vez que se tiene toda esta información, se elabora el Plan de proyecto que deberá ser aprobado antes de poder iniciar las tareas. Sin embargo, como hemos visto en apartados anteriores, al tratarse de metodología Agile, este documento no es definitivo, sino que sirve como punto de partida para el comienzo del proyecto e irá actualizándose a lo largo de la vida de éste para adaptarse a las necesidades en cada momento.

2.3.1 Toma de requerimientos, Historias de Usuario

La toma de requerimientos es uno de los aspectos fundamentales a la hora de comenzar un proyecto, ya que sienta las bases de lo que se va a desarrollar posteriormente. Si bien es cierto que en Agile los requerimientos pueden ir variando a lo largo del tiempo, no deja de ser fundamental hacer una buena definición de la necesidad, el alcance de ésta y qué implicaciones conlleva, de forma que facilite el trabajo tanto en la etapa de construcción como en la de pruebas.

Estas necesidades pueden recogerse y organizarse de muchas maneras, sin embargo, en Agile se suele hacer uso de las historias de usuario. y de las épicas o requerimientos. Las **épicas** son una serie de requerimientos a alto nivel que se pueden desglosar en funcionalidades más pequeñas. Estas funcionalidades más pequeñas son lo que se conoce como **historias de usuario** o **user stories**, que son narraciones cortas de una tarea desde el punto de vista del usuario. Generalmente, cada una de las historias está a su vez desglosada en **tareas** más pequeñas que deben completarse para darla por finalizada. Por lo tanto, la historia de usuario define el *qué*, mientras que las tareas que la componen definen el *cómo*.

Este conjunto de historias suele recogerse en lo que se denomina **Product Backlog** . que no es más que el catálogo de las funcionalidades que deben llevarse a cabo en un proyecto y que se verá con más detalle en apartados posteriores de este documento.

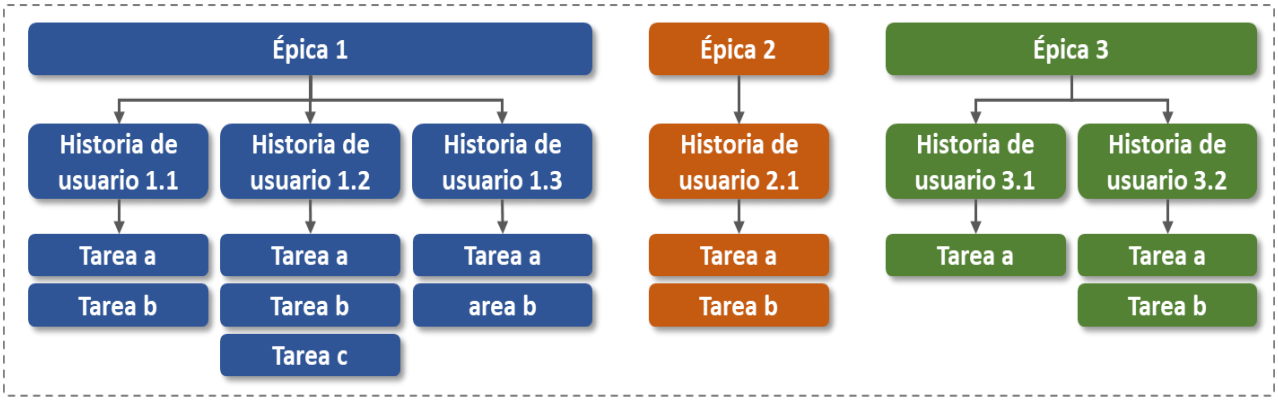

Figura 2.9 – Historias y Épicas (Fuente: elaboración propia)

Uno de los objetivos principales de Agile es poner a las personas en el centro del proyecto, por eso en este caso las historias de usuario son tan importantes. Estas historias deben ser breves y deben cumplir un patrón determinado que indique el quién, el qué y el para qué, este patrón es el que se presenta en el esquema siguiente:

Como [usuario] **quiero** [necesidad] **para** [beneficio]

Una vez concretada la historia de usuario es importante establecer de forma clara y concisa los criterios para identificar una historia como completada, estos son los **criterios de aceptación** que ayudarán a definir el cómo se va a construir y probar la funcionalidad asociada a esa user story. Esta parte es fundamental ya que aquí es donde se establecen los límites. Al igual que las users stories, los criterios de aceptación también suelen seguir una estructura determinada:

Dado [condiciones iniciales], **cuando ocurra** [evento], **debe producirse** [comportamiento esperado]

Para asegurar la completitud y calidad de una historia de usuario se utiliza la regla INVEST [11]:

- ✓ **I**ndependencia: deben ser independientes para poderlas priorizar y construir de forma separada
- ✓ **N**egociabilidad: los criterios de aceptación deben poder negociarse
- ✓ **V**alor: deben aportar valor al usuario final, este valor se indica en el "para"
- ✓ **E**stimación: tienen que poder ser estimables en función a las necesidades y los criterios establecidos
- ✓ **S**mall: pequeño en inglés, lo que quiere decir que debe poder abordarse en un plazo razonable
- ✓ **T**esteable: toda historia debe poder probarse para ser validada

Aunque las historias deben ser independientes, es decir se deben poder construir y probar de forma separada, en muchas ocasiones hay historias que no se podrán construir si no se han abordado otras primero, por lo que será necesario indicar las dependencias.

Además de los atributos ya mencionados, las user stories pueden incluir otros parámetros como la valoración o la prioridad, de las que se hablará posteriormente, o necesitar información adicional incluida en otros documentos. Esto ocurre especialmente en las funcionalidades que incluyen referencias a interfaces de usuario. En la siguiente imagen se puede ver un ejemplo de la estructura de una historia de usuario completa.

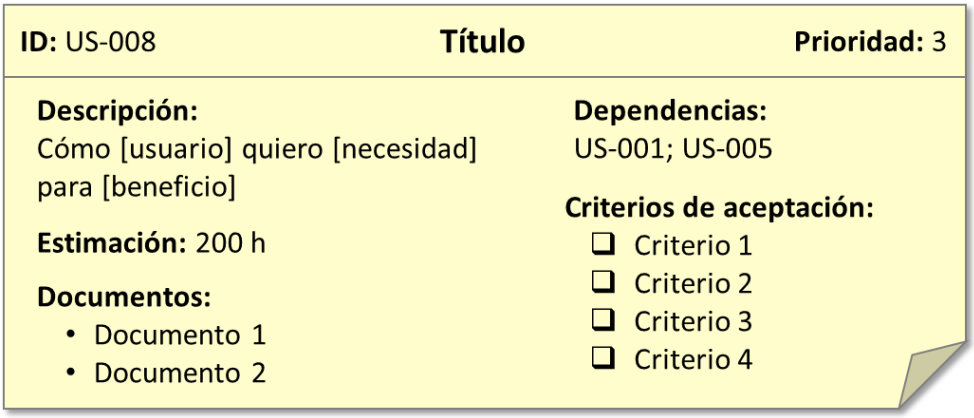

Figura 2.10 – Ejemplo user story (Fuente: elaboración propia)

2.3.2 Estimación

Otro proceso fundamental antes del comienzo de un proyecto es hacer una correcta valoración tanto del tiempo como del esfuerzo y los costes que implicarán llevarlo a cabo. Esta valoración se puede realizar de diferentes maneras y en base a muchos parámetros, en las metodologías tradicionales, por ejemplo, es muy usual hacer la estimación en horas persona, que llevará a su vez un importe asociado.

Cuando se habla de estimación en Agile hay que tener en cuenta que lo que se intenta medir es el tamaño y el esfuerzo de cada una de las tareas a realizar, es decir, establecer una valoración para cada una de las historias de usuario. Es un proceso bastante complejo y requiere de un análisis profundo de los objetivos, prioridades y limitaciones que se puedan encontrar.

Este proceso de estimación también es especialmente útil para evaluar correctamente la prioridad de un requerimiento, ya que el responsable de una tarea no siempre conoce las implicaciones y la complejidad necesaria para llevarla a cabo. Gracias a la valoración podrá conocer la importancia relativa de cada necesidad.

Sin embargo, se debe tener en cuenta que la estimación también es un proceso sujeto a la filosofía Agile y que, aunque sea necesario tener una valoración inicial para conocer la dimensión de las tareas a realizar, esta estimación puede variar y reajustarse conforme avanza el proyecto.

Para ello, en las metodologías ágiles se suele utilizar una estimación relativa, es decir, no se valora de forma independiente el esfuerzo y la complejidad de ejecutar cada user story, sino que se compara esta estimación con otras cuyo valor es conocido, teniendo en cuenta la incertidumbre como otro factor a valorar. De esta forma, conforme va avanzando el proyecto se reevalúa la estimación de las nuevas tareas, comparándolas con las estimaciones y esfuerzos reales empleados en las tareas anteriores.

Una de las herramientas más recomendadas en las metodologías ágiles para este tipo de estimación es el uso de los **Story Points** o Puntos Historia en castellano [12]. Los Story Points son una medida arbitraria que otorgan los equipos a las User Stories que van a desarrollar e indican el tamaño relativo y la complejidad que tiene una historia con respecto a las demás. Por lo tanto los puntos historias no son equiparables entre diferentes proyectos, sino que variarán en función del equipo que haga la estimación y de las condiciones del propio proyecto.

Para asignar estos Story Points a las diferentes historias de usuario, se suele emplear una técnica denominada Planning Poker que consiste en estimar el valor de cada user story mediante una baraja de cartas. En este método cada integrante del equipo debe tener una baraja, cuyos valores siguen generalmente la secuencia de Fibonacci[4], y que representan los Story Points que se le asociarán a la historia de usuario. Una vez que se conocen todas las especificaciones de la user story, los miembros del equipo seleccionan una carta con la valoración que hayan estimado y entre todos se debe llegar a un consenso sobre los puntos de usuario que se le asociarán finalmente. Las cartas se muestran todas a la vez, lo que permite que nadie se vea influenciado por la opinión de los demás. Además, si alguien ha dado una estimación mucho más alta o mucho más baja que los demás se estudian los motivos por los que se ha valorado así, por si existen impactos que el resto del equipo no ha tenido en cuenta o si hay una solución más sencilla que los demás no hayan identificado.

Figura 2.11 – Planning Poker (Fuente: [13])

[4] Secuencia de Fibonacci: Es una sucesión infinita de números naturales en la que cada término es la suma de los dos anteriores; 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89…

Un tema importante a tener en cuenta cuando se habla de story points, es que estos no son una estimación del tiempo o los costes que implica una historia de usuario, sino de la complejidad relativa dentro del Product Backlog. Para conocer estos parámetros se debe recurrir a KPIs[5] como la velocidad, que indica el periodo de tiempo que tarda el equipo en ejecutar x puntos de historia y que dependerá tanto del equipo en cuestión como de la valoración que se haya hecho.

2.3.3 Planificación

Finalmente, una vez que se conocen tanto las necesidades del proyecto como el coste de llevarlas a cabo, es posible planificar el proyecto. Se parte de la base de que todo proyecto busca obtener la mayor calidad en el menor tiempo posible y a un coste lo más reducido posible, por lo que habrá que llegar a un equilibrio entre estos tres factores: tiempo, alcance y coste. Esto es lo que se conoce como triángulo de hierro [8] de la gestión de proyectos.

Una de las premisas en las que se basa este triángulo es la dependencia entre estas variables, de forma que como máximo se pueden fijar dos, teniendo que estimarse la tercera. Esto quiere decir que, si re reduce por ejemplo el tiempo en el que se debe producir la entrega, se debe acortar el alcance o habría que aumentar el presupuesto para aumentar los recursos.

Figura 2.12 - Triángulo de la gestión de proyectos (Fuente: elaboración propia)

Para poder ajustar estas tres variables en Agile hay que tener en cuenta los siguientes aspectos relacionados con cada una de ellas:

- **Coste**: como se ha visto en el apartado de estimación, cuando se habla de coste en Agile, se habla de la valoración que se ha dado a cada una de las tareas a realizar.
- **Tiempo**: se debe tener en cuenta que en Agile se abordan los proyectos desglosándolos en agrupaciones más pequeñas. Para el caso de que se desarrolle en base a iteraciones, habrá que tener en cuenta tanto el tiempo que dura cada iteración como la duración global del proyecto. El primero vendrá establecido por la metodología Agile que se decida adoptar y el segundo variará según el proyecto. Se debe tener en cuenta que, como el resto de los parámetros, el tiempo que se determine es una estimación, pudiendo variarse en el futuro si fuera necesario.
- **Alcance**: el alcance global del proyecto se compondrá de aquellas necesidades recogidas en el Product Backlog que se hayan incluido en cada una de sus iteraciones.

Por lo tanto, para poder planificar el proyecto en primer lugar habrá que recoger las necesidades, estimarlas y valorarlas. Posteriormente será necesario ordenarlas y priorizarlas según su importancia. En el capítulo siguiente se mostrarán diferentes formas de organizar y planificar un proyecto en función del marco de trabajo que se decida implementar.

Es importante destacar que Agile no se pretende proporcionar una planificación cerrada, sino que generalmente se propone una estimación de cómo pueden organizarse las tareas, que se podrá modificar a lo largo del desarrollo del proyecto de forma iterativa.

[5] KPIs: indicadores

2.4 Ejecución del proyecto

Este apartado incluye tanto los procesos de construcción y su control, como las pruebas que deben realizarse para asegurar que la solución entregada cumple con las necesidades identificadas. Al igual que ocurre en el apartado anterior con la planificación, la organización de un proyecto durante el desarrollo de las soluciones viene determinada por la metodología que decida implementarse. Al ser un tema muy amplio, se ha decidido dedicarle el siguiente capítulo, por lo que en este apartado solo se revisarán algunos conceptos que de forma general pueden ser de utilidad para todos los proyectos ágiles durante estos procesos.

2.4.1 Métricas en Agile

En las metodologías ágiles una forma de controlar el grado de avance del proyecto es evaluar el trabajo que queda pendiente y el que ya se ha realizado y comprarlo con el grado de avance ideal que el proyecto debería tener. Para ello, se suele hacer uso de una gráfica conocida como Burn Down Chart o diagrama de quemado. En esta gráfica generalmente se presenta el volumen de trabajo pendiente en el eje vertical y el tiempo en el horizontal y se va registrando el volumen real que se ha finalizado.

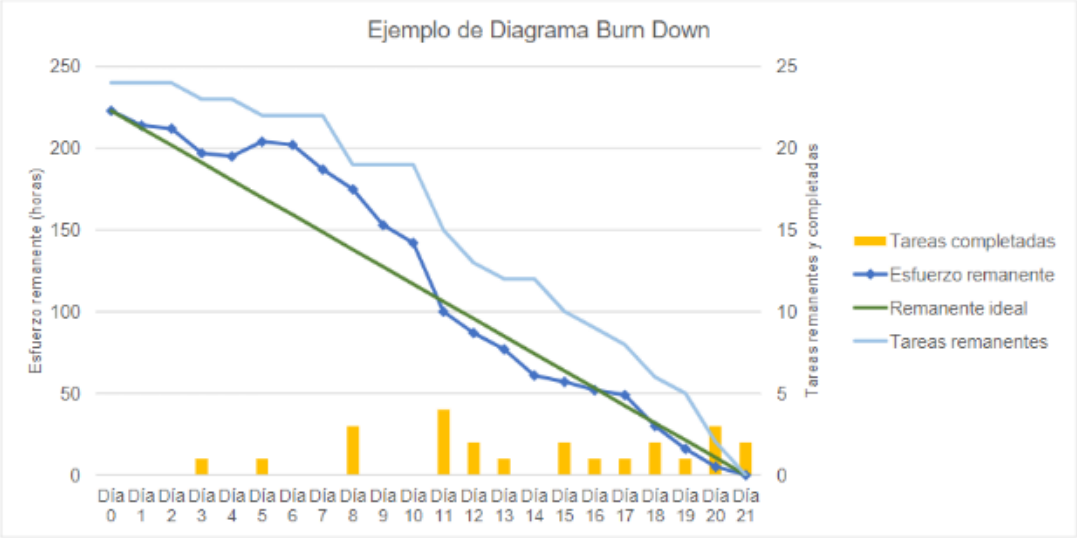

Figura 2.13 – Ejemplo de diagrama Burn Down Chart (fuente: [14])

Además del grado de avance, otro factor importante a evaluar en un proyecto es la velocidad. Ésta nos sirve para conocer la cantidad de funcionalidad que un equipo es capaz de implementar en un determinado periodo de tiempo. Generalmente se suele evaluar el número de puntos de historia que un equipo es capaz de acometer en cada iteración.

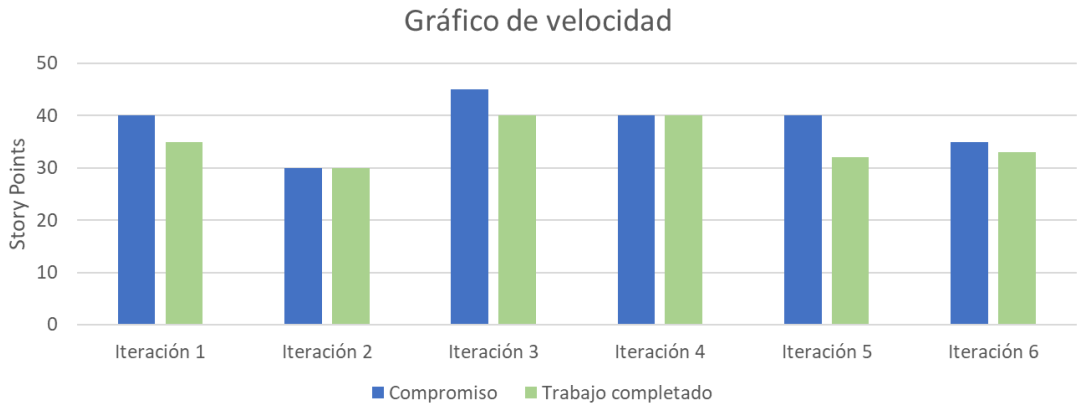

Figura 2.14 – Gráfico de velocidad (Fuente: elaboración propia)

Como norma general la velocidad suele ir en aumento conforme el proyecto va avanzando ya que el equipo empieza a conocer mejor los procesos y se van optimizando los flujos de trabajo. Este indicador es de mucha ayuda para decidir la cantidad de trabajo que se va a desarrollar en la próxima iteración.

2.4.2 Pruebas

El enfoque de las pruebas en las metodologías ágiles es muy diferente al que existía en el modelo en cascada. Mientras que en el modelo tradicional existía un equipo de testing que sólo intervenía en el proyecto en la etapa de pruebas, en el modelo ágil las pruebas se tienen que hacer de forma continua durante todos los procesos del proyecto. Además, en el modelo Waterfall la interacción entre este equipo y el de desarrollo sólo ocurría para reportar un error e indicar que las pruebas debían volver a ejecutarse. Sin embargo, en Agile las personas que ejecutan las pruebas y las que desarrollan la solución tienen que mantener una comunicación fluida y constante.

Otra diferencia con los modelos tradicionales es el nivel de conocimiento de los procesos y el grado de libertad del equipo de pruebas para evaluar las pruebas. En Waterfall, las pruebas a realizar tienen que estar muy detalladas de antemano y no es necesario que el equipo conozca el proceso, ya que únicamente debe comprobar que ocurre lo que el plan de pruebas indique como resultado esperado. En Agile la documentación de pruebas no tiene tanto detalle y es el propio equipo de pruebas quien determina si un proceso funciona correctamente. Para ello es necesario que conozcan en profundidad los procesos y la funcionalidad que requiere.

Un factor fundamental en las metodologías ágiles es el poder realizar entregas en un periodo de tiempo muy corto. Por este motivo, es necesario incluir herramientas que permitan la automatización de un porcentaje amplio de las pruebas.

En cuanto al tipo de pruebas que se pueden realizar en un proyecto ágil, éstas no distan mucho de las que se hacen en las metodologías tradicionales, aunque la distribución y la organización cambie según los factores presentados en este apartado.

Aunque existen muchos más tipos de pruebas en este documento se destacan las más relevantes:

- **Pruebas unitarias:**

  Se llaman unitarias porque su objetivo es evaluar la unidad de código. Consisten en probar las funcionalidades y códigos de forma individual para asegurar que su funcionamiento es correcto. En estas pruebas solo se evalúa cada parte de un flujo independientemente del funcionamiento del resto.

  Suelen ser ejecutadas de forma interna por los equipos de desarrollo, aunque también se pueden realizar conjuntamente con el usuario. De forma general son rápidas de ejecutar por lo que su automatización es poco costosa.

- **Pruebas integradas:**

  Suelen ser el paso posterior a las pruebas unitarias. Sirven para verificar que los diferentes módulos y funcionalidades encajan entre sí. En este tipo de pruebas también se verifica que las conexiones con otros sistemas funcionan correctamente. Suelen ser internas y aseguran que los procesos funcionan correctamente entre ellos.

  Pueden ser pruebas con un carácter más técnico, como pueden ser por ejemplo las conexiones entre servidores web, o con carácter mas funcional, como la verificación de que un flujo funciona correctamente. En los casos en que los procesos se ejecuten de principio a fin, estas pruebas se denominan end to end (E2E).

- **Pruebas de regresión:**

  Son pruebas que se realizan en un entorno dónde ya había funcionalidad implementada y se añaden nuevos desarrollos. Sirven para asegurar que los procesos existentes siguen funcionando de forma correcta cuando se incorpora una nueva funcionalidad. Es un caso muy común de pruebas automatizadas.

- **Pruebas de rendimiento:**

  Son pruebas técnicas y se emplean en los casos en los que se quiera evaluar el comportamiento del sistema bajo las condiciones de máxima exigencia.

- **Pruebas de aceptación de usuario (UAT):**

  Las pruebas de aceptación son las que realiza formalmente el usuario y con ellas se verifica que el

software satisface las necesidades y requerimientos del cliente. Estas pruebas suelen ser manuales y es el propio usuario quien determina su volumen y su alcance.

3 PRINCIPALES MARCOS DE TRABAJO

"Primero formamos los hábitos y luego ellos nos forman. Conquiste sus malos hábitos o ellos los conquistarán a usted"

- Rob Gilbert –

Como se ha visto en el capítulo anterior, cualquier organización que desee adoptar el modelo ágil debe estar dispuesta cambiar su cultura y forma de trabajar para adaptarse a las exigencias que presenta la gestión Agile, sin embargo, ésta no es una tarea fácil. Para ello, existen una serie de prácticas y herramientas que facilitan la adopción de este modelo. Esto es lo que se conoce como *frameworks*[6] o marcos de trabajo.

En este capítulo se detallarán los principales marcos de trabajo que se emplean dentro del mundo Agile, entre los que destacan *Scrum*, *Kanban* o *Extreme Programming (XP)* al ser algunas de las alternativas más utilizadas en la actualidad. Estos frameworks se diferencian entre sí en las métodos y herramientas que emplean, aunque todos tienen en común los mismos principios y valores.

Cabe destacar que estos marcos no son excluyentes, sino que pueden utilizarse de forma complementaria según las necesidades y particularidades de cada aplicación, como es el caso del *Scrumban* que unifica las prácticas Scrum con las herramientas del modelo Kanban.

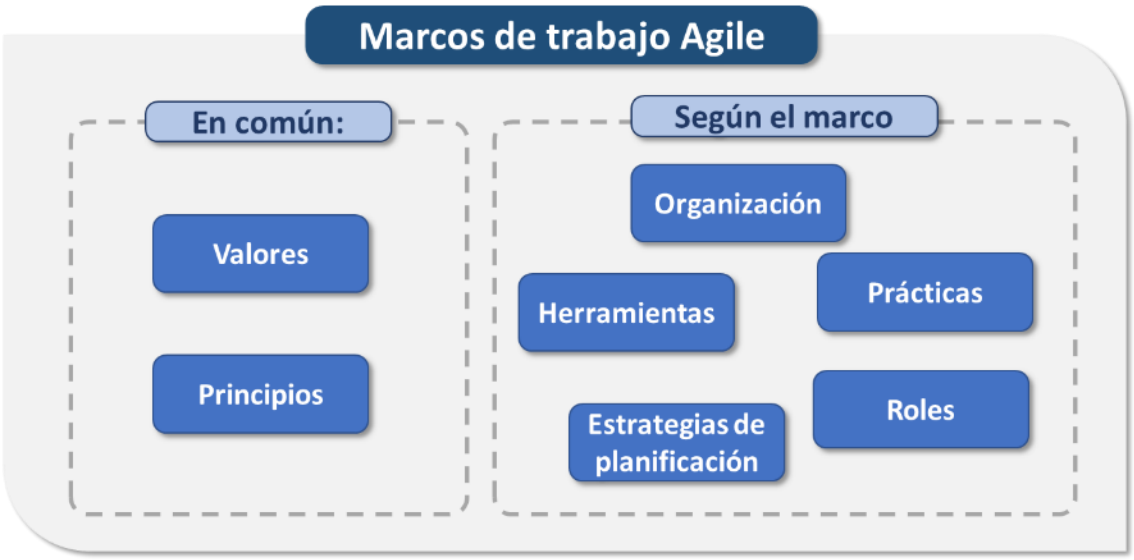

Figura 3.1 – Frameworks, características en común (Fuente: elaboración propia)

[6] Framework o marco de trabajo, es un conjunto de prácticas, criterios y herramientas enfocadas a resolver una problemática concreta

3.1 Scrum

Es el marco de trabajo más conocido y empleado dentro del modelo Agile. Su objetivo es estructurar y organizar las necesidades de manera que puedan ser abordadas por los equipos de una manera más manejable y flexible, asegurando que el producto entregado se ajusta a las necesidades reales de cada momento.

Su origen se remonta a 1986 cuando Ikujiro Nonaka y Hirotaka Takeuchi publican "The New Product Development Game" [1]. En este artículo, Nonaka y Takeuchi, comparan la formación en melé[7] de los equipos de Rugby con la forma de trabajo presentada en su estudio y de ahí proviene el término "Scrum".

A principio de la década de los 90, Ken Schwaber y Jeff Sutherland pusieron en práctica los conceptos de este artículo en sus respectivas compañías. En 1995, Schwaber presentó "Scrum Development Process" [6] en la conferencia OOPSLA 95.

Figura 3.2 - Ken Schwaber y Jeff Sutherland [15]

Desde entonces Sutherland y Schwaber colaboran para mejorar y evolucionar las prácticas englobadas dentro del marco Scrum. Para ello han creado una guía [15] que detalla todos los conceptos y mecánicas asociadas a este framework. La guía se actualiza con bastante regularidad y está disponible en varios idiomas en su página web "Scrum Guides" [16].

Según esta guía, Scrum se define como "un marco de trabajo a través del cual las personas pueden abordar problemas complejos adaptativos, a la vez que se entregan productos de forma eficiente y creativa con el máximo valor".

Este marco se sustenta en tres pilares fundamentales:

- **Transparencia:** todos aspectos relevantes del proyecto serán claros y accesibles para todos los participantes del proyecto.
- **Inspección:** se debe evaluar de forma frecuente el estado del proyecto para asegurar que el progreso que se está obteniendo es el esperado y corregir a tiempo si fuera necesario.
- **Adaptación:** las necesidades detectadas en un inicio pueden variar a lo largo de éste, por lo que es fundamental que se tenga la capacidad de adaptación a estos cambios.

[7] Melé: formación fija en el rugby en la que los integrantes del equipo funcionan como un mismo bloque para ejercer resistencia sobre el equipo oponente.

Scrum está basado en un modelo **iterativo** e **incremental**, en el que se listan y ordenan todas las necesidades y funcionalidades requeridas y se dividen en agrupaciones más pequeñas, que se irán implementando en iteraciones. De esta forma, se irá añadiendo funcionalidad a la que ya se ha implementado, lo que se conoce como Incremento. Al principio y al final de cada iteración se evaluará tanto los siguientes requerimientos que se van a abordar como el análisis de los ya entregados.

Para poder trabajar de esta forma, el marco Scrum proporciona ciertos conceptos y herramientas que facilitan el trabajo de los equipos y su organización. Estos conceptos se revisarán en los siguientes apartados, en los que se detallarán los ***artefactos***, que son las herramientas que permiten organizar la funcionalidad, los ***eventos***, que ayudan en la planificación y los ***roles*** y características del equipo que deberá ejecutar todo el proyecto.

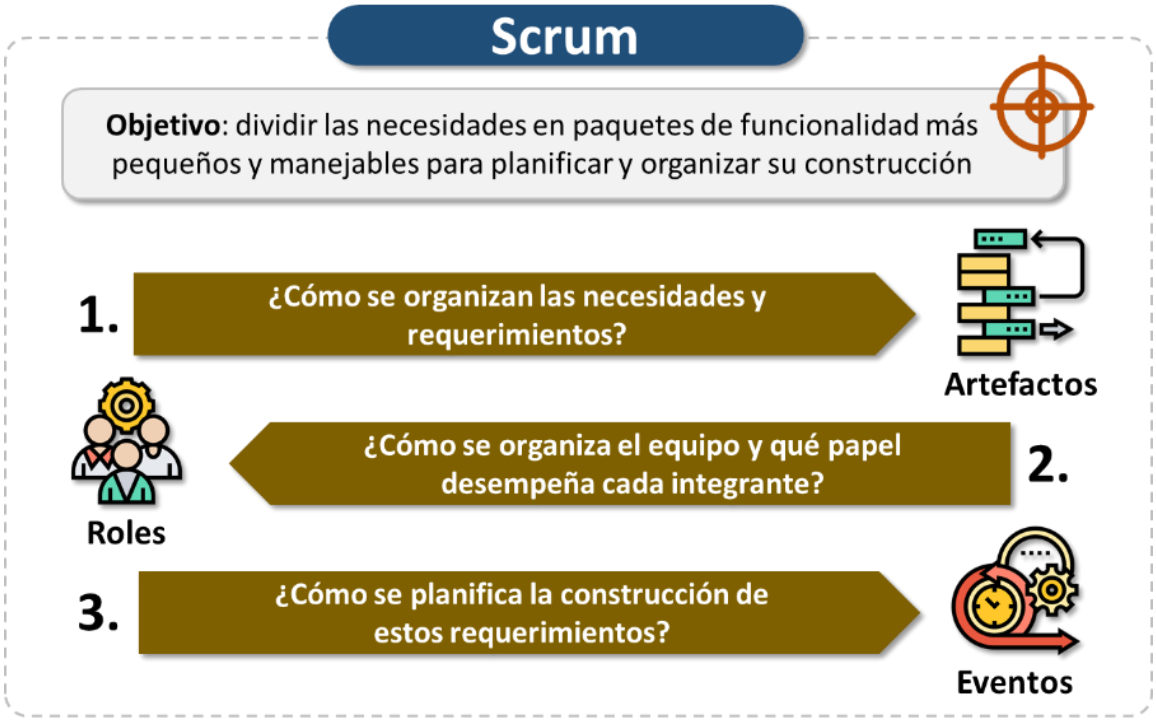

Figura 3.3 – Conceptos Scrum (Fuente: elaboración propia)

3.1.1 Artefactos, organización de los requerimientos

Los artefactos son las herramientas que se emplean en el marco scrum para organizar el trabajo y la funcionalidad que se va a desarrollar. Su objetivo es asegurar la transparencia y la claridad a la hora de implementar una solución. Dentro de estos artefactos se pueden encontrar el *Product Backlog,* el *Sprint Backlog* y el Incremento.

- **Product Backlog:** también denominado pila de producto en español, es un listado de todas las funcionalidades, historias de usuario, épicas o requerimientos, mejoras y correcciones que se deben incluir en el proyecto. Cada elemento de la lista debe tener una descripción, unos criterios de aceptación, una valoración y estar ordenado según su prioridad.

  Este listado no es fijo, sino que irá variando, ampliándose y reordenándose a lo largo del proyecto aumentando su valor y ajustándose a las necesidades detectadas en cada momento. De esta forma se puede decir que se trata de un artefacto vivo, que se revisará de forma constante.

- **Sprint Backlog:** para facilitar las tareas de planificación y desarrollo el Product Backlog se divide en porciones más pequeñas y manejables, estas agrupaciones reciben el nombre de Sprint Backlog y se decide su alcance al inicio de cada iteración o *Sprint*. En este Sprint Backlog tendrán que incluirse además las tareas y la planificación necesarias para llevar a cabo el requerimiento. Para asegurar el avance y la mejora continua del proyecto, debe contener al menos un requerimiento de alta prioridad.

- **Incremento:** está compuesto por el conjunto de elementos del Product Backlog que ya han sido completados, contiene todos los elementos de los Sprint Backlogs ya finalizados.

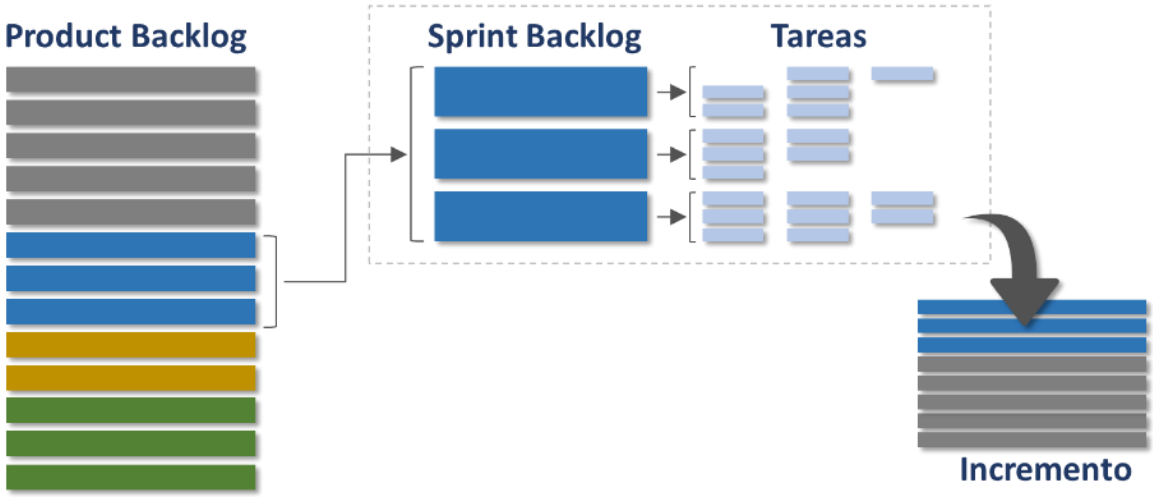

Figura 3.4 – Artefactos Scrum (Fuente: elaboración propia)

3.1.2 Roles, el equipo Scrum

Los equipos de Scrum o *Scrum Teams* son equipos auto-organizados y multifuncionales. Es decir, dentro del equipo se tienen todos los conocimientos y habilidades necesarias para cumplir los objetivos. Además, es el propio equipo quien decide qué tareas se asignan a cada integrante, sin necesidad de depender de otras personas.

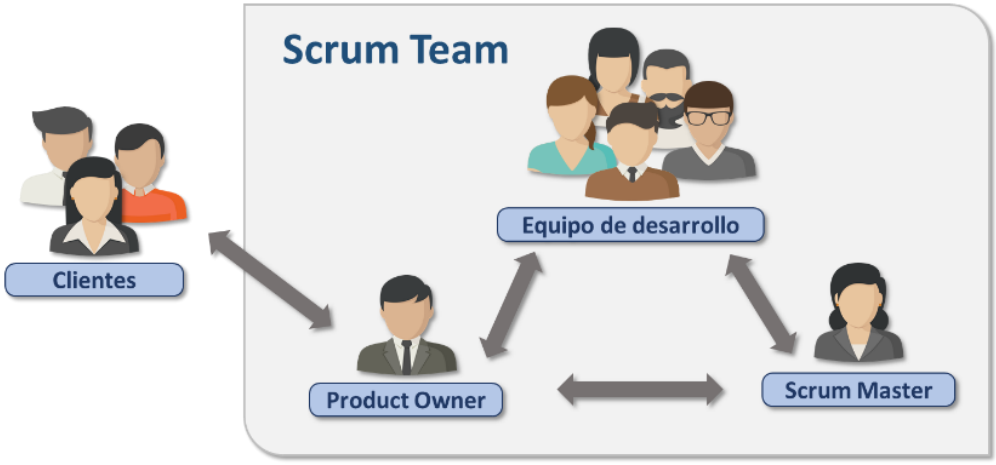

Figura 3.5 – Equipo Scrum (Fuente: elaboración propia)

Dentro del marco Scrum se pueden identificar tres roles principales: *Product Owner*, *Development Team* y *Scrum Master*.

- **Product Owner (PO):** . es el propietario del proyecto, es el encargado de conocer todas las necesidades del cliente para poder crear, validar y priorizar el *Product Backlog*. Para ello debe mantener una comunicación frecuente con el cliente. Su deber es asegurarse de que los requerimientos sean claros y entendibles para todos los participantes del proyecto.

  Puede delegar estas tareas en otros miembros del equipo, pero siempre debe revisarlas y validarlas ya que es el responsable último de este documento. Un equipo Scrum sólo puede tener un único *Product Owner* y este puede formar parte del equipo de desarrollo. Para el correcto desarrollo del proyecto, toda la organización debe seguir y respetar sus decisiones.

- **Development Team:** es el equipo de desarrollo. Encargado de la construcción e implementación de los elementos del *Product Backlog*. También es el responsable de determinar el esfuerzo necesario y las tareas que se deben realizar para completar cada uno de los requerimientos.

  Es un equipo autoorganizado y no jerarquizado, la planificación y distribución de tareas se organizan dentro del propio equipo en función de las fechas objetivo y las habilidades de cada uno. Estas habilidades deben ser suficientes para construir toda la solución. No existe división en subequipos, sino que todos son responsables de todas las tareas, aunque se hagan repartos.

- **Scrum Master:** es el encargado de facilitar el trabajo de todo el equipo, eliminando los posibles obstáculos que se puedan encontrar en el transcurso del proyecto. También debe asegurarse de que los plazos establecidos para los eventos se respeten.

3.1.3 Eventos, descripción del proceso

Los eventos en Scrum se emplean para mejorar la eficiencia de los procesos y reducir los riesgos de desvíos en la planificación. Para ello, se introduce un límite de tiempo en cada uno de ellos, reduciendo así la pérdida de tiempo en reuniones extensas e improductivas. Además, también se busca aumentar la frecuencia de los puntos de control donde se evalúa el estado del proyecto.

El proceso siempre comienza con una toma de requerimientos en la que se construirá una primera versión del Product Backlog, que irá evolucionando a lo largo del proyecto. Posteriormente, se introduce la figura del **Sprint**que es la base de Scrum, que consiste en ejecutar el proyecto en iteraciones cortas de máximo 3 o 4 semanas. La duración de cada Sprint no se puede modificar una vez fijada y es importante que no exceda este límite de tiempo, ya que si la duración es muy extensa es más probable que la complejidad aumente y que se produzcan cambios en la definición de las necesidades.

En cada iteración se abordarán los elementos que se hayan incluido en el Sprint Backlog que constituyen el objetivo del Sprint (*Sprint Goal*). La construcción de la solución se debe hacer siguiendo un diseño y un plan que debe ser flexible. Al final de cada iteración se debe tener un producto terminado y listo para ser usado, que será entregado al cliente como un incremento cuando se estime oportuno.

Dentro de cada Sprint se engloban todos los eventos que se verán en los puntos siguientes, que consisten en la Planificación del Sprint (Sprint Planning), los Scrums Diarios (Daily Scrums), la construcción de la solución, la Revisión del Sprint (Sprint Review), y la Retrospectiva del Sprint (Sprint Retrospective).

- **Sprint Planning:** tiene una duración máxima de 8 horas y consiste principalmente en dos procesos, la selección de los requerimientos que se van a desarrollar en el Sprint y la planificación de cómo se van a construir esos requerimientos.

  Para la selección de los requerimientos el equipo de desarrollo es el encargado de determinar qué elementos del Product Backlog son capaces de completar para ese Sprint. Para ello deben resolver las dudas que tengan con el cliente y el *Product Owner* con respecto a la definición de los requerimientos.

  Una vez que se ha establecido el objetivo del Sprint, se debe establecer un plan para ejecutarlo y las tareas que son necesarias para cada requerimiento, que no deben ser de más de uno o dos días. Con el conjunto de requerimientos más el plan para llevarlos a cabo, se construye el Sprint Backlog.

- **Daily Scrum:** se trata de una reunión diaria de 15 minutos en la que se evalúa el estado del Sprint. En estas reuniones se revisarán todas las tareas que se deben realizar hasta la siguiente reunión y los posibles bloqueos que se hayan encontrado. El equipo de desarrollo será el encargado de llevar estas reuniones, aunque el Scrum Master puede ayudar a controlar los límites de tiempo. Tras esta reunión es frecuente que el equipo o parte de él tenga que volver a reunirse para revisar ciertos detalles con más detenimiento.

- **Sprint Review:** al final de cada Sprint se realiza una revisión de éste en la que se debe hacer una demostración del incremento del producto ya terminado al Product Owner y a las personas que éste estime necesarias.

  Esta reunión debe tener una duración de no más de 4 horas y es el Scrum Master el encargado de facilitar que así sea. Además, se revisará de nuevo el Product Backlog y se modificarán e incluirán los requerimientos que sean necesarios, facilitando la tarea en la planificación del siguiente Sprint.

❖ **Sprint Restrospective:** tras Sprint Review se realiza una reunión de no más de 3 horas en la que el equipo evalúa cómo se ha desarrollado el Sprint, los errores y los aciertos y las posibles mejoras que se pueden implementar para el próximo Sprint.

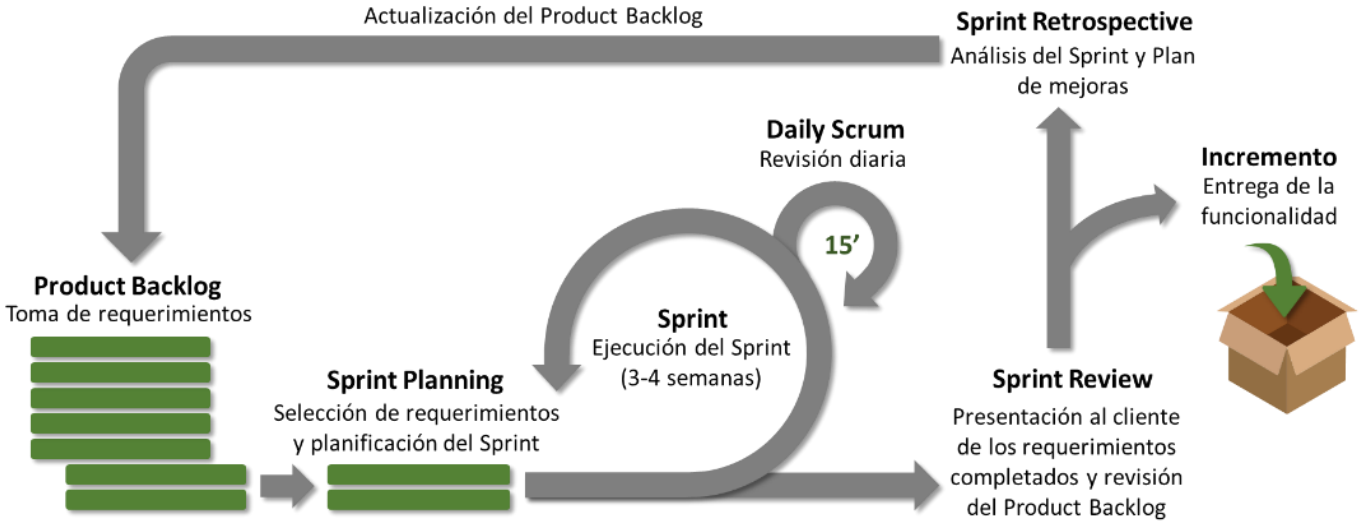

Figura 3.6 – Proceso Scrum (Fuente: elaboración propia)

3.2 Kanban

Como bien indica su nombre ("Kan" visual y "ban" tarjeta, en japonés), Kanban. es método para gestionar el trabajo de forma visual mediante el uso de tarjetas. Su objetivo es organizar visualmente las tareas que se deben llevar a cabo, de forma que se controle en todo momento el estado de cada una de ellas, y así poder identificar fácilmente el camino crítico y cuáles son las que impiden que el trabajo avance.

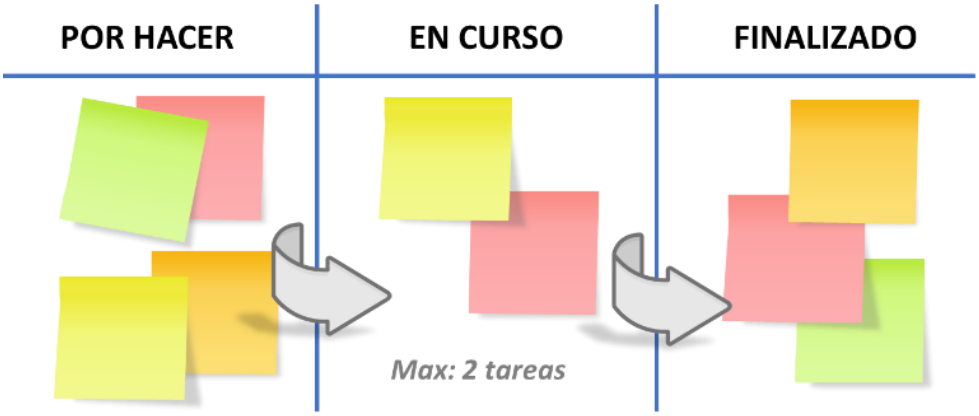

Figura 3.7 – Tablero básico Kanban (Fuente: elaboración propia)

Este método surgió a finales de los 40 en el sector automovilístico de la mano Taiichi Ohno, ingeniero en la empresa japonesa Toyota. Ohno implantó un sistema de gestión de la producción *Just in time* (JIT) [8] también conocido como Toyota Production System (TPS), cuyo objetivo era optimizar los recursos de tal forma que no fueran ni escasos ni excesivos sino que se dispusiera de la cantidad necesaria en cada momento. Este sistema se basa en ajustar la producción a la demanda del mercado, al contrario que en las prácticas tradicionales donde primero se produce y luego se busca la forma de vender el producto.

Sin embargo, se hizo popular dentro del ámbito de desarrollo de software gracias a David J. Anderson que adaptó los conceptos y prácticas del sistema Kanban original a este sector, publicando varios libros al respecto.

[8] Justo a tiempo

3.2.1 Organización del tablero

En este sistema se dividen las tareas en tarjetas que se organizan en un tablero según diferentes estados, permitiendo tener una visión completa del funcionamiento del proyecto. El tablero más básico consta de 3 estados organizados en columnas, "Por hacer", "En curso" y "Finalizado", aunque se pueden añadir o modificar en función de las necesidades identificadas.

Una de las principales prácticas en el método Kanban consiste en limitar el número de trabajos en curso, lo que se conoce como *Work in Progress* (WIP)[9]. De esta forma no se permite comenzar una tarea si no se han finalizado antes otras. Esto facilita que el equipo mantenga un ritmo de trabajo adecuado centrándose en finalizar cada tarea. Además, este sistema permite identificar de forma más rápida los cuellos de botella del proyecto, ayudando a prevenir bloqueos.

Otra práctica muy común a la hora de organizar los tableros Kanban, consiste en priorizar las tareas para mejorar la forma en que los equipos se organizan. Se puede hacer de muchas maneras, por ejemplo, organizando las tarjetas en filas o asignado un código de colores a las tarjetas según prioridad.

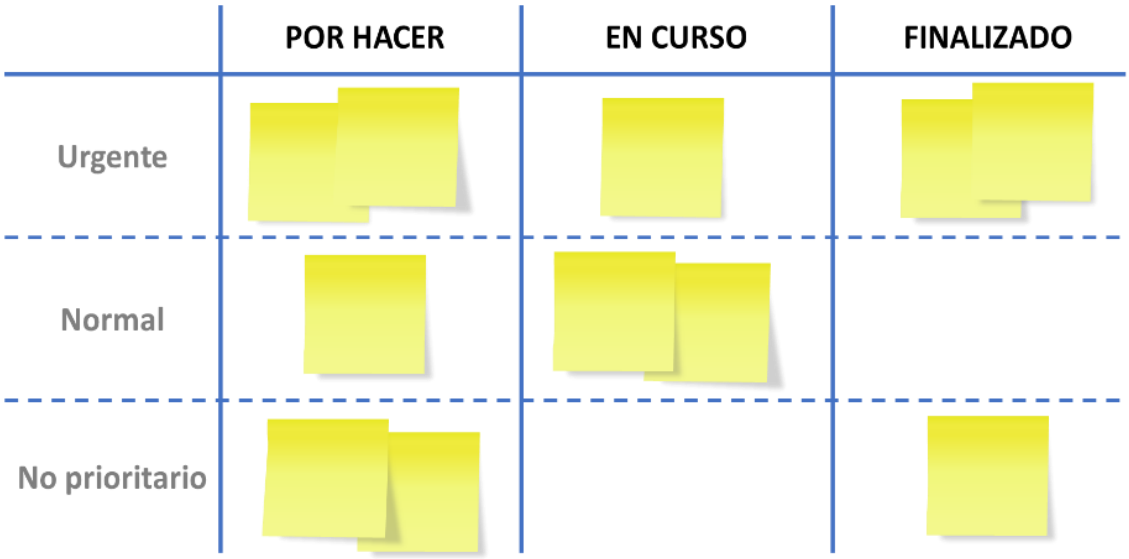

Figura 3.8 – Tablero Kanban priorizado por filas (Fuente: elaboración propia)

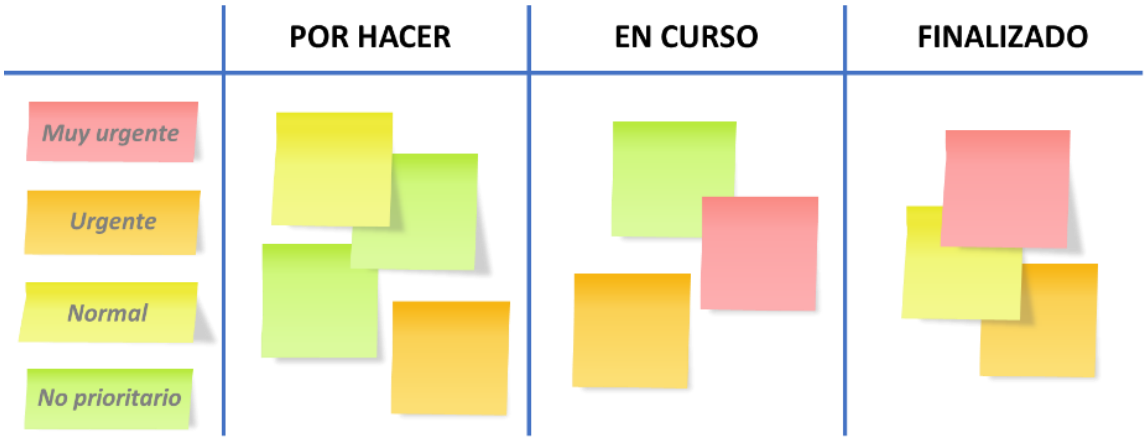

Figura 3.9 – Tablero Kanban priorizado por colores (Fuente: elaboración propia)

Además de la prioridad, se pueden organizar los tableros Kanban, clasificando las tarjetas en función de muchos otros factores, como el equipo, las líneas de trabajo, los entornos de desarrollo, etc. Permitiendo así, organizar el trabajo según las particularidades de cada proyecto.

[9] Trabajo en proceso

Debido a que muchas empresas trabajan en remoto y se establecen colaboraciones entre equipos que están a varios kilómetros de distancia, cada vez es más frecuente el uso de tableros digitales, con soportes en la nube que facilitan el acceso a la información en cualquier momento y en cualquier lugar. Entre las más destacadas se encuentran herramientas como *Kanbanize*, *MeisterTask* o *Trello*.

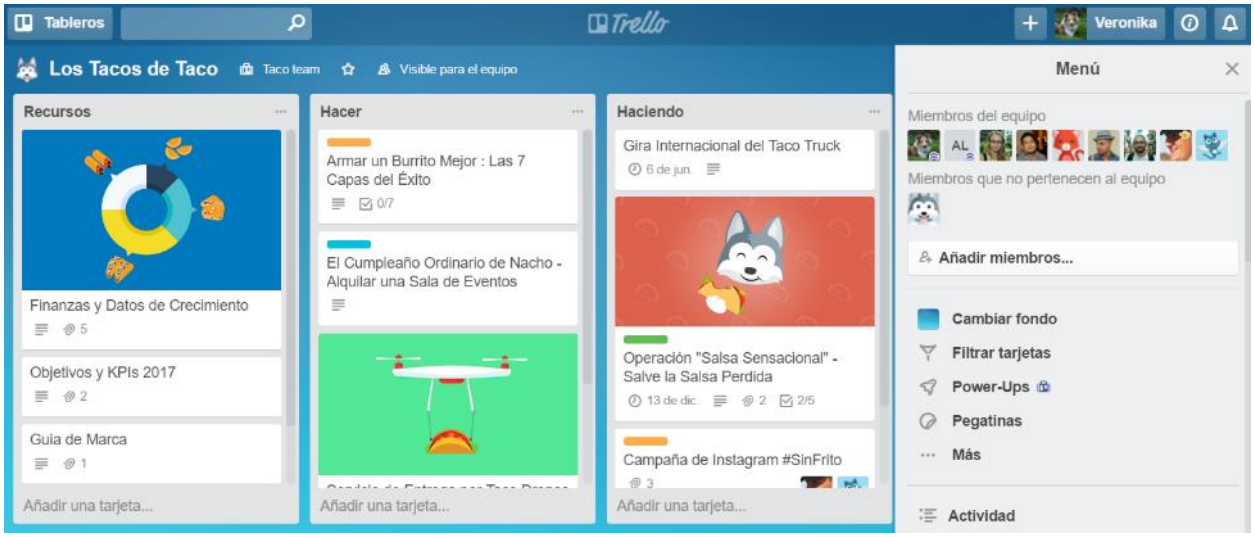

Figura 3.10 – Tablero Trello (Fuente: [17])

3.2.2 Tiempo de entrega y tiempo de ciclo

Para hacer un seguimiento correcto del avance del proyecto y facilitar su planificación, es necesario medir los tiempos de ejecución de las tareas. Para ello, Kanban se utilizan los términos **tiempo de entrega** y **tiempo de ciclo**.

El tiempo de entrega es el tiempo que pasa desde que se identifica una tarea y se coloca en la columna de "pendiente" hasta que pasa a la columna de "finalizada". Este parámetro ayuda a medir el tiempo que tarda una tarea en completarse desde que se decidió añadir al alcance.

El tiempo de ciclo sirve para medir una parte del proceso, es decir, el tiempo que tarde una tarea en pasar de una columna a otra. Este parámetro puede servir para medir muchos tipos de ciclos dependiendo de a que parte del proceso corresponda, como puede ser el tiempo de construcción o el de pruebas.

La comparación entre ambos parámetros puede informar, por ejemplo, de que las tareas pasan demasiado tiempo en espera antes de comenzar su desarrollo. De esta forma, el análisis de los tiempos puede ayudar a identificar problemas y cuellos de botella en el flujo y corregirlos.

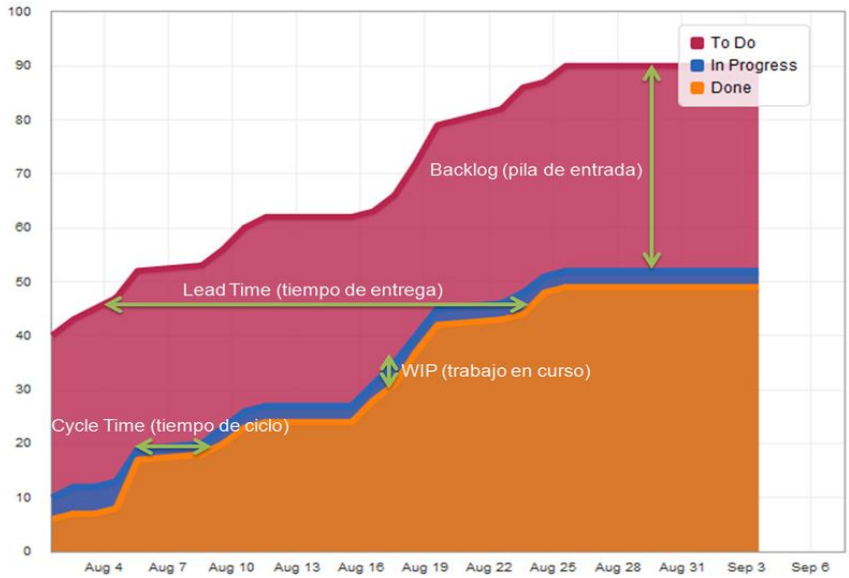

Figura 3.11 – Gráfica Kanban (Fuente: [18])

3.3 Extreme Programming (XP)

Extreme Programming o XP es una metodología de desarrollo de software incluida dentro de los marcos de trabajo ágiles. Fue presentada por primera vez por Kent Beck en su libro *Extreme Programming Explained: Embrace Change* (1999) y completada y mejorada posteriormente junto a Ron Jeffries y Ward Cunningham.

Esta metodología se caracteriza principalmente por enfocarse en la capacidad de adaptarse a los cambios más que en seguir un plan ya establecido, lo que la hace muy útil para abordar proyectos con requerimientos poco definidos o muy susceptibles de sufrir cambios. Esto se consigue revisando de forma continua e iterativa la funcionalidad que se va desarrollando, permitiendo modificar la solución en cuanto sea necesario.

Figura 3.12 – Bucle de retroalimentación en XP (Fuente: [19])

A continuación, se enumeran las principales características de esta metodología, basadas en los siguientes principios: simplicidad, comunicación, retroalimentación y coraje.

- Comunicación fluida y constante con el cliente y dentro del equipo
- Trabajo en iteraciones cortas
- Desarrollo del software de forma iterativa e incremental, es decir, implementando pequeñas mejoras unas detrás de otras
- Programación en parejas que consiste en que dos programadores trabajen a la vez en un mismo desarrollo, donde mientras uno se ocupa del teclado, el otro analiza y revisa lo que se está construyendo.
- Simplicidad en el código que permita que todos los miembros del equipo puedan entenderlo y trabajar con él de forma sencilla
- Alta frecuencia en la realización de pruebas unitarias, incluyendo pruebas de regresión
- Entregas frecuentes

XP tiene muchas similitudes con Scrum como el hecho de trabajar en base a iteraciones, sin embargo, existen ciertas diferencias entre ambas metodologías que se comentan a continuación. En XP al igual que en Scrum se trabaja en base a iteraciones, sin embargo, las de XP suelen ser mucho más cortas que los sprints de Scrum (en torno a 1 o 2 semanas como máximo). Además, los requerimientos definidos para estas iteraciones pueden modificarse siempre y cuando no se haya empezado su construcción, de forma más flexible que en Scrum, donde una vez cerrado la funcionalidad de un sprint, no se puede modificar. Esto es posible ya que el orden de desarrollo de los requerimientos los define el Product Owner y no el equipo como en el caso de Scrum, por lo tanto, se tiene más control sobre los requerimientos que se están llevando a cabo.

4 APLICACIÓN A CRM SALESFORCE

"Los clientes están discutiendo los productos y la marca de una empresa en tiempo real. Las empresas necesitan unirse a la conversación"

- Marc Benioff (fundador de Salesforce) -

Este capítulo pretende proporcionar una visión amplia sobre lo que implica la gestión de un proyecto en Salesforce basada en una metodología ágil. Para que esta visión sea completa es necesario abordar la gestión del proyecto desde todos los puntos de vista implicados en su desarrollo. Por ello, se estudiarán tanto los aspectos más técnicos, que generalmente competen a los equipos de desarrollo, como los aspectos más funcionales y de organización, relacionados con la recopilación de requerimientos y la aplicación de las metodologías ágiles presentadas en capítulos anteriores.

En cuanto a la estructura, en primer lugar se hará una breve descripción de qué es un CRM y para qué se utiliza, enumerando los CRM más relevantes en la actualidad. Posteriormente se particularizará para el caso concreto de Salesforce, donde se presentarán las capacidades que proporciona esta plataforma y los conocimientos que se necesitan para poder configurarla.

Finalmente se abordarán los conceptos principales que se deben tener en cuenta a la hora de gestionar un proyecto en Salesforce para cada una de las etapas de la vida de éste. Empezando por los procesos de análisis y diseño, continuando por las etapas de construcción y pruebas, hasta la etapa final de subida a producción. En cada uno de estos procesos se verán conceptos como por ejemplo las estrategias a la hora de definir o valorar una user story, la gestión de entornos, la elaboración de planes de pruebas o los requisitos para pasar a producción, todos ellos desde un enfoque ágil y siguiendo el manual de buenas prácticas recomendado desde la propia plataforma de Salesforce [20].

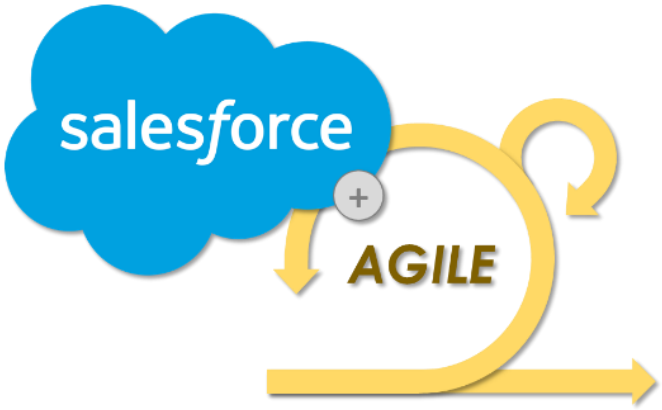

Figura 4.1 – Salesforce + Agile (Fuente: elaboración propia)

4.1 CRM

En la actualidad, las empresas y la sociedad en general están inmersos en una etapa de revolución digital, donde el acceso a la información está al alcance de cualquiera y las comunicaciones son cada vez más inmediatas.

Esto supone una enorme transformación en el mundo del marketing y la relación con los clientes en donde se ha pasado de un modelo de grandes campañas en las que el presupuesto era determinante, a una mayor capacidad de análisis y de medición de resultados, al alcance de casi cualquier bolsillo. Sin embargo, al extenderse estas capacidades, la competencia también es mayor, por ello surgen herramientas como los CRM, que permiten a las empresas ser más competitivas y adaptarse a estos cambios.

4.1.1 Qué es un CRM

CRM son las siglas en inglés de Customer Relationship Management que significa Gestión de las Relaciones con los Clientes. Este término generalmente suele referirse al CRM como un software, sin embargo, este término también se utiliza para definir un modelo de gestión. Es necesario conocer en qué consiste este modelo para entender el CRM como software.

Este modelo de gestión apuesta por situar al cliente en el centro de las nuevas estrategias comerciales con la finalidad de diferenciarse de otras empresas, estudiando sus preferencias con el fin de satisfacer sus necesidades de una forma mucho más precisa. El objetivo de este modelo es doble; por un lado la fidelización, basándose en la máxima de que un cliente fidelizado es más beneficioso que un nuevo cliente, y por otro la captación de nuevos clientes. Por lo tanto, un CRM no sólo busca ser una base de datos donde almacenar información de los clientes, sino que además debe servir como herramienta de análisis y optimización que permita segmentar los clientes para proporcionarles un trato más personalizado y poder así tomar las mejores decisiones en el presente y en el futuro.

Los software CRM sirven principalmente para gestionar los procesos denominados *front office* que comprenden principalmente tres áreas: el marketing, la gestión comercial (ventas) y por último la atención al cliente (posventa). Además, incorporan otras herramientas como las de análisis y reporting[10] para ayudar a mejorar el control y la gestión de los procesos, entre otras.

Estas herramientas pueden registrar todo lo relacionado con la interacción con el cliente, como el seguimiento de las carteras, las visitas, los emails, los clientes potenciales (también llamados leads) o la implementación de diferentes flujos de venta. Todo ello dependerá de la necesidad de cada negocio y las capacidades concretas de la herramienta que esté utilizando.

Figura 4.2 – Áreas de un CRM (Fuente: elaboración propia)

[10] Reporting: generación de informes

Según la localización del servidor podemos distinguir principalmente dos tipos:

- CRM On-Premise: basado en el modelo tradicional de implementación de software de forma local en servidores internos. En este modelo el control, el mantenimiento y las actualizaciones del sistema recaen por completo en la propia empresa.
- CRM On-Demand: también denominado CRM online o Cloud[11] CRM. Está basado en el modelo SaaS (Software as a Service), que significa software como servicio y que consiste en externalizar tanto los servidores como el mantenimiento de las aplicaciones software que se necesiten. Es un tipo de CRM al que se accede a través de internet y que suele funcionar con un sistema de suscripción por licencia de usuario.

La elección de un tipo u otro de CRM dependerá como es lógico de las necesidades de cada empresa, por lo que es importante conocer las ventajas e inconvenientes que presenta un modelo frente al otro. A continuación se muestra una tabla que resume las principales características de ambos modelos:

Funcionalidad	Modelo On-Premise	Modelo SaaS
Localización de los servidores	Interno, alojados en servidores locales	Servidores externos, en la nube
Acceso	Limitado por la necesidad de tener una conexión con el servidor interno	Acceso mediante internet, generalmente disponible en cualquier momento
Mantenimiento	La instalación, el mantenimiento y las actualizaciones corren a cargo de la propia empresa	El mantenimiento y las actualizaciones lo realiza el proveedor externo
Personalización	Configuración totalmente personalizada ya que el software está hecho a medida	Depende del proveedor y de las herramientas y capacidades de personalización que éste ofrezca
Seguridad	Se tiene un control total, pero depende la inversión realizada en seguridad y de la capacidad de la propia empresa para hacer frente a los posibles ataques	Dependiente completamente del proveedor, lo que, aunque en un principio pueda parecer un riesgo, puede convertirse una ventaja especialmente en los proveedores más consolidados en el tema de la seguridad
Costes	Se necesita una inversión inicial para la instalación, más los costes de seguridad mantenimiento y actualización	Suele funcionar con un sistema de licencias por usuario

Tabla 4.1 – Modelo On-Premise vs SaaS (Fuente: elaboración propia)

En la actualidad, las ventajas que ofrecen los CRM Cloud frente a los locales los hacen más demandados, ya que permiten a las empresas hacer uso de este tipo de softwares in tener que preocuparse de la instalación y el mantenimiento del mismo. Por este motivo este documento se centrará especialmente en estos últimos.

[11] Cloud: nube en inglés, término utilizado para referirse a los sistemas gestionados a través de internet en servidores externos

4.1.2 Principales CRM SaaS del mercado

En la actualidad, cada vez proliferan más empresas dedicadas a ofrecer herramientas CRM basadas en el modelo SaaS en la nube. A continuación se presentarán algunas de las más populares junto a sus características principales. En este listado no se incluye Salesforce, ya que se tratará en un apartado específico posterior de una forma más completa.

4.1.2.1 Hubspot

Hubspot CRM es un software que posee un módulo de ventas gratuito y que proporciona al equipo de ventas las aplicaciones y herramientas que necesitan en sus procesos de negociación. Permite almacenar toda la información relativa a clientes, contactos y negociaciones. Además, también tiene la capacidad de registrar tareas que ayuden a la planificación de los usuarios.

Fue fundada en el 2006 por Brian Halligan y Dharmesh Shah con el fin de implementar nuevas estrategias de marketing basadas en lo que ellos llaman Inbound Marketing. Se basa en la filosofía de atraer al cliente en base a contenidos en lugar de imponerle una publicidad cuando está consumiendo otro producto, como ocurre con el marketing tradicional u Outbound Marketing, al que consideran obsoleto.

Figura 4.3 – HubSpot

4.1.2.2 SugarCRM

SugarCRM es un software de código abierto para gestionar los procesos relacionados con los clientes. Permite un nivel de personalización muy alto y tiene una gran capacidad de integración con otros sistemas. Desde esta plataforma se pueden gestionar las tres áreas del CRM, marketing, ventas y atención al cliente. Esto lo hace a través de sus tres módulos principales: Sugar Market para el marketing, Sugar Sell para la gestión del proceso de ventas y Sugar Serve para los procesos de atención al cliente.

Según su página web [21], existen diferentes planes cuyos precios actuales parten de los 52$ al mes por usuario para contratos anuales, sin embargo, se debe tener en cuenta que al menos se deben contratar 10 usuarios.

Figura 4.4 – SugarCRM

4.1.2.3 Zoho CRM

Zoho CRM es una herramienta enfocada en la gestión de clientes y seguimiento de las ventas. Permite diseñar y automatizar procesos de venta para optimizarlos y mejorar la productividad. Además, también permite la integración con diferentes herramientas del propio proveedor o externas como pueden ser Outlook o GSuite (Google Apps).

Al igual que SugarCRM, este software propone varios paquetes que se ajusten a las diferentes empresas y sus necesidades. Los precios oscilan entre 12 y 45 euros según su página web [22] dependiendo del paquete por el que se opte.

Figura 4.5 – Zoho CRM

4.2 Introducción a Salesforce

Según la definición que aparece en su web [23] "Salesforce es una plataforma de gestión de las relaciones con los clientes (CRM) basada en la nube que proporciona a todos los departamentos de su organización, incluidos los de marketing, ventas, servicio al cliente y comercio electrónico, una visión unificada de sus clientes en una plataforma integrada"

4.2.1 Historia de la compañía

Salesforce comenzó su andadura en 1999 de la mano de Marc Benioff, exdirectivo de Oracle, compañía en la que trabajó durante 13 años y de la que llegó a ser el vicepresidente más joven hasta la fecha [24]. Su objetivo era crear aplicaciones de software empresarial, a través del modelo conocido como Software-as-a-Service (SaaS), que fueran sencillas de utilizar y que no necesitaran una instalación y un mantenimiento muy costosos.

Benioff inició este proyecto el 8 marzo en un apartamento de una habitación en San Francisco junto con otros tres compañeros: Parker Harris, Frank Domínguez y Dave Moellenhoff, desarrolladores especializados en SaaS [25]. El primer prototipo apareció tras el primer mes de desarrollo y tenía como inspiración el aspecto y la usabilidad de la web Amazon, basada en pestañas situadas en la parte superior de la pantalla. En julio de ese mismo año la empresa ya contaba con 10 empleados y una nueva oficina en Rincon Center.

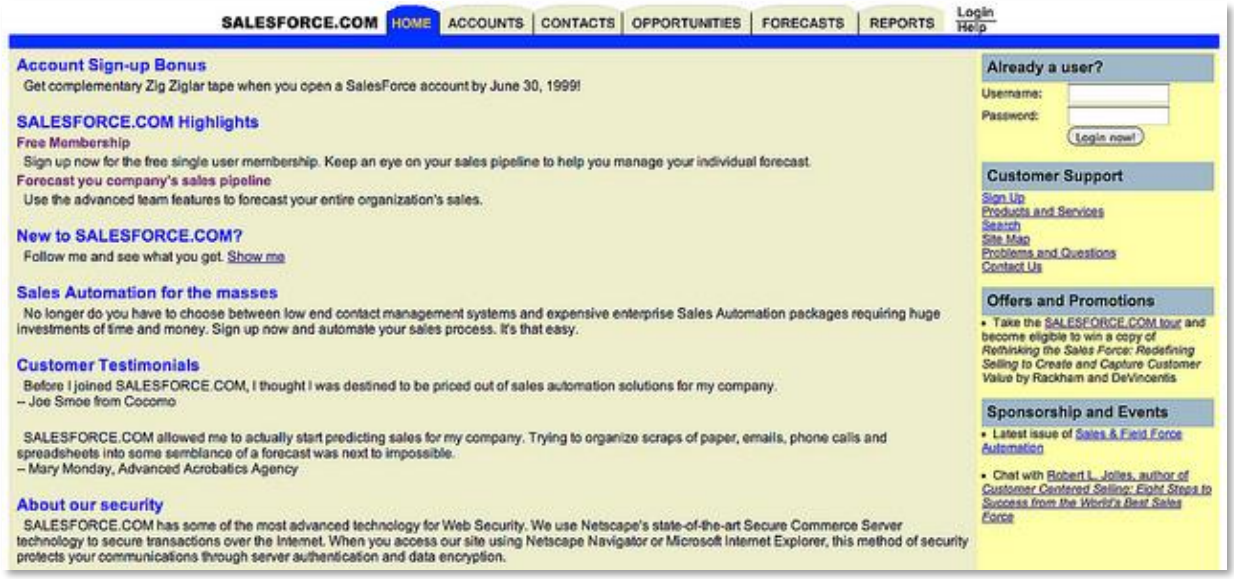

Figura 4.6 – Primer prototipo de Salesforce (Fuente: [26])

Para noviembre del año siguiente el número de empleados había aumentado considerablemente y se trasladaron unas oficinas en One Market Street. Sin embargo, necesitaban darse a conocer en el mundo empresarial y para ello lanzaron la campaña de marketing "The End of Software" con carteles y pegatinas con el mensaje "No Software", para indicar que el modelo de software tradicional On-Premise quedaba obsoleto.

Figura 4.7 – Campaña de Salesforce "No Software" (Fuente: [27])

En el año 2003 organizaron su primer evento a gran escala en Westin St.Francis de San Francisco, con una duración de varios días con más de 50 presentaciones y mil inscripciones, denominado Dreamforce. Este evento se sigue organizando hoy en día y ha contado en el 2019 con la participación de más de 170.000 participantes de 120 países tal y como indican desde su propia página web [28].

En el 2005 lanzó la primera tienda de aplicaciones de la historia que denominó Salesforce App Exange [29], una plataforma donde los usuarios podían descargar e instalar numerosas aplicaciones de software. Este modelo sirvió como inspiración para la actual tienda de aplicaciones de Apple, la AppStore. De hecho, fue Salesforce quién inicialmente registró la URL de la App Store pero se la acabó cediendo a Apple debido a la buena relación que existía entre los directivos de ambas compañías [30].

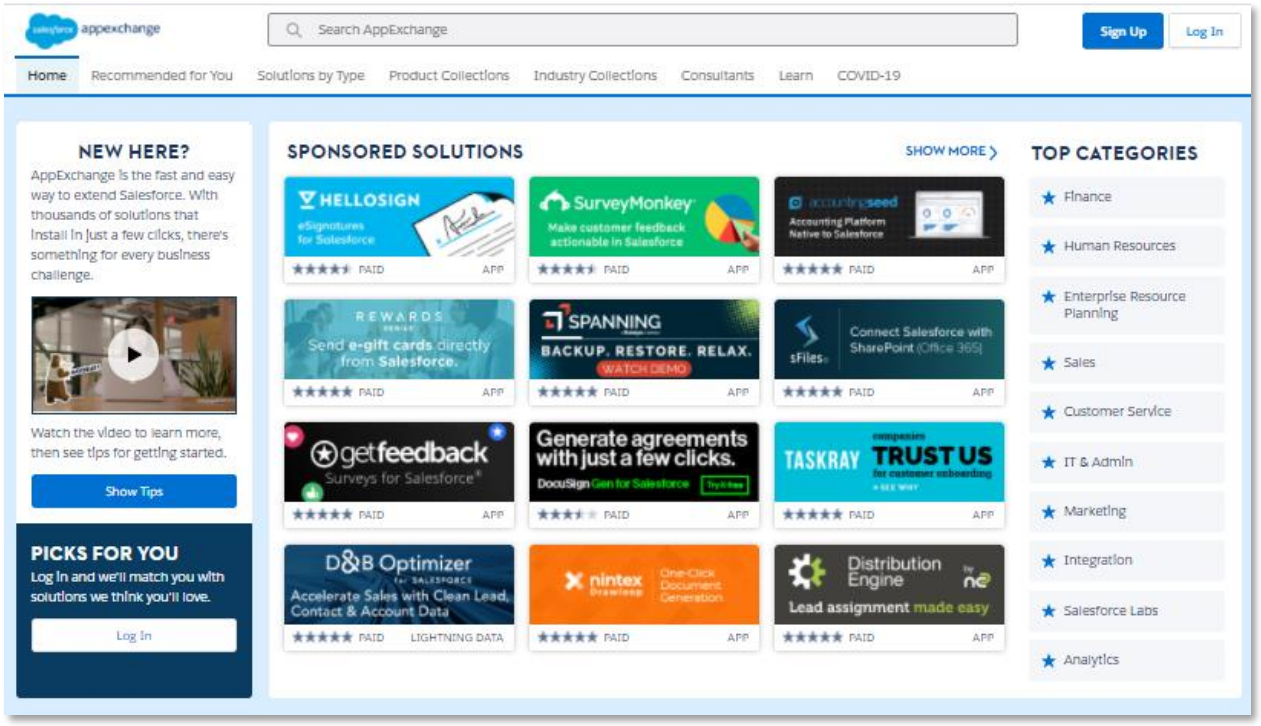

Figura 4.8 – Salesforce AppExchange (Fuente: [29])

Además de los logros mencionados, desde entonces Salesforce ha ido creciendo y asociándose con diferentes empresas para ampliar sus capacidades y poder llegar a un público más amplio. Convirtiéndose así en una de las empresas más valoradas en la actualidad con más de 50.000 empleados y presencia en los 5 continentes.

Logros de Salesforce 1999-2019	
1999	Lanzamiento de Salesforce en San Francisco
2000	Lanzamiento del producto y debut de "No Software"
2003	El primer evento Dreamforce
2004	Salesforce Ohana (filosfía de empresa)
2005	Lanzamiento de AppExchange
2006	Desarrollo de Apex, Visualforce y más allá
2012	Introducción de Salesforce Marketing Cloud
2013	Salesforce para móviles
2014	Lanzamiento de la plataforma de aprendizaje Trailhead
2015	Lanzamiento de la nueva Salesforce Lightning Experience
2016	Inteligencia artificial y adquisición de más de 10 empresas
2018	Introducción de Mulesoft
2019	Adquisición de Tableau

Tabla 4.2 – Logros de Salesforce 1999-2019 (Fuente: [27])

4.2.2 Módulos de Salesforce

Salesforce se estructura en torno a diferentes módulos o plataformas, conocidas también como Clouds (nubes) en inglés, enfocados a cada una de las áreas involucradas en la gestión de los clientes. A continuación, se enumera cada una de ellas explicando en qué consiste.

Figura 4.9 – Clouds de Salesforce (Fuente: [31])

- **Sales Cloud**: es el módulo más popular y empleado de Salesforce, enfocado en la gestión de todo el proceso de la venta, desde el registro de contactos y oportunidades hasta el cierre de las negociaciones. Permite hacer un seguimiento en tiempo real de todo el pipeline de ventas[12].
- **Service Cloud**: diseñado para administrar los procesos relacionados con la atención al cliente. Permite que el cliente pueda contactar con la empresa usando diferentes vías según las necesidades de cada negocio, como puede ser por teléfono, SMS, email, formulario en una página web, o chat en tiempo real entre otros.

  Además, permite automatizar ciertos procesos y hacer el seguimiento de todo el proceso de comunicación en una misma herramienta, dando así una respuesta más rápida y satisfactoria, y mejorando la productividad y la satisfacción del cliente.
- **Marketing Cloud**: este módulo permite personalizar los flujos de comunicación con los clientes potenciales y actuales para captar nuevas oportunidades de negocio. Permite crear potentes campañas con una participación activa de los clientes a través de emails, redes sociales, publicidad digital, etc.
- **Community Cloud**: con este módulo se pueden crear communities o comunidades personalizadas, que son plataformas conectadas con el propio sistema que permiten a usuarios externos acceder a los datos de la organización en Salesforce, controlando los permisos de acceso y las capacidades de interacción con el sistema.
- **App Cloud**: esta herramienta permite crear aplicaciones personalizadas de una forma mucho más sencilla y práctica
- **Analytics Cloud**: también conocido como Einstein Analytics, sus aplicaciones tienen conexión directa con Salesforce y permiten explotar los datos y la información almacenada en el sistema a través de informes y paneles dinámicos, facilitando la toma de decisiones en base a los datos.

4.2.3 Presupuestos y licencias

Salesforce proporciona diferentes licencias para ajustarse a las dimensiones, el presupuesto y las necesidades de cada empresa. Según su página web, se pueden encontrar cuarto licencias diferentes para sus dos módulos principales: Sales Cloud [32] y Service Cloud [33]. La descripción que proporcionan de cada una de ellas es la siguiente, en orden ascendente de funcionalidades y capacidades incluidas:

- **Essentials** (25 € usuario/mes para facturación anual): "CRM disponible en cuestión de segundos para equipos de hasta 10 usuarios"

[12] Pipeline de ventas: herramienta de gestión para hacer seguimiento de todas las etapas de los ciclos de ventas

- **Professional** (75 € usuario/mes para facturación anual): "Servicio de CRM completo para equipos de cualquier tamaño"
- **Enterprise** (150 € usuario/mes para facturación anual): "CRM personalizable para ofrecer un servicio integral"
- **Unlimited** (300 € usuario/mes para facturación anual): "Potencia de CRM ilimitada"

A continuación, se muestran unas tablas comparativas con las capacidades que ofrece cada licencia según el módulo que corresponda.

salesforce sales cloud	Essentials € 25 usuario/mes*	Professional € 75 usuario/mes*	Enterprise € 150 usuario/mes*	Unlimited € 300 usuario/mes*
Gestión de cuentas, contactos, candidatos y oportunidades	✓	✓	✓	✓
Integración de correo electrónico con Outlook o Gmail	✓	✓	✓	✓
Aplicación móvil Salesforce	✓	✓	✓	✓
Registro de candidatos y puntuación de candidatos basada en reglas	✗	✓	✓	✓
Previsiones colaborativas	✗	✓	✓	✓
Automatización de flujos de trabajo y aprobaciones	✗	✗	✓	✓
Servicios ininterrumpidos de asistencia y configuración	✗	✗	✗	✓

Tabla 4.3 – Comparativa de precios Sales Cloud (Fuente: [32])

salesforce service cloud	Essentials € 25 usuario/mes*	Professional € 75 usuario/mes*	Enterprise € 150 usuario/mes*	Unlimited € 300 usuario/mes*
Gestión de casos	✓	✓	✓	✓
Aplicaciones de consola de servicio	✓	✓	✓	✓
Base de conocimientos	✓	✓	✓	✓
Asignaciones y contratos de servicio de atención al cliente	✗	✓	✓	✓
Integración de telefonía (CTI)	✓	✓	✓	✓
API de servicios web	✗	✗	✓	✓
Servicios ininterrumpidos de asistencia y configuración	✗	✗	✗	✓

Tabla 4.4 – Comparativa de precios Service Cloud (Fuente: [33])

4.3 Gestión de un proyecto en Salesforce

En este apartado se revisarán los puntos ya vistos en los capítulos anteriores desde el punto de vista de Salesforce, incluyendo desde la selección de la metodología más conveniente, hasta las capacidades que ofrece la herramienta y la gestión de entornos que debe seguirse para los diferentes procesos.

4.3.1 Qué metodología emplear y porqué

Antes de afrontar un nuevo proyecto de software es necesario determinar qué enfoque se quiere seguir y qué metodología es la más adecuada. Esta decisión depende en gran medida de la tecnología o tecnologías sobre las que se va a implementar la solución. Pero ¿qué metodología es la más recomendable para un proyecto desarrollado en Salesforce? esta pregunta es la que se responderá en este apartado.

En el caso de Salesforce, desde la propia plataforma recomiendan gestionar los proyectos en base al modelo Agile [34]. Esto es debido a varios factores:

- Salesforce mantiene una política de lanzar 3 actualizaciones de su software al año: Spring, Summer y Winter. Lo que encaja perfectamente con el modelo de entregas por iteraciones o sprints en el caso de Scrum, ya que permite sacar el máximo partido a las novedades que se incluyen en la herramienta y adaptar los desarrollos conforme ésta evoluciona.
- El modelo SaaS permite que se puedan acortar los periodos de entrega ya que el mantenimiento del software se gestiona de forma controlada e independiente al desarrollo de los productos
- En un sistema tan competitivo como el de la actualidad, un CRM tiene que estar preparado para responder a las exigencias del cliente en el menor tiempo posible. Por ello es necesario que la implantación de nuevas soluciones sea rápida y flexible.
- Además, Salesforce proporciona una aplicación, disponible en el AppExange que ayuda a la gestión de proyectos en Agile llamada Agile Accelerator
- A diferencia de otras plataformas en Salesforce es relativamente sencillo desplegar una solución en el entorno productivo (aplicación disponible para el usuario). Lo que facilita que se puedan hacer entregas en plazos más ajustados sin que estos procesos impacten en la planificación.
- Según el tipo de licencia, Salesforce ofrece la posibilidad de tener varios entornos que permitan desarrollar las soluciones, realizar pruebas y entregar los productos sin que estos procesos impacten unos con otros. Lo que permite que no sea necesario terminar una etapa para empezar la siguiente, sino que se pueden gestionar diferentes desarrollos en paralelo.

Por estos motivos se concluye que los marcos de trabajo ágiles son los más recomendables para llevar a cabo un proyecto en Salesforce. Aunque esto no siempre sea posible ya que, además de estos factores, en la decisión de adoptar un modelo u otro también influye la capacidad que tenga la empresa de adoptar este modelo, el grado de conocimiento de la solución y si intervienen otras tecnologías en el proyecto.

Dentro de los marcos de trabajo más recomendables para Salesforce destacan principalmente dos: Scrum y Kanban. Pero ¿cuándo es más recomendable emplear uno u otro? La respuesta a esta pregunta dependerá de la naturaleza del proyecto y de si éste está sometido a interrupciones o puede seguir una planificación determinada.

Para el caso de los proyectos en los que las necesidades vengan en un flujo continuo y necesiten mucha flexibilidad a fin de interrumpir los desarrollos y reorganizar las tareas, es más recomendable el empleo de Kanban. Esto puede ocurrir por ejemplo en un proyecto de soporte a incidencias donde van llegando los requerimientos de forma continua y se deben solucionar en función de su criticidad.

Sin embargo, en proyectos donde se conozcan de antemano las funcionalidades que se quieren implementar, aunque no sea necesario tener todo el detalle, suele ser más eficiente utilizar el marco Scrum. Esto ocurre en proyectos de implementación de un nuevo flujo cuyo funcionamiento a alto nivel se puede conocer de antemano.

Se debe tener en cuenta que estos dos marcos de trabajo no son incompatibles por lo que muchos equipos deciden implementar un modelo mixto: el Scrumban. Este modelo permite mantener la estructura de la revisión y planificación regular que proporciona Scrum, y a la vez hacer uso de los límites de trabajo en curso de Kanban

4.3.2 Generación y valoración del Product Backlog

Como se ha explicado en capítulos anteriores, antes de poder comenzar cualquier proyecto es fundamental establecer el alcance de éste, es decir, determinar que se va a hacer y de qué forma. Esto se traduce principalmente en cuatro procesos: la toma de requerimientos, definición de los criterios de aceptación, valoración. y finalmente, la definición y planificación del alcance priorizando en base a estos parámetros.

En la metodología Scrum los primeros dos procesos son responsabilidad de Product Owner y del cliente, y para ello, hacen uso del Product Backlog, que recoge un conjunto de necesidades en forma de user stories. Aunque el formato y el proceso de recopilación de los requerimientos en Scrum es independientemente de la tecnología que se quiera emplear, desde el punto de vista de Salesforce se pueden seguir una serie de recomendaciones a la hora de redactar las user stories que facilitarán su comprensión y la construcción de la solución

El tercer proceso lo realiza generalmente el equipo de desarrollo, que debe tener un conocimiento amplio y profundo de las capacidades que ofrece Salesforce y la complejidad que implica un requerimiento en función de la solución que se adopte.

Para ofrecer una visión global de esta etapa inicial, en este apartado se abordarán estos procesos. Proporcionando una serie de recomendaciones a la hora de redactar las user stories y reglas que pueden ayudar a determinar la complejidad de la solución.

4.3.2.1 Generación y redacción de las user stories

El formato y el proceso de recopilación de las user stories debe ser el definido en el apartado 2.3.1 de este documento, cuya estructura sigue el siguiente esquema: *Como [usuario] quiero [necesidad] para [beneficio]*. Se debe tener en cuenta que una user story es un **requerimiento funcional y no técnico**, por lo que se debe prestar atención a la hora de redactarlas para no caer en ese error. Esto ocurre especialmente cuando el Product Owner tiene un conocimiento técnico alto de la herramienta y pierde de vista el verdadero propósito de las historias de usuario. Sin embargo, es recomendable emplear términos que sean fácilmente trasladables al ámbito técnico, sin perder ese carácter funcional.

Un ejemplo claro de la utilidad de este enfoque es el uso de los perfiles y roles para indicar el *usuario* en la estructura de la user story. Ya que el *usuario* se refiere al papel que la persona desempeña dentro de la organización, que es precisamente el propósito de estas dos herramientas.

Como ejemplo se puede poner el caso de dos user stories enfocadas a un gestor de ventas y a su responsable, en las que el perfil sería “Fuerza de ventas” y los roles “Gestor” y “Responsable” respectivamente. Las historias de usuario quedarían de la siguiente forma:

- US 1: Cómo Gestor de la Fuerza de ventas quiero [necesidad 1] para [beneficio 1].
- US 2: Cómo Responsable de la Fuerza de ventas quiero [necesidad 2] para [beneficio 2].

De esta forma se conoce de antemano para qué perfil y rol se debe construir y probar la solución. Se recomienda generar una user story diferente para cada uno de los perfiles que necesite hacer uso de la funcionalidad, aunque para todos sea la misma. De lo contrario se puede caer en el error de realizar las pruebas únicamente con uno sólo de los perfiles, con lo que podrían aparecer errores por falta de permisos en el resto de los perfiles una vez la solución ya ha sido entregada.

Se aconseja que la segunda parte de las user stories sea lo más sencilla posible, la mínima unidad que aporte valor añadido. Es este último punto al que hay que prestar especial atención, el valor añadido. Esto quiere decir que cada user story debe representar una funcionalidad en sí misma, que se pueda probar y entregar de forma independiente y que no requiera de otras user stories para completarse en la medida de los posible. Además, también es conveniente no solo tener en cuenta el flujo positivo (acciones que se deben hacer), sino también incluir el negativo (restricciones que impiden llevar a cabo esa acción)

En base esto, cuando una historia es excesivamente compleja, se recomienda desglosarla en otras más sencillas. Para ello se pueden usar diferentes criterios, que dependerán de las personas encargadas de redactarlas.

Un ejemplo podría ser el siguiente: Se quiere registrar un cliente en el sistema teniendo en cuenta una serie de criterios que aseguren que la calidad del dato es suficiente. Para llevar este caso a formato user story se pueden usar diferentes maneras como las que se muestran en la siguiente tabla.

Opción	User Stories	Comentario
1	· *Cómo* [rol usuario] · *quiero* poder registrar un cliente · *para* almacenar en el sistema la información relativa a éste	Esta user es bastante genérica, por lo que los criterios de aceptación probablemente sean muy extensos. Además, es probable que sea fácil olvidarse de incluir las restricciones.
2	· *Cómo* [rol usuario] · *quiero* poder registrar un cliente comprobando que los datos cumplen unos criterios de calidad · *para* almacenar en el sistema la información relativa a éste	Esta user es algo más completa que la anterior, pero sigue abarcando bastante funcionalidad. Por lo que los criterios de aceptación seguirán siendo bastante extensos
3	· *Cómo* [rol usuario] · *quiero* poder registrar un cliente · *para* almacenar en el sistema la información relativa a éste	En este caso la user es la misma que en la opción 1, sin embargo, ya no es necesario incluir todos los criterios de aceptación relativos al flujo negativo ya que se incluyen en la siguiente user story
	· *Cómo* [rol usuario] · *quiero* que el sistema no me permita registrar un cliente si los datos no cumplen los criterios de calidad suficientes · *para* evitar almacenar información errónea o incompleta	Las validaciones con respecto al flujo negativo quedarán recogidas en los criterios de aceptación de esta user story

Tabla 4.5 - Ejemplo granularidad user stories (Fuente: elaboración propia)

En el ejemplo anterior las tres opciones pueden ser igualmente aceptables, sin embargo, en la tercera al desglosar la funcionalidad en dos historias, es más sencillo abordar tanto su construcción como las pruebas relativas a cada una de ellas por separado.

4.3.2.2 Valoración de la complejidad de la solución

Uno de los criterios principales para identificar el grado de complejidad es determinar si se puede implementar la solución de forma declarativa o programática. Un desarrollo declarativo es todo aquel que no necesita de un código informático para su ejecución, sino que se puede implementar haciendo uso de las herramientas que el propio Salesforce pone a disposición del desarrollador. Esto es lo que se conoce como tareas de tipo *admin*[13]. El desarrollo programático es aquel que si necesita hacer uso de lenguajes de programación. Por regla general, los desarrollos programáticos suelen ser más costosos de implementar y mantener ya que suelen implicar un grado más alto de personalización.

A continuación, se verán una serie de reglas que pueden ayudar a la hora de determinar qué implicaciones tiene un requerimiento concreto y que complejidad puede tener una solución frente a otra. Estas reglas no son una ciencia exacta que se cumpla en el 100% de los casos, pero pueden ser útiles para determinar la complejidad en un gran número de ellos.

- ✓ Si el objetivo contiene las palabras registrar, almacenar o crear y/o se hace referencia a cumplimentar un formulario, probablemente se pueda implementar de forma declarativa. Aunque se debe estudiar si es necesario validar los datos y contemplar los flujos negativos, ya que la complejidad aumentaría.
- ✓ Si se necesita una validación únicamente entre los datos del propio registro es más sencillo que si se requiere hacer comparaciones con otros registros ya que estas últimas comprobaciones deben hacerse generalmente haciendo uso de la programación.

[13] Admin: abreviatura de administrator (administrador en inglés)

- ✓ Si se hace referencia a niveles de visibilidad entre usuarios con una jerarquía determinada, es decir, los que están en un nivel jerárquico superior deben visualizar los registros de los que están por debajo, la funcionalidad estándar cubre la solución. En cambio, si se necesita una configuración muy compleja habrás que hacer desarrollos específicos para poder implementarlas.
- ✓ Salesforce ofrece una interfaz estándar y una serie de opciones para personalizar sin necesidad de hacer uso de la programación. Sin embargo, si se necesita que la interfaz de usuario tenga un aspecto determinado, la complejidad aumentará considerablemente.

Como conclusión, se puede determinar que los desarrollos declarativos son más sencillos de implementar ya que no requieren tener habilidades de programación y se pueden configurar con relativa facilidad. Además, cuanto más uso se haga de las capacidades estándar que ofrece Salesforce, más sencillo será tanto el desarrollo como el mantenimiento. En el siguiente apartado se presentarán de forma más detallada las diferentes herramientas en función de su finalidad.

4.3.3 Desarrollo de la solución

Para poder realizar cualquier desarrollo es necesario conocer las herramientas y capacidades que ofrece Salesforce. Aunque esta plataforma proporciona infinidad de herramientas para conseguir personalizar e implementar incluso los procesos más complejos, en este apartado se mostrarán los conocimientos básicos que permiten sustentar la mayoría de las soluciones.

Para los interesados en usar estas herramientas y conocer un poco más sobre las capacidades de Salesforce, la propia plataforma ofrece de forma gratuita desde su web unos módulos de aprendizaje denominados Trailheads [9]. Dentro de estos módulos se podrán encontrar diferentes rutas dependiendo de la función que se desee estudiar y del nivel que se tenga. Las principales funciones para un usuario que configure Salesforce son la de administrador y desarrollador. Además, también existen módulos para el uso de la plataforma a nivel de usuario.

4.3.3.1 Modelos de datos: objetos y campos

Todo el modelo de datos de Salesforce está basado en una estructura de objetos, campos y registros, que representan tablas de datos. Estas tablas podrán consultarse mediante consultas SOQL (Salesforce Object Query Language) que es similar al lenguaje SQL[14] pero adaptado a Salesforce.

Clientes				
Id	Nombre	Dirección	Teléfono	Página web
2548158	Edge Communications	312 Constitution Place Austin, TX 78767, USA	(512) 757-6000	http://edgecomm.com
2700710	Grand Hotels & Resorts Ltd	2334 N. Michigan Avenue, Suite 1500 Chicago, IL 60601, USA	(312) 596-1000	www.grandhotels.com
2738848	United Oil & Gas Corp.	1301 Avenue of the Americas New York, NY 10019, USA	(212) 842-5500	http://www.uos.com
2776986	Express Logistics and Transport	620 SW 5th Ave. Suite 400 Portland, Oregon 97204 , USA	(503) 421-7800	www.expressl&t.net
2815124	University of Arizona	888 N Euclid Hallis Center, Room 501 Tucson, AZ 85721, USA	(520) 773-9050	www.universityofarizona.com

Figura 4.10 – Modelo de datos (objetos, campos y registros) (Fuente: elaboración propia)

Los **objetos o entidades** son las diferentes tablas que existen en el sistema y que se relacionan entre sí mediante campos que hacen referencia a otras tablas. Estos objetos pueden ser de tipo estándar, predeterminados por el sistema, como son los clientes, las oportunidades o los contactos, o personalizados, también llamados custom, para incluir tablas nuevas.

[14] SQL: lenguaje de programación diseñado para consultas en bases de datos relacionales

Los **campos** representan las columnas de la tabla, es decir los diferentes datos. En el caso de los clientes, los campos podrían ser el nombre, la dirección, el número de teléfono, etc. Estos campos pueden tener diferentes formatos (texto, numérico, booleano, fórmula…) y pueden hacer referencia a otras tablas creando relaciones de tipo 1:N. Estas relaciones pueden ser de dos tipos:

- Relaciones Master-Detail: relaciones entre dos objetos en las que uno es el padre (master) y el otro el hijo (detail). En este tipo de relaciones entre objetos para poder crear un registro hijo es necesario que exista el padre y en caso de que se elimine éste, el hijo también se eliminaría. Por cada objeto sólo se pueden crear 2 relaciones de este tipo como máximo.
- Relaciones Lookup: Relaciones entre dos objetos en el que uno (hijo) hace referencia al otro (padre). A diferencia de las relaciones master-detail, para crear un registro hijo no es obligatorio que haya un registro padre al que asociarlo. Además, si éste último se elimina, el hijo permanece en el sistema. Por cada objeto sólo se pueden crear 25 relaciones de este tipo como máximo.

Los **registros** serían los datos que se quieren almacenar en el sistema, en este caso los propios clientes con toda su información. Esto correspondería a las filas de la tabla. Todos los registros quedarán identificados de forma única mediante un código alfanumérico de 18 cifras generado de forma automática por el sistema, es lo que se conoce como *Id*.

4.3.3.2 Visibilidades y permisos

Salesforce permite asignar a los usuarios distintos grados de acceso tanto a los objetos y campos como a los registros, esto lo hace principalmente a través de la configuración de *roles* y *perfiles*. Cada usuario debe tener asignados obligatoriamente un rol y un perfil. Para entender la diferencia entre uno y otro, hay que saber diferenciar entre los permisos sobre objetos y campos y los accesos a los registros.

Los **permisos sobre objetos** y campos son los que permiten hacer operaciones CRUD[15], como crear, leer, modificar o eliminar datos de las tablas. En Salesforce estos permisos los proporcionan los perfiles, que permiten definir permisos tanto a nivel de objetos como a nivel de campos. En el caso en el que dentro de un mismo perfil exista un grupo de usuarios que necesiten permisos adicionales, se hace uso de los *Permission Sets*, que se pueden configurar de la misma forma que los perfiles y actúan como extensiones de ellos. Los **permisos sobre registros** son los que te permiten acceder a los datos dentro de las tablas.

Por lo tanto, para que un usuario vea, por ejemplo, los clientes que tiene en su cartera, pero no los de los demás, será necesario que pueda acceder a la pestaña de clientes (permiso sobre el objeto cliente) y que dentro de los clientes pueda ver sólo los que le pertenecen (permisos sobre los registros de cliente).

En Salesforce el acceso a los registros puede gestionarse de diferentes maneras. Un usuario puede ser propietario de un registro y por lo tanto tendrá acceso a él. Además, si se quiere proporcionar el acceso puntual a un registro concreto, se pueden utilizar las *Custom Permisions*. Otra forma de acceder a un registro es mediante las *Reglas de Compartición*, que permiten a los usuarios acceder a los registros que cumplan ciertos criterios.

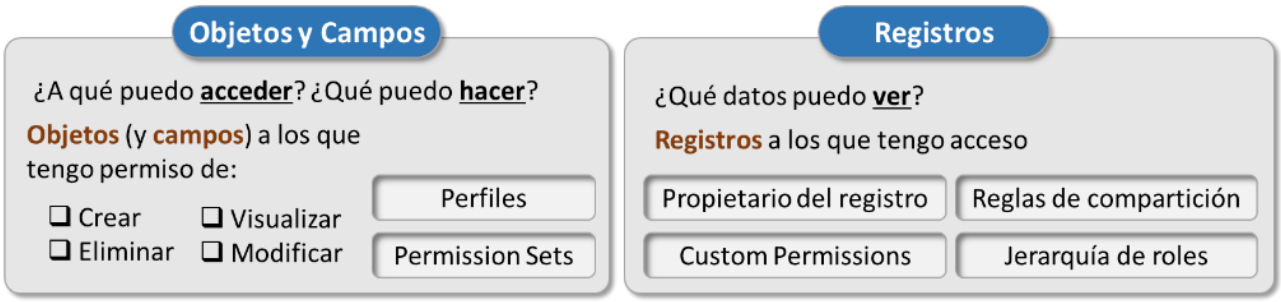

Figura 4.11 – Permisos sobre objetos y registros (Fuente: elaboración propia)

[15] CRUD: Siglas en inglés de Create, Read, Update y Delete. Definen las operaciones de lectura y escritura en la base de datos.

Además, en Salesforce se puede establecer una jerarquía de permisos en base a los roles de cada usuario. De esta forma es posible proporcionar a los usuarios que están en niveles superiores sobre los registros que pueden ver los niveles inferiores de la jerarquía, nunca al revés ni sobre los registros de los usuarios que comparten rol.

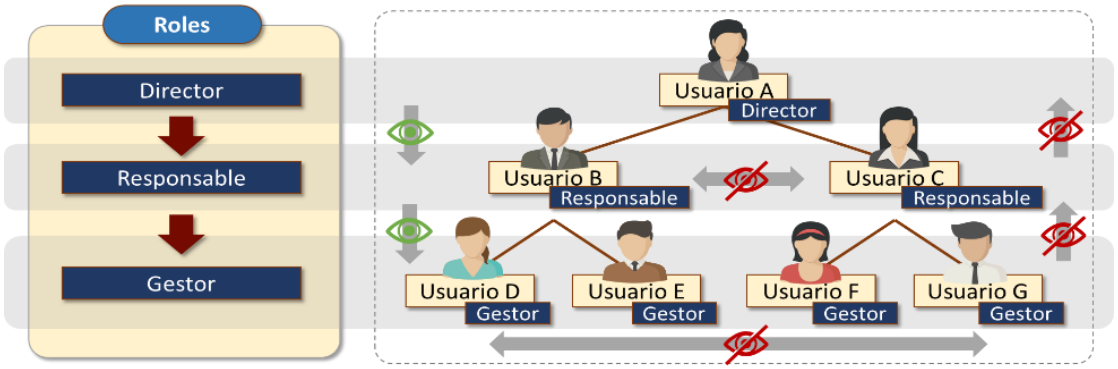

Figura 4.12 – Acceso a registros mediante roles (Fuente: elaboración propia)

4.3.3.3 Flujos y procesos personalizados

Para poder gestionar los datos e implementar procesos, Salesforce proporciona de una serie de herramientas que permiten personalizar los flujos y la gestión de los datos en el sistema. A continuación, enumeraremos y describiremos las herramientas principales que ofrece Salesforce en función de su naturaleza.

Flujos y lógica de procesos

- Reglas de validación: conjunto de condiciones que deben cumplir los datos de un registro para asegurar la calidad y coherencia de la información. En caso de que una regla no se cumpla aparecerá y mensaje de error y el sistema no permitirá crear o modificar el registro. No necesitan programación, pero si tener conocimientos en el manejo de fórmulas. (Declarativo)
- Workflow Rules: también llamadas Reglas de flujo de trabajo. Son procesos que se lanzan cuando se crea o se edita un registro y que permiten realizar una o varias acciones si se cumplen unos criterios determinados. No necesitan programación, pero si tener manejo de fórmulas. (Declarativo)
- Process Builder: proceso que se lanza cuando se crea o se edita un registro y que permite realizar una o varias acciones si se cumplen unos criterios determinados. A diferencia de las workflow rules, que sólo permiten un criterio de comprobación, en los process builder se pueden establecer varias condiciones en serie. (Declarativo)
- Flows: son flujos en Salesforce, permiten un nivel más alto de personalización que los process builder y puede ser invocados desde estos. No necesitan tener conocimientos de programación ya que pueden configurarse mediante bloques en una aplicación específicamente diseñada para ello. (Declarativo)
- Triggers: también llamados desencadenadores, son códigos de programación basados en lenguaje Apex[16] . que se invocan cuando se realiza una operación CRUD sobre un registro. Sólo pueden hacer referencia a un objeto a la vez y se recomienda únicamente tener un trigger por objeto [20]. (Programático)

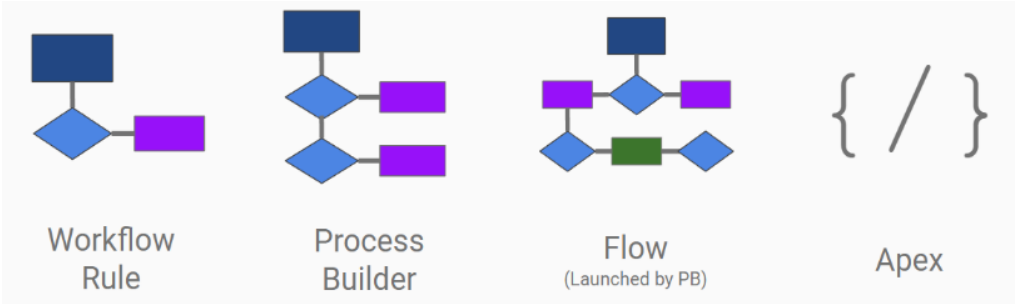

Figura 4.13 – Flujos y procesos en Salesforce (Fuente: [35])

[16] Apex: lenguaje orientado a objetos, nativo de Salesforce. Presenta una sintaxis similar a la de Java pero con particularidades que permiten la comunicación con la base de datos de Salesforce.

Interfaz de usuario

- Layouts y Lightning Pages: Son dos herramientas que permiten distribuir la información de los registros para visualizarlos en la pantalla, su configuración es sencilla y no se necesita tener conocimientos sobre programación. (Declarativo)
- Componentes Lightning: Componentes que permiten mostrar pantallas personalizadas accediendo a la base de datos de Salesforce. Necesitan conocimientos en programación ya que están basados en los lenguajes de programación web HTML, Javascrip y css pero con algunas modificaciones. Para la comunicación con la base de datos de Salesforce se necesita tener un controlador Apex. (Programático)

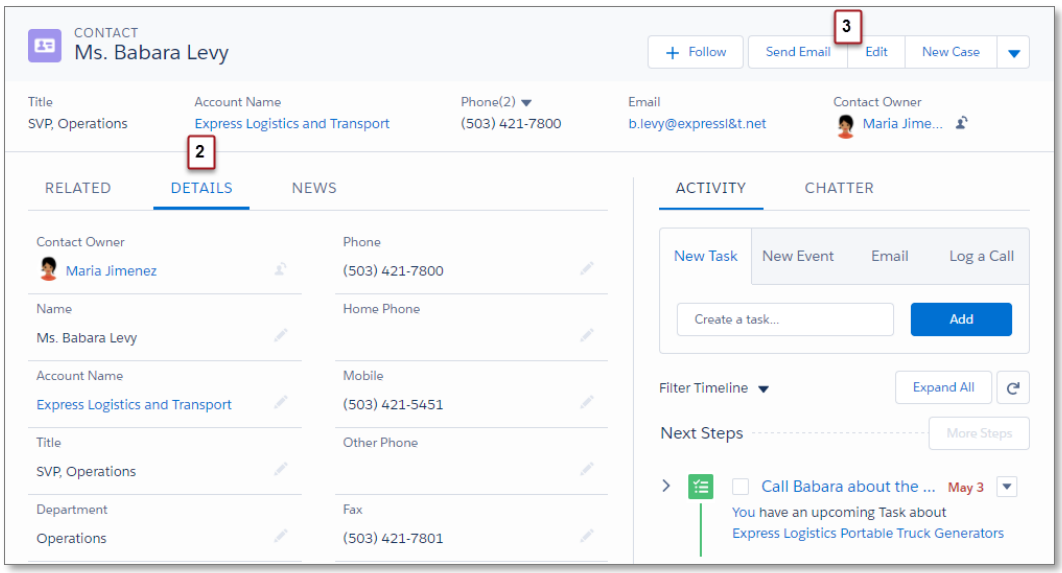

Figura 4.14 – Ejemplo intefaz de usuario en Salesforce (Fuente: [9])

4.3.4 Gestión de entornos

Uno de los conceptos fundamentales a la hora de comenzar cualquier desarrollo en Salesforce es la gestión de entornos. Un entorno en Salesforce es una plataforma en la que se pueden implementar una o varias aplicaciones y a la que se puede acceder a través de un usuario específico.

Existen principalmente dos tipologías, los entornos productivos, que son los entornos a los que accede el usuario final para hacer uso de las funcionalidades implementadas en la aplicación, y las "Sandboxes" que son copias del entorno de producción que sirven para las tareas de desarrollo, mantenimiento o pruebas. A un entorno productivo se accede mediante la url <https://login.salesforce.com> en la que se introduce un usuario y contraseña, y a una sandbox se accede mediante la url <https://test.salesforce.com>, donde el nombre de usuario generalmente es el mismo que el de producción añadiendo un punto y el nombre de la sandbox.

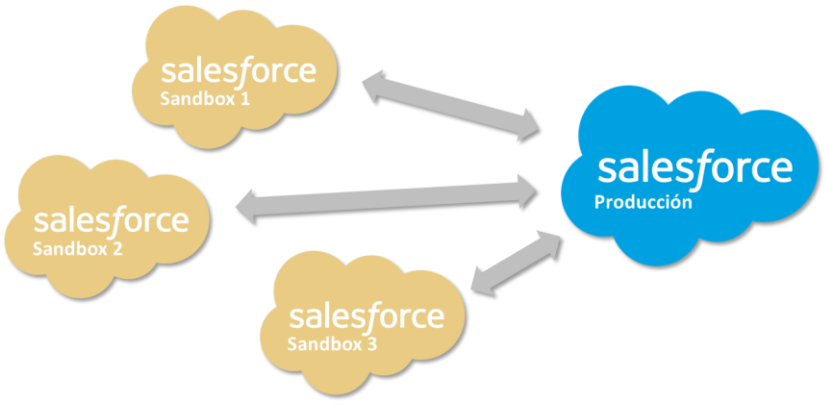

Figura 4.15 – Sandbox en Salesforce (Fuente: elaboración propia)

Dependiendo de las capacidades de la sandbox y si se ha copiado únicamente los metadatos o los datos existentes en el entorno de producción se pueden diferenciar diferentes tipos de sandboxes. Además, para cada una de ellas existirá un periodo mínimo que se debe producir entre un refresco[17] y el siguiente.

TIPO DE SANDBOX	PERIODO DE ACTUALIZACIÓN	LÍMITE DE ALMACENAMIENTO	¿QUÉ SE HA COPIADO?
Developer	1 día	Almacenamiento de datos: 200 MB Almacenamiento de archivos: 200 MB	Solo metadatos
Developer Pro	1 día	Almacenamiento de datos: 1 GB Almacenamiento de archivos: 1 GB	Solo metadatos
Partial Copy	5 días	Almacenamiento de datos: 5 GB Almacenamiento de archivos: 5 GB	Metadatos y una muestra de los datos
Full Copy	29 días	Almacenamiento de datos y archivos: Igual que el entorno de producción	Metadatos y todos los datos

Tabla 4.6 – Tipos de sandboxes (fuente: [36])

Según la licencia de Salesforce contratada se tendrán por defecto los siguientes entornos disponibles, que podrán ampliarse en caso de que sea necesario.

TIPO DE SANDBOX	PROFESSIONAL EDITION	ENTERPRISE EDITION	UNLIMITED EDITION
Developer	10	25	100
Developer Pro			5
Partial Copy		1	1
Full Copy			1

Tabla 4.7 – Sandboxes por licencia (fuente: [36])

Una vez que se conocen los tipos de entornos que existen, es necesario conocer cuando es más recomendable emplear uno u otro. A continuación, se muestra una tabla de los entornos más recomendados en función en función de la tarea que se quiera realizar, según la guía de Salesforce de buenas prácticas en la gestión de entornos [37].

Actividad	**Developer**	**Developer Pro**	**Partial Data**	**Full Copy**
Desarrollos	✔	✔	✔	✘
Pruebas calidad QA (usuario, funcionales...)	✔	✔	✔	✘
Pruebas integradas	✘	✘	✔	✔
Pruebas de aceptación de usuario UAT	✘	✘	✔	✔
Mantenimiento y soporte	✘	✘	✘	✔
Staging (Pre-producción)	✘	✘	✘	✔

Tabla 4.8 – Funcionalidad según tipología de Sandbox (fuente: [37])

[17] Refresco: derivado del término "refresh" en inglés, se llama refresco a la actualización de un entorno de prueba o de desarrollo con los datos de Producción

Por lo tanto, antes de comenzar un proyecto es necesario planificar una buena estrategia de entornos, teniendo en cuenta que actividades se van a realizar, posibles impactos cruzados con otras líneas, fechas de entrega, etc. En el siguiente esquema se puede ver un plan de entornos para un proyecto con varias líneas de desarrollo en paralelo, donde existe un entorno para integrar todos los desarrollos y entornos propios para realizar las pruebas y el mantenimiento.

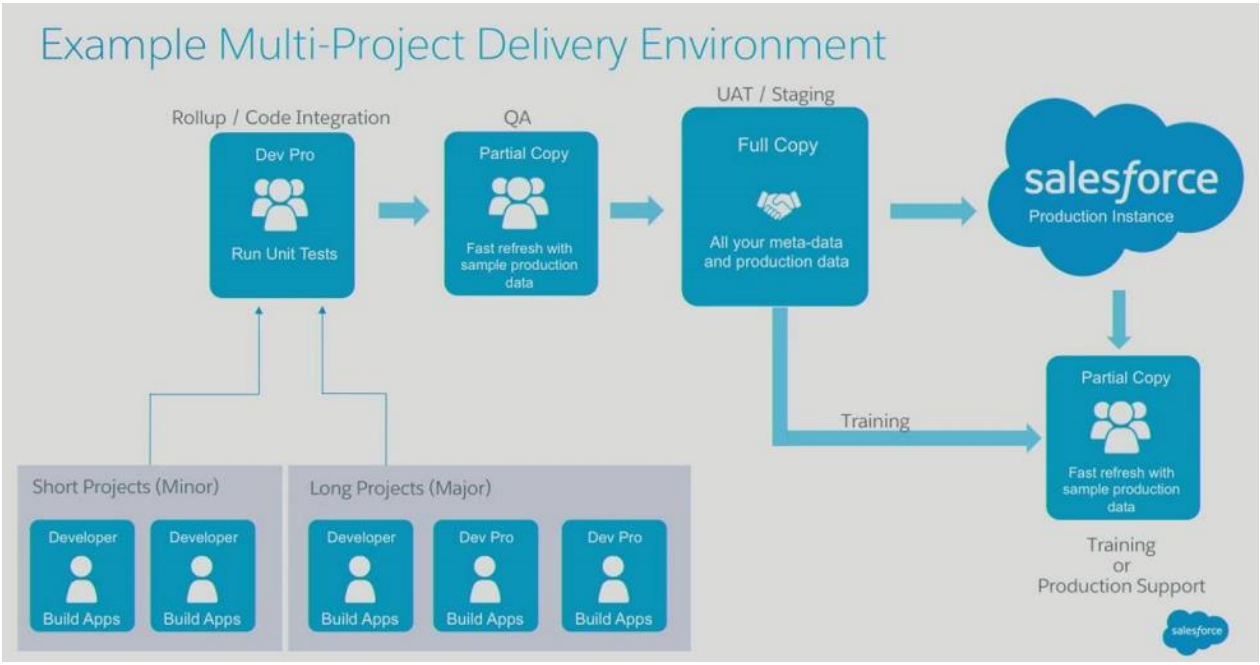

Figura 4.16 – Ejemplo estrategia de entornos (fuente: [36])

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

REFERENCIAS

[1] Eurostat, «Enterprises using software solutions, like CRM to analyse information about clients for marketing purposes,» 04 marzo 2020. [En línea]. Available: https://ec.europa.eu/eurostat/databrowser/view/tin00116/default/table?lang=en.

[2] «Worldwide Semiannual Software Tracker,» *IDC Trackers,* abril 2020.

[3] H. Takeuchi y I. Nonaka, «New New Product Development Game,» *Harvard Business Review,* p. 10, 01 01 1986.

[4] K. Beck, Extreme Programming Explained: Embrace Change, 75 Arlington Street, Suite 300 Boston, MAUnited States: Addison-Wesley Longman Publishing Co., Inc., 1999, p. 190.

[5] P. D., «History: Some pictures and PDFs of the Agile Manifesto meeting on 2001,» 27 03 2018. [En línea]. Available: https://siamchamnankit.co.th/history-some-pictures-and-pdfs-of-the-agile-manifesto-meeting-on-2001-a33c40bcc2b. [Último acceso: 19 04 2020].

[6] «Manifiesto por el Desarrollo Ágil de Software (Español),» 2001. [En línea]. Available: http://agilemanifesto.org/iso/es/manifesto.html.

[7] «Scrum México - Scrum y Agile: Conoce que es Scrum y que es Agile,» [En línea]. Available: https://www.scrum.mx/informate/scrum-y-agile.

[8] «Netx Idea Lab,» [En línea]. Available: https://nextidealab.com.pe/.

[9] Salesforce, «Trailhead | The fun way to learn,» [En línea]. Available: https://trailhead.salesforce.com/en/home. [Último acceso: 18 04 2020].

[10] J. N. V. A. Ferrari Nahuel, «Emaze,» [En línea]. Available: https://www.emaze.com/@AWIWFZZT.

[11] «Scrum manager - Invest,» [En línea]. Available: https://www.scrummanager.net/bok/index.php?title=INVEST.

[12] J. Sutherland, «ScrumInc - Story Points: Why are they better than hours?,» 16 05 2013. [En línea]. Available: https://www.scruminc.com/story-points-why-are-they-better-than/.

[13] X. Q. Allue, «Proyectos ágiles .org - Introducción a la estimación y planificación ágil,» 8 Junio 2009. [En línea]. Available: https://proyectosagiles.org/2009/06/08/introduccion-estimacion-planificacion-agil/.

[14] «Wikimedia - Ejemplo digarama Burn Down Chart,» [En línea]. Available: https://upload.wikimedia.org/wikipedia/commons/2/23/EjemploDeDiagramaBurnDown.svg.

[15] K. Schwaber y J. Sutherland, La Guía de Scrum. La Guía Definitiva de Scrum: Las Reglas del Juego, noviembre 2017.

[16] J. Sutherland y K. Schwaber, «Scrum Guides,» [En línea]. Available: https://scrumguides.org/index.html.

[17] «Trello,» [En línea]. Available: https://trello.com.

[18] A. Menzinsky, «Blog de un apóstol de Scrum y Kanban,» 19 septiembre 2016. [En línea]. Available: https://scrum.menzinsky.com/2016/09/como-leer-un-cfd-un-diagrama-de-flujo.html.

[19] «Science Encyclopedia: Extreme Programming,» [En línea]. Available: https://science.jrank.org/programming/Extreme_Programming.html.

[20] Salesforce, «Configuración y mantenimiento de su organización de Salesforce,» [En línea]. Available: https://help.salesforce.com/articleView?id=setup_overview.htm&type=5. [Último acceso: 17 04 2020].

[21] «SugarCRM pricing,» [En línea]. Available: https://www.sugarcrm.com/es/pricing/. [Último acceso: agosto 2020].

[22] «Zoho Pricing,» [En línea]. Available: https://www.zoho.com/es-xl/crm/zohocrm-pricing.html.

[23] «Qué es Salesforce,» [En línea]. Available: https://www.salesforce.com/es/products/what-is-salesforce/.

[24] «Salesforce - Biografía Marc Benioff,» [En línea]. Available: https://www.salesforce.com/company/leadership/bios/bio-benioff/.

[25] «Salesforce Blog ES,» 20 agosto 2019. [En línea]. Available: https://www.salesforce.com/es/blog/2019/08/20-Aniversario.html.

[26] M. Benioff, «Marc Benioff: How to Turn a Simple Idea into a High-Growth Company,» 2013 marzo 07. [En línea]. Available: http://answers.salesforce.com/blog/2013/03/how-to-turn-a-simple-idea-into-a-high-growth-company.html.

[27] «Brief history of Salesforce,» [En línea]. Available: https://www.salesforceben.com/brief-history-salesforce-com/.

[28] «Salesforce History,» septiembre 2020. [En línea]. Available: https://www.salesforce.com/news/stories/the-history-of-salesforce/#:~:text=and%20business%20world.%E2%80%9D-,2003,event%2C%20which%20has%2052%20presentations..

[29] «Salesforce AppExange,» [En línea]. Available: https://appexchange.salesforce.com/.

[30] «Salesforce CEO: Why I gave Apple the 'App Store' trademark in exchange for great advice from Steve Jobs,» [En línea]. Available: https://9to5mac.com/2011/08/26/salesforce-boss-tells-a-story-of-how-he-trademarked-and-urld-the-app-store-term-and-why-he-gave-it-to-steve-jobs/.

[31] «Different clouds in salesforce,» 07 06 2017. [En línea]. Available: https://sfdcbrothers.wordpress.com/2017/06/07/remote-objects-in-salesforce/.

[32] «Salesforce Pricing Sales Cloud,» [En línea]. Available: https://www.salesforce.com/es/editions-pricing/sales-cloud/. [Último acceso: julio 2020].

[33] «Salesforce Pricing Service Cloud,» [En línea]. Available: https://www.salesforce.com/es/editions-pricing/service-cloud/. [Último acceso: julio 2020].

[34] «Comprender el motivo de la adopción de la agilidad en Salesforce,» [En línea]. Available:

https://trailhead.salesforce.com/es-MX/content/learn/modules/salesforce-agile-basics/understand-why-salesforce-adopted-agile?trail_id=learn-salesforce-agile-practices.

[35] «Workflow, Process Builder, Flow, or Apex?,» 22 enero 2018. [En línea]. Available: https://www.sfdc99.com/2018/01/22/workflow-process-builder-flow-apex/.

[36] «Licencias y almacenamiento por tipo de entorno sandbox,» [En línea]. Available: https://help.salesforce.com/articleView?id=data_sandbox_environments.htm&type=5.

[37] «Sandbox Management Best Practices,» [En línea]. Available: https://help.salesforce.com/articleView?id=000338970&type=1&mode=1.

[38] «TIC Portal, Gestión de Proyectos,» 11 09 2008. [En línea]. Available: https://www.ticportal.es/glosario-tic/gestion-proyectos.

[39] Salesforce, «Certification in Salesforce,» [En línea]. Available: https://trailhead.salesforce.com/credentials/administratoroverview. [Último acceso: 18 04 2020].

[40] Ceolevel, «¿Conoces las diferencias entre Gestión Tradicional y Agile?,» 18 03 2019. [En línea]. Available: http://www.ceolevel.com/gestion-de-proyectos-agile. [Último acceso: 18 04 2020].

[41] D. Buonamico, «Caminoagil,» [En línea]. Available: http://www.caminoagil.com/.

[42] J. Roche, «Historia del movimiento Agile, Hacia la Agilidad y más allá,» [En línea]. Available: https://www2.deloitte.com/es/es/pages/technology/articles/historia-movimiento-agile.html. [Último acceso: 19 04 2020].

[43] J. Sutherland y K. Schwaber, «Scrum Development Process,» 1995.

[44] D. J. Anderson y A. Carmichael, Essential Kanban Condensed, Blue Hole Press, 2015.

[45] «IEBS, Herramientas para la gestión ágil de proyectos,» [En línea]. Available: https://www.iebschool.com/blog/herramientas-gestion-agil-proyectos-agile-agile-scrum/.

[46] D. ZS, «Metodologias agiles: Extreme Programming,» 16 octubre 2012. [En línea]. Available: http://danielzs75.blogspot.com/2012/10/metodologias-agiles-extreme-programming.html.

[47] M. Abad, «Blog Teaamleader: CRM fundamental,» 24 julio 2019. [En línea]. Available: https://blog.teamleader.es/crm-fundamental.

[48] J. Navalón, «La historia de Salesforce, de un garaje al rascacielos más alto de San Francisco,» 2019 agosto 02. [En línea]. Available: https://www.showerthinking.es/blog/la-increible-historia-salesforce/.

ÍNDICE DE CONCEPTOS

GLOSARIO

Agile: Término para agrupar las metodologías y la filosofía de gestión ágil 15

Apex: Lenguaje de programación para la comunicación con la base de datos de Salesforce 54

CRM: Customer Relationship Management 41

Extreme Programming: metodología ágil basada en la adaptabilidad 39

Framework: Marco de trabajo. Conjunto de prácticas enfocadas a resolver una problemática concreta 31

Kanban: Metodología ágil basada en el uso de tarjetas para organizar las tareas de forma visual 36

PO: Product Owner o Propietario del Proyecto, es el encargado de tomar las decisiones relativas a éste 34

Product Backlog: Catálogo de funcionalidades de un proyecto 23

SaaS: Software as a Service, Software como servicio 43

Scrum: Marco de trabajo Agile basa en iteraciones por sprints 32

Sprint: Ciclos de unas cuatro semanas utilizados en Scrum 35

User Story: Historia de usuario, necesidad recogida desde el punto de vista del usuario 23

Waterfall: Metodología de gestión de proyectos secuencial 18