

## 1.2. Planificación en SLDC

### 1.2.1 ¿Cómo se definen los objetivos y el alcance en SLDC?

En el contexto del **SDLC** (Software Development Life-Cycle) la *planificación* no termina hasta que **objetivos** y **alcance** están escritos, validados y firmados. A continuación tienes la **plantilla de trabajo** que usan muchas organizaciones (adaptable a cascada, ágil o mixto):

Objetivos (qué se quiere conseguir)

---

- Deben ser **SMART** (Specific, Measurable, Achievable, Relevant, Time-bound).
- Se escriben **siempre** en términos de impacto en el negocio, no de tecnología.
- Cada objetivo lleva **un indicador** y **un valor meta** con fecha.

Ejemplos corregidos:

- 1.«Reducir el tiempo promedio de atención al cliente de 12 min a  $\leq 5$  min en 6 meses tras el despliegue.»
  - 2.«Disminuir el desfase de inventario del 8 % actual a  $\leq 2$  % anual.»
  - 3.«Cumplir la normativa ISO-27001 antes del 30/09/2025.»
- 

2.Alcance (qué está DENTRO: In-Scope y qué está FUERA: Out-Scope)

---

Se documenta en tres bloques:

Sección	Qué poner	Ejemplo
<b>In-Scope</b>	Funcionalidades que el equipo <b>sí</b> construirá, con nivel de detalle suficiente para cualquier stakeholder reconozca.	el«CRUD( <b>C</b> –Create, <b>R</b> – Read, <b>U</b> – Update, <b>D</b> – Delete-Borrar/queEliminar) de productos, alertas de lasstock, reportes mensuales PDF/Excel, roles Admin/Vendedor, respaldo diario automático.»
<b>Out-of-Scope</b>	Funcionalidades <b>explícitamente excluidas</b> para evitar creep.	«Facturación electrónica, app móvil nativa, integración con SAP, gestión de nómina.»
<b>Criterios de aceptación</b>	Condiciones medibles que, cumplirse, dan por válida entrega.	al«Tiempo de respuesta $\leq 1$ s, ladisponibilidad 99 % en horario comercial, capacitación de 4 h al personal terminada.»

### 3. Plantilla rápida de 1 página

Markdown

Proyecto: \_\_\_\_\_

Versión: \_\_\_\_\_

#### 1. Objetivo general (SMART)

«\_\_\_\_\_»

#### 2. Objetivos específicos (lista)

- OE1: «Descripción...» (indicador: \_\_\_\_, meta: \_\_\_\_, fecha: \_\_\_\_)
- OE2: ...

#### 3. Alcance

##### 3.1 In-Scope

- ☒ ...
- ☒ ...

##### 3.2 Out-of-Scope

- ☒ ...
- ☒ ...

##### 3.3 Criterios de aceptación

- CA1: «...»
- CA2: ...

#### 4. Restricciones

- Presupuesto: \_\_\_\_\_
- Duración: \_\_\_\_\_
- Normativas: \_\_\_\_\_

### 4. Validación

Revisión conjunta con **sponsor (patrocinador), usuarios clave y equipo**.

- Firma digital o física → baseline del proyecto (punto de referencia).
- Cualquier cambio posterior se gestiona mediante **change-control** o **backlog ágil** con impacto en cronograma y presupuesto.

Con estos pasos objetivos y alcance quedan **definidos, medibles y blindados contra el scope-creep**.

### 1.2.2 ¿Qué es CRUD?

CRUD es el acrónimo de las **cuatro operaciones básicas** que se pueden hacer sobre cualquier conjunto de datos persistentes:

1. **C** – Create (Crear / Insertar)
2. **R** – Read (Leer / Consultar)
3. **U** – Update (Actualizar / Modificar)

#### 4. D – Delete (Borrar / Eliminar)

En la práctica significa: “el usuario puede dar de alta nuevos registros, consultarlos, modificarlos y borrarlos”.

Se aplica a: (a) bases de datos,

(b) APIs REST, (Una API REST (Interfaz de Programación de Aplicaciones de Transferencia de Estado Representacional) es un estilo arquitectónico para diseñar servicios web que permite a diferentes aplicaciones comunicarse entre sí usando protocolos estándar, principalmente HTTP, para intercambiar datos de forma sencilla, escalable y sin estado, mediante peticiones y respuestas estructuradas (usualmente en formatos como JSON o XML).

(c) Interfaces de usuario,

(d) Repositorios Git,

(e) Servicios en la nube, etc.

#### 1.2.3 ¿Qué es evitar el creep?

Evitar el “**creep**” (en inglés *scope creep* o *requirement creep*- funcionalidades no aprobadas o necesidades) significa **evitar que funcionalidades, tareas o requisitos NO aprobados se cuelen progresivamente en el proyecto**, haciéndolo más grande, costoso y largo sin control.

Cómo se previene:

1. **Alcance escrito y firmado** (baseline).
2. **Proceso formal de cambio**: toda modificación se solicita, evalúa impacto (tiempo, dinero, riesgo) y se aprueba/rechaza.
3. **Gestión activa del backlog** o **change control board** (Para modificación en los proyectos).
4. **Comunicación clara** al cliente: “sí se puede, pero esto implica X semanas y Y coste”.
5. **Hitos de entrega cortos** (puntos de control cortos clave - iterativos) que obligan a priorizar.

En resumen: ***cubrirse las espaldas para que el proyecto no “se salga por las ramas”***.

### 1.2.4 ¿Qué son los Hitos de entrega?

En la gestión de proyectos (y dentro del SDLC) un **hito de entrega** (*delivery milestone*) es un **evento temporal clave** que marca el **cumplimiento de un producto o resultado parcial** verificable y aprobado por los stakeholders (Personas con interés en el proyecto).

Características:

- **Fecha concreta** (p. ej. 15-abr-2025).
- **Entregable tangible**: documento, prototipo, módulo funcionando, versión alfa, etc.
- **Criterios de aceptación** definidos (¿qué debe contener y cómo se prueba?).
- **Aprobación formal**; una vez alcanzado, el proyecto avanza al siguiente paquete de trabajo.

Ejemplos típicos:

1. «Especificación de requisitos firmada» – **M0**
2. «Diseño de arquitectura aprobado» – **M1**
3. «Demo del MVP (producto mínimamente viable) con **3 flujos críticos** funcionando» – **M2**
4. «Entrega de versión 1.0 en producción» – **M3**

Sirven para:

- Medir progreso real sin confiar solo en % completado.
- Detectar desviaciones temprano.
- Motivar al equipo con metas cortas y visibles.

### ¿Qué es MVP?

MVP significa **Minimum Viable Product** (Producto Mínimamente Viable). Es la **versión más reducida** de un sistema que **entrega valor real al usuario** y **permite obtener feedback** con el mínimo esfuerzo de desarrollo.

Características:

- **Suficiente** para que los usuarios early-adopters (primeros usuarios) la usen y paguen.
- **Mínima** en funciones: solo lo imprescindible para validar la idea.
- **Viable**: funciona sin errores críticos y cumple su propósito central.

Ejemplo:

Para un sistema de inventario, el MVP podría ser solo «alta de productos + registro de entradas/salidas + reporte de stock actual».

Sin facturación, sin alertas avanzadas, sin integraciones.

Objetivo: **aprender rápido** si vale la pena seguir invirtiendo tiempo y dinero.