

Explicación de jerárquica del marco de trabajo **Scrum**, relacionando **proyectos**, **épicas**, **historias de usuario**, **tareas**, **actividades**, **sprints** y más, aplicable a equipos de desarrollo de sistemas:

---

### 1. Proyecto (Project)

- Es el **objetivo general** que se desea lograr.
  - Ejemplo: “*Desarrollar el Sistema de Tutorías Académicas de la UQROO*”.
  - Un proyecto se divide en **épicas** y se desarrolla a través de múltiples **sprints**.
- 

### 2. Épicas (Epics)

- Son **grandes bloques de funcionalidad** del proyecto.
  - Tienen un alcance amplio y se dividen en historias de usuario.
  - Ejemplo de épica: “*Gestión de tutorados*”, “*Reportes institucionales*”, “*Notificaciones al correo institucional*”.
- 

### 3. Historias de Usuario (User Stories)

- Describen una necesidad desde la **perspectiva del usuario final**.
- Tienen la estructura:

*Como [rol], quiero [funcionalidad] para [beneficio]*

- Ejemplo:

*Como tutor, quiero ver la lista de mis tutorados para poder darles seguimiento.*

- Cada historia puede tener un tamaño estimado en puntos (Story Points) y un criterio de aceptación.
- 

### 4. Tareas (Tasks)

- Son **subdivisiones técnicas** de una historia de usuario.
- Las realiza el equipo de desarrollo.

- Ejemplo para la historia anterior:
    - Crear tabla de tutorados en la base de datos.
    - Diseñar el componente visual en Angular.
    - Implementar endpoint en Spring Boot.
    - Realizar pruebas de integración.
- 

## 5. Actividades (Activities)

- Son las **acciones concretas** que se ejecutan día a día para completar una tarea.
  - Ejemplo:
    - Escribir una consulta SQL.
    - Hacer commit de una clase Java.
    - Escribir una prueba unitaria.
- 

## 6. Sprint

- Es un **ciclo de trabajo** fijo (usualmente 1 a 4 semanas).
  - En cada sprint se selecciona un conjunto de historias que el equipo se compromete a entregar.
  - Cada sprint incluye:
    - **Sprint Planning** (planificación del sprint)
    - **Daily Scrum** (reunión diaria de sincronización)
    - **Sprint Review** (demostración al final del sprint)
    - **Sprint Retrospective** (análisis de mejora continua)
- 

## 7. Incremento

- Al final del sprint se obtiene un **producto entregable** que cumple con la “Definición de Terminado”.

- Este incremento debe estar listo para ser potencialmente liberado al usuario final.
- 

## RELACIÓN JERÁRQUICA (Resumen gráfico)

PROYECTO

- └ Épica
  - └ Historia de Usuario
    - └ Tareas
      - └ Actividades

---

### Ejemplo aplicado:

**Proyecto:** Sistema de Tutorías

└ **Épica:** Módulo de Seguimiento

└ **Historia:** "Como tutor quiero ver la lista de tutorados"

└ **Tareas:** Crear API, mostrar tabla, conectar con BD

└ **Actividades:** Escribir código, hacer pruebas, revisar commits

→ **Sprint:** Se planifica completar esa historia durante el Sprint 2

---

# Artefactos

En Scrum, **los artefactos** son los documentos o representaciones tangibles que permiten **transparentar la información clave** del proyecto para facilitar la inspección y adaptación. Scrum define **tres artefactos principales**:

---

## 1. **Product Backlog (Lista del Producto)**

- Es una lista **ordenada** de todo lo que se necesita en el producto.
  - Evoluciona constantemente a medida que se descubren nuevas necesidades.
  - Lo gestiona el **Product Owner**.
  - Cada ítem se llama **PBI (Product Backlog Item)** y puede incluir historias de usuario, errores, mejoras, etc.
- 

## 2. **Sprint Backlog (Lista del Sprint)**

- Es el conjunto de elementos seleccionados del Product Backlog que se comprometen a completar durante un **Sprint**.
  - Incluye además un plan para entregarlos (normalmente en forma de tareas).
  - Lo gestiona el **Equipo de Desarrollo**.
  - Es dinámico: puede adaptarse durante el Sprint, pero solo el equipo puede hacerlo.
- 

## 3. **Incremento**

- Es la **suma de todos los PBIs completados** en el Sprint y los anteriores.
  - Debe estar en condiciones de ser entregado, es decir, cumplir con la **Definición de Terminado (Definition of Done)**.
  - Representa un paso hacia el objetivo del producto.
- 

**Extras recomendados:**

Scrum solo exige esos tres, pero también se suelen usar otros artefactos complementarios:

- **Burndown Chart** (Gráfica de trabajo pendiente)
- **Definition of Done** (acuerdo explícito de cuándo una tarea está realmente completada)
- **Product Goal** (objetivo general del producto)

# Ejemplos de Artefactos Scrum

## 1. Product Backlog

Título del Proyecto: Sistema de Tutorías Académicas

ID	Historia de Usuario	Prioridad	Estimación (pts)	Estado
001	Como estudiante, quiero registrarme	Alta	5	Pendiente
002	Como tutor, quiero ver a mis tutorados	Media	3	Pendiente
003	Como admin, quiero generar reportes	Alta	8	Pendiente

## 2. Sprint Backlog

Sprint: Sprint 1 | Duración: 2 semanas | Fecha inicio: 03-jun-2025 | Fecha fin: 17-jun-2025

Objetivo del Sprint: Tener lista la funcionalidad de registro y consulta básica de estudiantes.

Tarea	Historia Asociada	Responsable	Estimación (hrs)	Estado
Diseño del formulario de registro	001	Ana	4	En progreso
Programación del backend de registro	001	Luis	6	Pendiente
Validaciones del lado del cliente	001	Pedro	3	Pendiente

## 3. Incremento

Sprint: Sprint 1 | Fecha de revisión: 17-jun-2025

Definición de Terminado:

- ✓ Pruebas unitarias aprobadas
- ✓ Código subido a repositorio principal

✓ Revisión de QA

✓ Documentación mínima

Entregables del Sprint:

- Módulo de registro de estudiantes funcional.
- Validaciones básicas implementadas.
- Base de datos actualizada con nueva tabla `usuarios` .

Estado del Incremento: Listo para producción ✓

A

# ALNEXOS

Una **historia de usuario** y un **caso de uso** comparten similitudes, pero no son lo mismo. Se utilizan en contextos diferentes y tienen enfoques distintos:

---

## SIMILITUDES

Aspecto	Historia de Usuario	Caso de Uso
 Representan requisitos	Sí	Sí
 Describen interacciones	Sí (desde la perspectiva del usuario)	Sí (entre actor y sistema)
 Tienen un "actor"	Sí (rol del usuario)	Sí (actor definido explícitamente)
 Usan para desarrollo	Sí, especialmente en ágil (Scrum)	Sí, en metodologías tradicionales/UML

---

## DIFERENCIAS CLAVE

Característica	Historia de Usuario	Caso de Uso
 Formato	Texto breve y simple (1-3 líneas)	Documento estructurado o diagrama UML
 Enfoque	Necesidad y valor del usuario	Comportamiento detallado del sistema
 Nivel de detalle	Bajo (puede dividirse en tareas)	Alto (pasos, condiciones, excepciones)
 Estructura típica	"Como [rol], quiero [meta], para [beneficio]"	Actor, flujo principal, alternativo, errores
 Uso en metodologías	Ágil (Scrum, XP, Kanban)	Tradicionales (RUP, UML) o híbridas

---

## Ejemplo comparado

### **Historia de Usuario:**

**Como** tutor académico,  
**quiero** poder registrar observaciones de cada sesión,  
**para** hacer seguimiento al desempeño del estudiante.

### **Caso de Uso (resumen):**

**Nombre:** Registrar observaciones

**Actor:** Tutor académico

**Precondición:** El tutor ha iniciado sesión.

**Flujo principal:**

1. El tutor accede al perfil del estudiante.
2. Selecciona “Aregar observación”.
3. Ingresa el texto y guarda.
4. El sistema registra la fecha y hora.

**Postcondición:** Observación guardada en la base de datos.

---

### **¿Cuándo usar uno u otro?**

- Use **historias de usuario** si trabaja con **Scrum** o métodos ágiles → son breves, flexibles y centradas en el valor.
  - Use **casos de uso** si necesita documentar **flujos detallados**, escenarios alternativos, validaciones y condiciones → más útiles en análisis técnico o contratos formales.
-

# Plantilla de Historia de Usuario

---

*Formato basado en metodologías ágiles (Scrum).*

---

## **Historia de Usuario**

**\*\*ID:\*\* HU-001**

**\*\*Título:\*\*** Registro de observaciones por parte del tutor académico

**\*\*Como\*\*** tutor académico,

**\*\*quiero\*\*** registrar observaciones de cada sesión de tutoría,

**\*\*para\*\*** dar seguimiento adecuado al desempeño del estudiante.

## **Criterios de Aceptación**

- El tutor debe poder acceder al perfil del estudiante.
- Debe existir un botón “Agregar observación”.
- El sistema debe registrar automáticamente la fecha y hora.
- Las observaciones deben poder visualizarse en sesiones futuras.

## **Notas Técnicas / Tareas**

- Crear la tabla de observaciones en la base de datos.
- Diseñar el formulario de observación en el frontend.
- Implementar la lógica de guardado en backend.
- Validar campos obligatorios.

# Requerimientos

En Scrum no existe un momento único ni formalmente llamado "levantamiento de requerimientos", como ocurre en metodologías tradicionales. Sin embargo, **sí se levantan requerimientos, pero de forma incremental, continua y colaborativa.**

## ✳️ ¿Cómo se abordan los requerimientos en Scrum?

### 1. Durante la creación del Product Backlog

- El **Product Owner** trabaja con los stakeholders para identificar necesidades iniciales del producto.
- Estas se traducen en **épicas o historias de usuario**.
- Este proceso **no es único ni cerrado**, sino **iterativo**.

### 2. En cada Sprint Planning

- Se detallan y seleccionan historias prioritarias para ser desarrolladas.
- El equipo de desarrollo puede hacer preguntas, discutir condiciones, y **afinar requerimientos antes de comprometerse**.

### 3. A través del refinamiento del Product Backlog (Backlog Grooming)

- Se hace regularmente (idealmente cada semana).
- Se descomponen épicas, se aclaran historias, se ajustan estimaciones, y se priorizan elementos.
- Involucra al **Product Owner** y al equipo de desarrollo.

### 4. Mediante conversaciones constantes con el Product Owner

- Las historias de usuario están pensadas como **invitaciones al diálogo**.
- El equipo interactúa con el PO para comprender el **valor** y el **contexto** del requerimiento.

---

## ⌚ Diferencias clave con el enfoque tradicional

### Enfoque Tradicional

Requerimientos completos al inicio

### Scrum (Ágil)

Requerimientos evolucionan con el tiempo

Enfoque Tradicional	Scrum (Ágil)
Documento formal de especificaciones	Historias de usuario + criterios de aceptación
Poco cambio permitido	Cambios son esperados y gestionados
Análisis separado del desarrollo	Análisis y desarrollo van de la mano

---

### Conclusión

En Scrum:

- **Sí se levantan requerimientos**, pero de forma **continua, colaborativa y ligera**.
- No hay una **fase única ni documento rígido**.
- El foco está en **entender el valor, priorizar lo más importante y adaptarse al cambio**.

# Roles fundamentales

En Scrum, existen **tres roles fundamentales**, cada uno con responsabilidades bien definidas. No se permiten otros roles dentro del marco formal. Aquí te explico cada uno:

---

## 1. Product Owner (PO) – *Dueño del producto*

**Rol:**

- Representa los intereses del **cliente o negocio**.
- Es responsable de **maximizar el valor del producto** desarrollado por el equipo.
- Administra el **Product Backlog**.

**Responsabilidades:**

- Definir y priorizar las **historias de usuario**.
- Asegurar que el backlog esté **visible, claro y entendido**.
- Tomar decisiones sobre el producto y **aceptar o rechazar entregables**.

**Personalidad ideal:**

- Orientado al negocio.
  - Comunicador eficaz.
  - Decisivo, con visión estratégica.
- 

## 2. Scrum Master – *Facilitador del proceso*

**Rol:**

- Es un **líder servicial** que guía al equipo en la correcta aplicación de Scrum.
- Ayuda a eliminar obstáculos y promueve la mejora continua.
- Protege al equipo de interferencias externas.

**Responsabilidades:**

- Facilitar las ceremonias Scrum (Daily, Planning, Review, Retrospective).
- Capacitar y apoyar al Product Owner y equipo.
- Promover la **autoorganización y colaboración**.

#### **Personalidad ideal:**

- Empático, paciente y observador.
  - Habil para resolver conflictos.
  - Firme en procesos, flexible con personas.
- 



### **3. Equipo de Desarrollo (Developers) – *Equipo de entrega***

#### **Rol:**

- Son los responsables de **construir el incremento del producto**.
- Se autoorganizan para lograr el objetivo del sprint.

#### **Responsabilidades:**

- Participar en la planificación del sprint.
- Ejecutar tareas técnicas, pruebas, diseño, documentación, etc.
- Mantener la calidad y cumplir con la “Definición de Terminado”.

#### **Personalidad ideal:**

- Colaborativos, con mentalidad de equipo.
  - Proactivos y autoorganizados.
  - Orientados a resultados y mejora continua.
- 

## **Relación y Colaboración**

<b>Rol</b>	<b>Interacción principal</b>
Product Owner	Stakeholders / Equipo
Scrum Master	Todo el equipo Scrum
Developers	Entre sí y con el PO

En **Scrum** no existen fases formales secuenciales como en los modelos tradicionales (análisis → diseño → codificación → pruebas → despliegue). En lugar de fases, Scrum trabaja con **iteraciones llamadas Sprints**, y dentro de cada Sprint se pueden realizar **actividades de análisis, diseño, desarrollo, pruebas y despliegue**, pero de forma **concurrente, continua y colaborativa**.

---

### En Scrum, todo esto ocurre dentro de un Sprint:

Actividad tradicional	¿Existe en Scrum?	¿Cómo se aborda?
 Análisis	<input checked="" type="checkbox"/> Sí	Durante la creación/refinamiento del backlog y en Sprint Planning.
 Diseño	<input checked="" type="checkbox"/> Sí	Se hace en conjunto con el desarrollo, muchas veces justo a tiempo (just-in-time design).
 Codificación	<input checked="" type="checkbox"/> Sí	Parte central del trabajo del equipo de desarrollo en el Sprint.
 Pruebas	<input checked="" type="checkbox"/> Sí	Se hacen dentro del mismo Sprint. Se busca entregar software probado y funcional.
 Despliegue	<input checked="" type="checkbox"/> Opcionalmente	Puede hacerse si el incremento está listo; se promueve la entrega continua.

---

### Estructura del Sprint

Sprint (1–4 semanas)

- |— Refinamiento del backlog (previo o durante el sprint)
- |— Sprint Planning (inicio)
- |— Daily Scrums (diario)
- |— Desarrollo (incluye análisis, diseño, pruebas, documentación...)
- |— Sprint Review (presentación del producto)
- └ Sprint Retrospective (mejora del proceso)

---

### Beneficios de este enfoque iterativo

- **Adaptabilidad:** puedes incorporar cambios cada Sprint.
  - **Entrega temprana y continua de valor.**
  - **Análisis y pruebas integradas**, no separadas.
- 

### ¿Qué pasa con QA, testing y documentación?

- Las actividades de **QA y testing** se integran al Sprint.
  - El equipo es **responsable de la calidad**, no solo un tester.
  - La **documentación** se hace solo cuando es necesaria para entregar valor.
- 

Si deseas, puedo prepararte una tabla o cronograma tipo Gantt que muestre cómo se distribuyen estas actividades dentro de varios Sprints, o generar un documento Word con esta explicación. ¿Lo deseas?

# Actividades de Análisis, Pruebas y Despliegue en Scrum

En Scrum no existen fases tradicionales separadas (como análisis, diseño, pruebas o despliegue). En su lugar, todas estas actividades se integran dentro de iteraciones llamadas Sprints. Esto permite que el equipo desarrolle, pruebe y entregue valor continuamente durante todo el proyecto.

## Distribución de Actividades por Sprint

Actividad Tradicional	¿Existe en Scrum?	¿Cómo se aborda?
Análisis	Sí	Durante la creación y refinamiento del Product Backlog, y en Sprint Planning.
Diseño	Sí	Se realiza en el Sprint de forma conjunta con el desarrollo.
Codificación	Sí	Actividad principal del equipo de desarrollo en cada Sprint.
Pruebas	Sí	Incluidas dentro del Sprint; el software entregado debe estar probado.
Despliegue	Sí (opcional)	Si el incremento está listo, se puede desplegar; se promueve la entrega continua.
Documentación	Sí	Solo la necesaria para generar valor, se hace dentro del Sprint.

## Conclusión

Scrum no separa las actividades por fases. Todo el trabajo necesario para entregar un incremento funcional se realiza dentro del Sprint. Esto incluye análisis, diseño, codificación, pruebas, documentación e incluso despliegue, si se decide. Esta forma de trabajo iterativa permite al equipo adaptarse rápidamente a los cambios y entregar valor de forma continua.

Sí, en Scrum existen actividades antes y después de los Sprints, aunque no se llaman "fases" en el sentido tradicional. Estas actividades ayudan a **preparar, iniciar, revisar y mejorar** el desarrollo del producto de forma ágil y continua.

---

## ANTES DE LOS SPRINTS

### 1. Visión del producto

- Definición general del **objetivo o propósito del producto**.
- Elaborada por el **Product Owner** con stakeholders.
- Guía todo el desarrollo.

### 2. Roadmap del producto (opcional)

- Línea de tiempo de alto nivel con **épicas y versiones previstas**.

### 3. Creación del Product Backlog

- Lista inicial de **requerimientos priorizados** (épicas e historias).
- No tiene que estar completa al inicio, **evoluciona constantemente**.

### 4. Refinamiento del Backlog (Product Backlog Refinement)

- Actividad continua para:
  - Dividir épicas en historias más pequeñas.
  - Estimar esfuerzo (story points).
  - Aclarar criterios de aceptación.

---

## DURANTE CADA SPRINT

- Sprint Planning
- Daily Scrum
- Desarrollo, pruebas, documentación
- Sprint Review
- Sprint Retrospective

---

## DESPUÉS DE LOS SPRINTS

## 1. Sprint Review

- Presentación del incremento terminado.
- Recibe retroalimentación del **Product Owner** y **stakeholders**.
- Puede generar nuevos ítems para el backlog.

## 2. Sprint Retrospective

- Reunión del equipo Scrum para analizar:
  - ¿Qué fue bien?
  - ¿Qué puede mejorarse?
  - ¿Qué acciones se tomarán para el siguiente Sprint?

## 3. Revisión del Product Backlog

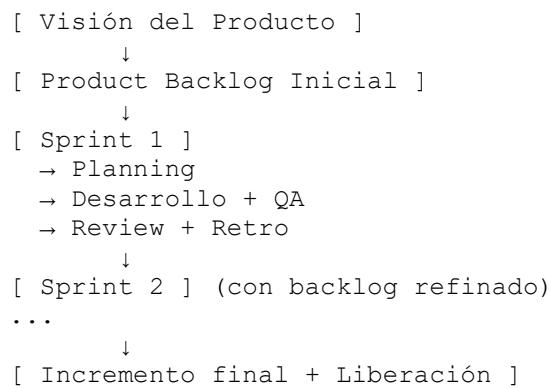
- Se ajusta según el feedback recibido.
- Se priorizan nuevos requerimientos o cambios.

## 4. Liberación (Release)

- Si el incremento cumple con la "Definition of Done", puede ser **desplegado a producción**.
- Scrum no obliga a liberar en cada Sprint, pero lo permite.



## VISIÓN GLOBAL DEL CICLO SCRUM



¿Desea que le prepare un diagrama o un documento Word con esta explicación estructurada?