Requerimientos Agiles

▼ Relación entre Proceso, Proyecto y Producto 🎯

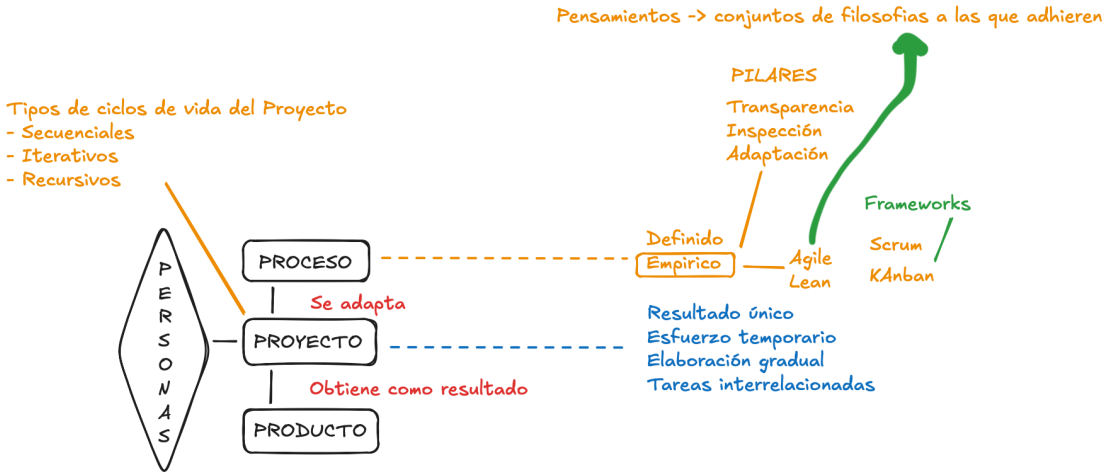
El **proceso** se adapta a un proyecto, y de ese proyecto surge como resultado un producto.

Un proceso no es más que un **conjunto de actividades estructuradas** cuyo objetivo es obtener un producto o un servicio. En este caso, hablamos de un software que nace a partir de ideas y necesidades. Para llevarlo a cabo, el proceso consume recursos y requiere personas que transformen las entradas en resultados concretos.

Ahora bien, esas actividades del proceso se instancian y se ejecutan dentro de un **proyecto**.

El proyecto es la **unidad de gestión** que organiza tanto el trabajo como los recursos humanos y materiales. Es el medio que permite cumplir objetivos concretos. Mientras que el proceso me dice *cómo* hacer las cosas, el proyecto es el lugar donde ese proceso cobra vida. Dicho de otra manera: **los procesos serían como las clases en programación, y los proyectos serían los objetos instanciados de esas clases**.

Por ejemplo, un proceso puede indicar que “el arquitecto definirá la arquitectura del software”. Eso es un rol genérico. En cambio, en el proyecto ese rol se asigna a una persona en particular, como decir: *Florencia Pavon va a definir la arquitectura del software*.



▼ Características de un Proyecto 📌

Un proyecto tiene varias características que lo distinguen:

- **El resultado siempre es único**: cada proyecto de software genera un producto distinto, nunca se producen en masa como en una línea de producción.
- **Es temporal**: tiene un inicio y un fin definidos.
- **Se elabora gradualmente**: se construye de a poco, a través de tareas interrelacionadas que van dando forma al producto final.

Además, todo proyecto debe estar **conducido por alguien**. En metodologías tradicionales es el **líder del proyecto**, mientras que bajo el paraguas de Agile aparece el **Scrum Master**, que se encarga de velar porque las cosas se hagan como corresponde.

▼ Procesos Definidos vs Empíricos ⚖️

Un **proceso definido** busca tener todo detallado y documentado para ser predecible. Existen ingenieros de procesos que lo mantienen actualizado y formalizado. La idea es que bajo las mismas entradas, se puedan obtener siempre las mismas salidas. Es más burocrático, estructurado y depende fuertemente de los lineamientos de la organización.

En cambio, un **proceso empírico** se basa en la experiencia que el equipo genera y realimenta en el día a día. Esa experiencia no es extrapolable a otros equipos o proyectos: cada grupo aprende haciendo. Se trabaja con

mayor flexibilidad, adaptando el proceso según lo que surja. Se dice que “el agua del río no pasa dos veces por el mismo lugar”: cada proyecto es único y el aprendizaje ocurre en el camino.

Los **pilares de los procesos empíricos** son tres:

1. **Transparencia**
2. **Inspección**
3. **Adaptación**

La transparencia implica sinceridad y comunicación constante: si un integrante del equipo se da cuenta de que no va a llegar a un plazo, debe levantar la mano lo antes posible para pedir ayuda. La inspección y adaptación se dan en ciclos cortos que permiten aprender, corregir y volver a intentar.

▼ Ciclo de Vida del Proyecto 🔄

En esta materia se habla sobre el **ciclo de vida del proyecto**, también conocido como **modelo de proceso**.

Existen distintos tipos de ciclos:

1. **Secuenciales → Cascada**
2. **Iterativos → Incremental / Proceso unificado de desarrollo**
3. **Recursivos → Espiral**

El proceso indica **qué tareas hacer**, y el ciclo de vida determina **en qué orden y en qué tiempo** deben realizarse. Por ejemplo, se puede elegir el proceso de desarrollo unificado con ciclo de vida en cascada. Sin embargo, el modelo en cascada tiene la desventaja de que no permite ajustar los requerimientos que van cambiando a lo largo del tiempo.

Por eso, si se trabaja en un **contexto empírico**, se necesita sí o sí un ciclo de vida **iterativo**. Los ciclos cortos permiten generar experiencia, inspeccionar y adaptarse constantemente. Agile y Lean, por ejemplo, requieren obligatoriamente un ciclo iterativo.

▼ Filosofía Ágil 🏃

El **Agilismo no es una metodología ni un proceso**, sino una forma de pensar. Surge como un movimiento de desarrolladores que se rebelaron contra la rigidez de los procesos definidos. El software es intangible, y ellos necesitaban formas de **evidenciar avances reales**.

De ahí nació el **Manifiesto Ágil**, que plantea 4 valores y 12 principios. Aunque parecen simples, deben cumplirse rigurosamente. La clave no está en la velocidad, sino en **entregar valor al cliente**. El software no es un fin en sí mismo, sino un medio para darle al cliente lo que realmente necesita.

En este contexto aparece el **Product Owner (PO)**: un rol de negocio, no técnico, que sabe priorizar qué es lo más importante para el cliente. Trabaja junto al equipo técnico y al analista, que lo ayudan a transformar esas ideas en requerimientos. El objetivo es identificar un **MVP (producto mínimo viable)** y avanzar con entregas frecuentes de valor.

▼ Requerimientos Ágiles 📋

En el enfoque ágil, los requerimientos se manejan de forma dinámica a través del **Product Backlog**.

- El PO es el dueño de este artefacto.
- Se organiza como una pila de prioridades.
- Es un artefacto vivo, que nunca representa el 100% del producto final.
- Los ítems pueden ser **casos de uso o user stories**.

Con el principio de *just in time*, el equipo trabaja en los requerimientos con el nivel justo de detalle necesario para la iteración en curso, evitando desperdiciar energía en definir cosas que podrían cambiar más adelante.

El **mejor medio de comunicación es cara a cara**: los analistas y el PO deben conversar frente al pizarrón, no limitarse a mails o documentos. Esto reduce malentendidos y fortalece la transparencia.

▼ User Stories 📝

Las **user stories** son una forma corta y simple de describir una necesidad del usuario. Tienen tres partes, conocidas como las **3 C**:

- **Card** (tarjeta): la descripción breve.
- **Conversation** (conversación): el diálogo con el equipo, la parte más importante.
- **Confirmation** (confirmación): los criterios de aceptación.

La sintaxis típica de una user story es:

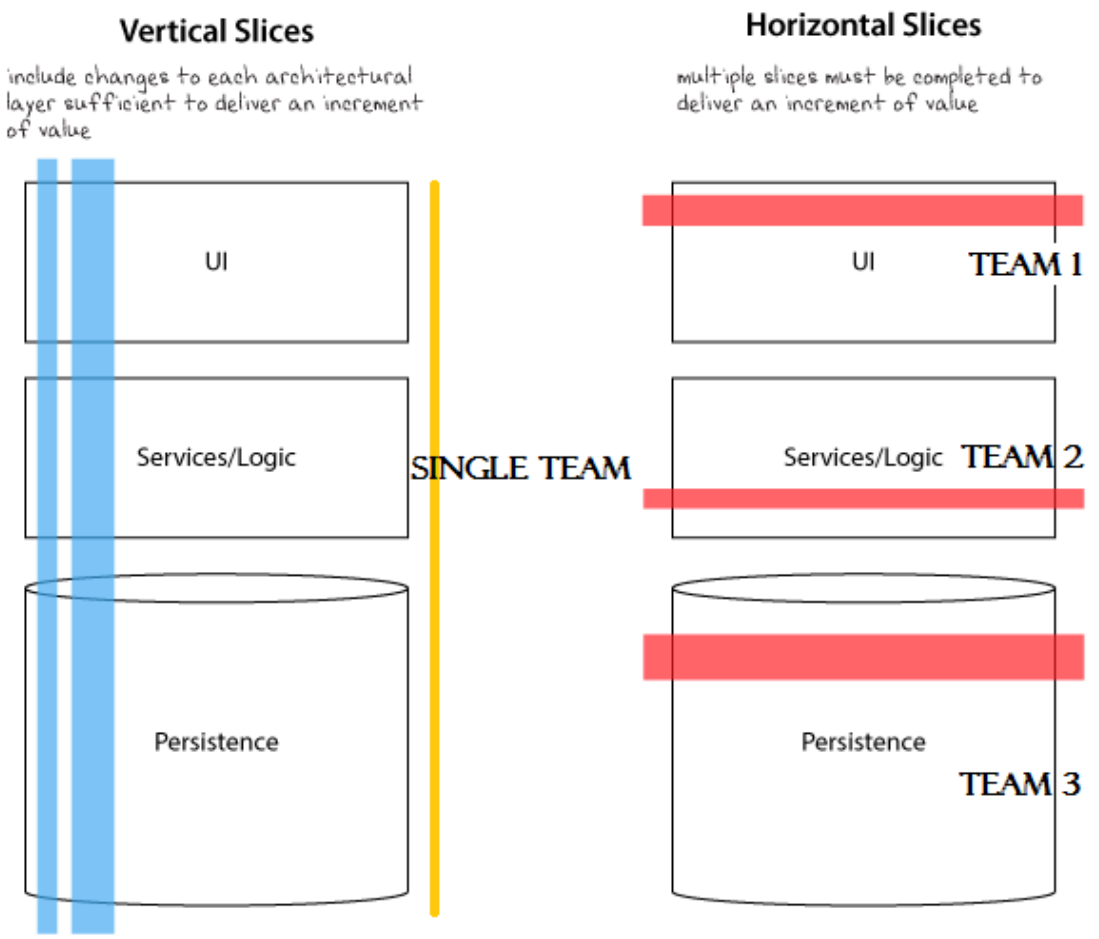
Como [rol]

Quiero [actividad]

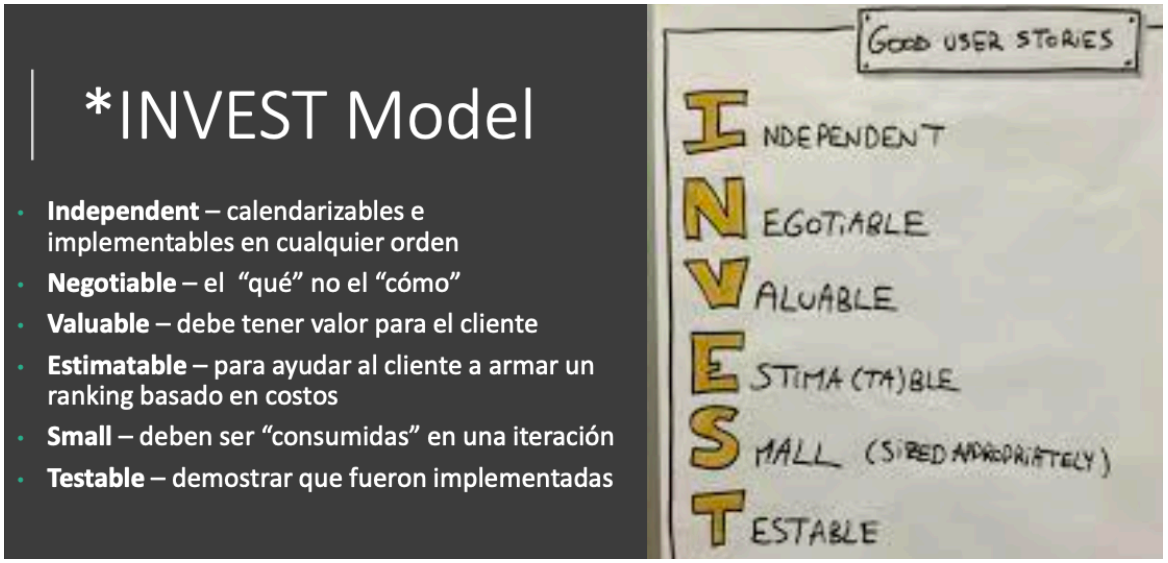
Para [objetivo/valor de negocio]

Por ejemplo: *Como conductor, quiero ver la ruta más rápida, para ahorrar tiempo en mi viaje*.

El valor de las user stories es que permiten pensar en **porciones verticales** de producto (un poco de front, un poco de back y un poco de base de datos), de modo que siempre se entregue software funcionando.



Los **criterios de aceptación** funcionan como base para las pruebas de usuario. Mientras más inmaduro es el equipo, más detallados deben ser estos criterios. En cualquier caso, siempre debe haber al menos uno.



Antes de ingresar una user story a la siguiente iteración, se utiliza el checklist **INVEST**:

- **I**ndependent
- **N**egotiable
- **V**aluable
- **E**stimable
- **S**mall
- **T**estable

Esto asegura que la user story esté realmente lista para ser trabajada.