

8. Historias de usuario

Es una técnica asociada a las metodologías ágiles, busca *describir una funcionalidad del sistema desde la perspectiva del usuario*. Se estructura con un:

- como ...
- quiero ...
- para ...

Hace foco en ¿qué valor agregado le vamos a dar al usuario? qué requiere el sistema para proveer ese valor?

Cómo escribir historias de usuario de calidad

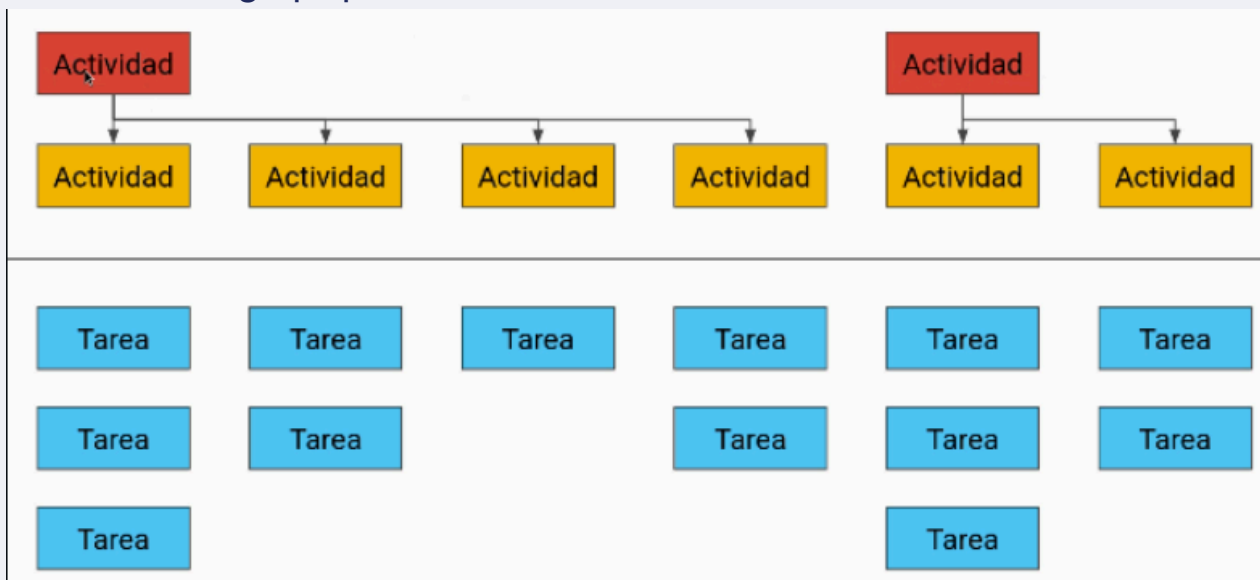
- Framework QUS (Quality User Story)
 - calidad sintáctica: fácil de entender, atómico, formato "como... quiero... para..."
 - calidad semántica: con conceptos de dominio bien conocidos y no ambiguos. Se enfoca en expresar qué problema (oportunidad) expresa un usuario, la solución propuesta es secundaria. No deben de haber conflictos entre historias.
 - calidad pragmática: no se repiten, son estimables en cuanto a esfuerzo que requerirá, son completas, son independientes, usan oraciones completas.
 - *se recomienda* que las tareas del backlog cumplan los *criterios INVEST*:
 - Independientes (no se repiten y se puede paralelizar el desarrollo de las tareas)
 - Negociables (se puede ajustar según necesidad)
 - Valiosas (aporta valor al user)
 - Estimable (se puede medir el esfuerzo requerido)
 - Small (es pequeña, atómica, fácil de implementar, se pueda desarrollar en una iteración)
 - Testeable (se puede verificar que está bien hecha)
 - *No se recomienda*:

- que las historias se centren en la solución
- que presupongan detalles de implementación o de cómo se solucionará (a veces el usuario sí requiere que ciertos detalles sean de una forma, esto se menciona en la historia)
- que si nos damos cuenta que hay otra oportunidad, se deje en la user story (como por ejemplo que un grupo quiera algo para hacer otra cosa manualmente, entonces la automatización de lo generado manualmente es otra historia de usuario)

⚡ cómo encontrar historias de usuario

1. establecer dominio del sistema (qué no se abarca y qué sí)
2. identifica actores involucrados en el sistema.
3. busca objetivos y actividades dentro del sistema (relaciona a los usuarios y su interacción con el sistema). Se usan herramientas como:
 - User story mapping
 - Impact mapping.

User story Mapping: ayuda a definir los productos a desarrollar. el diagrama resultante ==agrupa por funcionalidades. ==

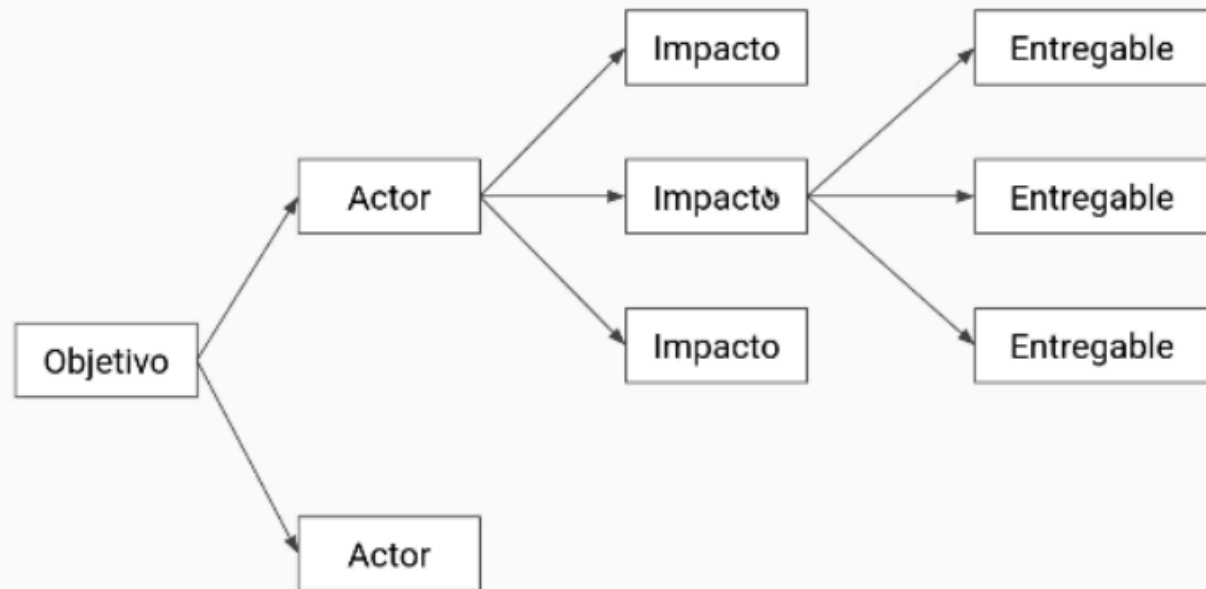


Donde:

- las actividades rojas (nivel superior) son funcionalidades principales del sistema
- las amarillas son los pasos dentro de cada funcionalidad
- las azules son las tareas técnicas específicas que pueden tener diferentes



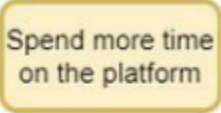
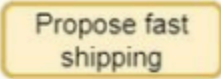
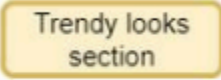
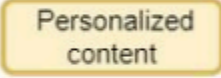

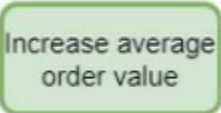
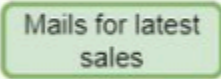
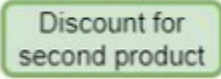
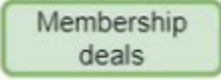
versionados (el que esté más abajo es la última versión).

Impact mapping: orientada a los **objetivos del negocio**, buscan que los entregables se alinee con los objetivos del negocio. TODO EL DESARROLLO debe tener un propósito claro y valor real.



1. definimos el objetivo a lograr (ejm: aumentar el uso de tal app)
2. definimos los actores involucrados con el objetivo
3. definimos los impactos que queremos tener sobre esos actores que se alineen al objetivo
4. definimos los entregables que creemos que podrían provocar el impacto esperado.

por ejemplo:

Goal	Actor	Impact	Deliverable
	 Ann		  
	 Dave		  

🔗 3 Cs (Card Conversation, Confirmation)

- **Card**: describe la intención del usuario en una tarjeta
- **Conversation**: los interesados se comunican para refinar las historias, para descubrir y documentar los requisitos.
- **Confirmation**: se define los criterios de aceptación

Criterios de aceptación

Son condiciones específicas que se deben de cumplir para que el trabajo se acepte y se considere hecho. Se expresan como **listas de condiciones y escenarios**. Se usa el formato:

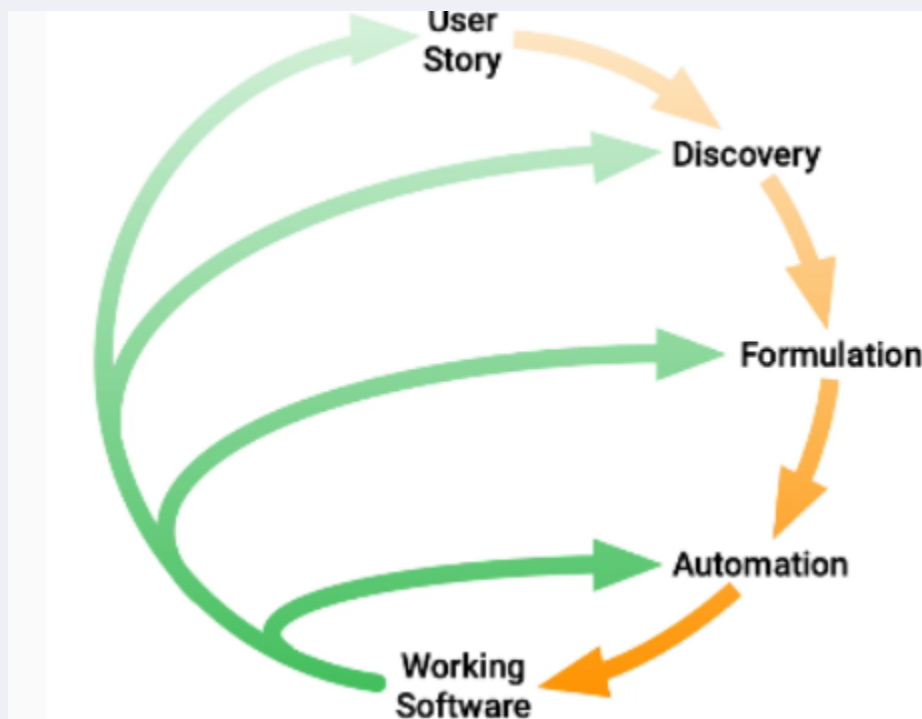
- Dado que **tal contexto**, cuando suceda **tal evento**, entonces **tal consecuencia**.

Deben tener claridad, ser concisos, verificables, independientes y orientados al problema

se pueden definir criterios de aceptación tanto para historias de usuario como para casos de uso.

Behaviour Driven Development

- El desarrollo se orienta al comportamiento.
- Es una extensión de la metodología TDD (Test-Driven-Development)
- define cómo debe de comportarse *el sistema* antes de implementarlo.
- Es un proceso iterativo que sigue el orden:
 1. Planteamiento de las user storys
 2. discovery
 3. formulation
 4. automation



🔥 Casos de uso

Describe una serie de *acciones realizadas por un sistema* que *generan* un *resultado observable de valor* para un *actor en particular*.

- Se compone de un **escenario principal** y un conjunto de **escenarios alternativos**. (definen flujos de trabajo del sistema)
- Suele estructurarse como un documento con **pasos detallados**, condiciones y excepciones
- describe cómo *interactúa el user con el sistema* y con otros sistemas para lograr un objetivo
- Suele ser formal y estructurado, es menos iterativo.

- Se representa con modelado tradicional (UML, waterfall, RUP)
- En cuanto los criterios de aceptación, se incluyen múltiples escenarios y validaciones del flujo del caso de uso

Por ejemplo:

Versiones más compactas:

Consultar Precio de un Producto

1. El cliente ingresa el código de producto
2. El sistema informa el precio del producto
 - 2.1. Si el código de producto no corresponde a un producto registrado, el sistema informa que desconoce el producto

Consultar Precio de un Producto

El cliente ingresa el código de producto. El sistema informa el precio del producto o que desconoce el producto

DIAGRAMA DE CASOS DE USO

