# Metodologías de Desarrollo Ágil

## Seminario 2: Historias de Usuario (HU)

Apuntes Exhaustivos de Teoría, Esquemas Visuales y Ejercicios de Examen

Grado en Ingeniería Informática · Especialidad Ingeniería del Software

Departamento de Lenguajes y Sistemas Informáticos · Universidad de Granada (UGR)

### Síntesis Integral de Contenidos Evaluables (99 Diapositivas Oficiales):

1. **Fundamentos y Comparativa:** Definición de Beck y Fowler, "Promesas de una conversación" y tabla completa *Casos de Uso (RUP) vs. Historias de Usuario (XP/Scrum)*.
2. **Roles, Autoría y Formato LEGO:** Quién escribe las HU (XP, Scrum, Customer Team), modelado de roles y sintaxis canónica `Como [rol], Quiero [objetivo], Para [beneficio]`.
3. **Anatomía CCC y Pruebas BDD:** Las 3 "C" de Ron Jeffries (*Card, Conversation, Confirmation*), tarjeta física (anverso/reverso), pruebas de caja negra y formato Gherkin (`Dado/Cuando/Entonces`).
4. **Calidad Ágil y Granularidad:** Modelo **I.N.V.E.S.T.** (Bill Wake), *Definition of Done* (DoD), pirámide *Temas* → *Épicas* → *HU* → *Tareas* y tratamiento de Requisitos No Funcionales (RNF).
5. **Planificación, Velocidad y Estimación:** *User Story Mapping* (Jeff Patton), priorización MoSCoW, *Planning Poker*, cálculo numérico de Velocidad con factor de dedicación y asignación de Sprints.
6. **Descomposición y SPIKES:** Los 9 Patrones de División de Richard Lawrence, algoritmo de 3 etapas, repriorización vertical y las 5 Reglas de Oro de los **SPIKES** (Técnicos y Funcionales).
7. **Preparación Directa del Parcial:** Cuaderno de 10 ejercicios prácticos resueltos paso a paso y banco de 20 preguntas cortas de examen con respuesta modelo razonada.

Plataforma **repasaYA** · Docente: María Luisa (Marisa)

Curso Académico 2025–2026 / 2026–2027

# Índice general


**1. Fundamentos de las Historias de Usuario, Roles y Formato Canónico** **5**
- 1.1. ¿Qué son las Historias de Usuario (HU)? . . . 5
  - 1.1.1. Naturaleza Operativa y Eje Comunicación–Colaboración . . . 6
- 1.2. ¿Qué NO son las Historias de Usuario? . . . 6
- 1.3. Comparativa Exhaustiva: Casos de Uso vs. Historias de Usuario . . . 7
- 1.4. ¿Quién escribe las Historias de Usuario y cómo se definen los Roles? . . . 8
  - 1.4.1. Autoría según el Marco Metodológico (Diapositivas 15–16) . . . 8
  - 1.4.2. Identificación y Organización de los Roles del Sistema (Diapositivas 17–18) 9
- 1.5. Formato Canónico de una Historia de Usuario (Diapositivas 19–25) . . . 10

**2. Anatomía «CCC», Pruebas BDD, Granularidad e INVEST** **12**
- 2.1. Elementos de una Historia de Usuario: El Modelo "CCC" (Ron Jeffries) . . . 12
  - 2.1.1. Información y Narración en la Tarjeta (Diapositivas 28–30) . . . 12
- 2.2. Anatomía Canónica de la Tarjeta Física de HU (Diapositivas 31–34) . . . 13
- 2.3. Buscar Detalles de la HU: La Conversación (Diapositivas 35–36) . . . 13
- 2.4. Pruebas de Aceptación: La Confirmación y Formato BDD (Diapositivas 37–43) 14
  - 2.4.1. Reglas y Propiedades de las Pruebas de Aceptación . . . 14
- 2.5. Características de las HU, "HU Terminada" (Done) y ¿Por qué usar HU? . . . 15
  - 2.5.1. Síntesis de Características y Ventajas de las HU (Diapositivas 44 y 48) . 15
  - 2.5.2. HU Terminada: *Definition of Done* (DoD) (Diapositivas 45–47) . . . 16
- 2.6. Granularidad de las HU y Requisitos No Funcionales (Diapositivas 49–54) . . . 16
  - 2.6.1. Jerarquía de Granularidad: Temas, Épicas, HU y Tareas . . . 16
  - 2.6.2. Requisitos No Funcionales (RNF) en Ágil (Diapositivas 51–54) . . . 17
- 2.7. ¿Cuándo es buena una HU? El Modelo I.N.V.E.S.T. (Diapositivas 55–57) . . . 17

**3. Pila de Producto, Priorización, Estimación, Velocidad, Patrones y SPIKES** **19**
- 3.1. Pila de Producto (*Product Backlog*) y Mapa de Historias (Diapositivas 58–64) . 19
  - 3.1.1. Mapa de Historias de Usuario (*User Story Mapping*) (Diapositivas 60–63) 19
  - 3.1.2. Proceso Completo de Definición de las HU (Diapositiva 64) . . . 20
- 3.2. Priorización de las Historias de Usuario (Diapositivas 65–68) . . . 20
  - 3.2.1. Factores de Priorización (Diapositiva 66) . . . 21
  - 3.2.2. Categorías de Criticidad y Método MoSCoW (Diapositivas 67–68) . . . 21
- 3.3. Estimación del Tamaño y *Planning Poker* (Diapositivas 69–73) . . . 21
  - 3.3.1. Puntos de Historia vs. Días Ideales y Cono de Incertidumbre . . . 21
  - 3.3.2. *Planning Poker* (El Juego de la Planificación) (Diapositivas 71–73) . . . 22
- 3.4. Velocidad del Equipo y Planificación de Iteraciones (Diapositivas 74–79) . . . 22
  - 3.4.1. Los 3 Métodos para Calcular la Velocidad (Diapositiva 76) . . . 23
  - 3.4.2. Planificación de Iteraciones y Descomposición en Tareas (Diapositivas 78–82) . . . 23
- 3.5. Los 9 Patrones para Partir HU Grandes (Richard Lawrence) (Diapositivas 83–91) 23
- 3.6. SPIKE: Investigación Técnica y Funcional (Diapositivas 92–96) . . . 25

3.6.1. Tipos de SPIKE y Causas que los Originan (Diapositivas 92–94) . . . . 25

**4. Algoritmo de División, Cuaderno de Ejercicios y Banco de Examen 27**
4.1. Algoritmo en 3 Fases para Dividir una Historia (Diapositiva 96) . . . . . 27
4.2. Cuaderno de 8 Ejercicios Prácticos de Examen Resueltos . . . . . 28
4.3. Banco de 15 Preguntas Cortas de Examen con Respuesta Modelo . . . . . 33
4.4. Bibliografía Oficial del Seminario (Diapositiva 99) . . . . . 35

Bloque 1

# Fundamentos de las Historias de Usuario, Roles y Formato Canónico

**Síntesis del Bloque y Objetivos de Evaluación:**

**Alcance del Bloque:** Diapositivas 1–25 del Seminario 2 (*Unidad 1: Desarrollo Ágil de Software*).
**Conceptos Clave:** Definición canónica de Kent Beck y Martin Fowler, eje Comunicación–Colaboración, qué son y qué no son las Historias de Usuario, comparativa exhaustiva de 13 dimensiones entre Casos de Uso (RUP) e Historias de Usuario (XP/Scrum), autoría según metodología, identificación de roles de negocio y sintaxis canónica de tres bloques (`Como / Quiero / Para`).

## 1.1 ¿Qué son las Historias de Usuario (HU)?

En el contexto de las metodologías ágiles (especialmente *Extreme Programming* –XP– y *Scrum*), las Historias de Usuario constituyen el artefacto central para gestionar las necesidades de los usuarios y conducir el desarrollo incremental del software.

**Definición Canónica: Historia de Usuario (Kent Beck y Martin Fowler)**

*"The story is the unit of functionality in an XP project. We demonstrate progress by delivering tested, integrated code that implements a story. A story must be understandable to customers, developer-testable, valuable to the customer, and small enough that programmers can build half a dozen in an iteration."*
— **Kent Beck & Martin Fowler**, *Planning Extreme Programming.*

De esta definición fundacional se extraen las **cuatro propiedades estructurales** exigibles a toda Historia de Usuario:

1. **Unidad mínima de funcionalidad y progreso:** En un proyecto ágil el avance real no se mide por documentos redactados, diagramas UML ni porcentaje de arquitectura completada, sino exclusivamente mediante la entrega de **código probado e integrado** que implementa una historia.

2. **Comprensible para el cliente (*Understandable to customers*):** Debe redactarse en el lenguaje natural del dominio de negocio, libre de tecnicismos internos de programación.

3. **Probable por los desarrolladores (*Developer-testable*):** Debe permitir diseñar pruebas objetivas que determinen sin ambigüedad si la historia funciona según lo acordado.

4. **Valiosa y de tamaño acotado (*Valuable and small enough*):** Debe aportar un beneficio real al cliente o usuario y ser lo suficientemente pequeña como para que los programadores puedan construir e integrar **alrededor de media docena (entre 4 y 8)** en el transcurso de una única iteración (*Sprint*).

### 1.1.1 Naturaleza Operativa y Eje Comunicación–Colaboración

Desde una perspectiva práctica, una Historia de Usuario puede entenderse como:

- Una **corta declaración de intención** que describe algo que el sistema necesita hacer para el usuario.
- Una **descripción simple de una tarea concreta** que aporta valor directo al usuario o al negocio.
- El resultado directo de las reuniones de toma de contacto con los *stakeholders* (partes interesadas en el producto).




- **Simplicidad deliberada:** Las historias tienen que ser simples, cortas, fáciles de entender, de aceptar y de recordar, **sin que tengan que escribirse todos sus detalles por adelantado**.
- **Soporte en tarjetas:** Se describen usando tarjetas físicas o visuales (una descripción escrita en lenguaje del negocio que identifica y recuerda el requisito y ayuda directamente a su priorización).

**Importante / Pregunta de Examen: Objetivo Primordial de las Historias de Usuario (Diapositiva 8)**

El objetivo de las Historias de Usuario es, principalmente, **lograr la interacción con el equipo y con el usuario, por encima de documentar los requisitos que hay en el sistema**. La tarjeta física o digital no busca reemplazar el diálogo verbal, sino provocarlo y encauzarlo.

## 1.2 ¿Qué NO son las Historias de Usuario?

Uno de los errores más frecuentes en la transición desde metodologías tradicionales en cascada hacia marcos ágiles consiste en confundir las Historias de Usuario con requisitos de software clásicos.

**Principio / Patrón Metodológico: Tres Principios sobre lo que NO es una HU (Diapositiva 9)**

1. **No es un requisito rígido del sistema:** No representa una imposición unilateral del tipo *"algo que el sistema deba hacer obligatoriamente"*; hay que verlo siempre como **algo negociable** (*"necesitaríamos hacer algo similar a esto..."*).
2. **No son requisitos del software, son necesidades de un usuario:** El foco no reside en las tripas tecnológicas del sistema, sino en el problema u objetivo que el

usuario humano necesita resolver.

3. **No son una especificación detallada, son "promesas de una conversación":** Expresión acuñada por Alistair Cockburn. Una HU es un recordatorio (*token*) para mantener una conversación futura con el cliente justo antes o durante su implementación.

## 1.3 Comparativa Exhaustiva: Casos de Uso vs. Historias de Usuario

En los exámenes parciales de teoría es habitual que se solicite comparar los **Casos de Uso** (propios de metodologías guiadas por planes como *RUP – Rational Unified Process*) frente a las **Historias de Usuario** (propias de *Extreme Programming* y *Scrum*). A continuación se recoge íntegramente la comparativa oficial de las diapositivas 11 a 14 en sus **13 dimensiones**:

| CONCEPTO | CASOS DE USO (RUP) | HISTORIAS DE USUARIO (XP / Scrum) |
|---|---|---|
| **Objetivo** | Modelar detalladamente la interacción formal entre un "actor" y el sistema. | Redactar una descripción breve de una funcionalidad tal y como la percibe el usuario. |
| **Estructura** | Plantillas densas, múltiples páginas, flujos alternativos formales y conceptos técnicos. | 1 ó 2 frases cortas y consistentes expresadas en el lenguaje natural del usuario. |
| **Planificación** | **No se utilizan para planificar** directamente las iteraciones. | **Se utilizan directamente para planificar**, estimar y medir la velocidad. |
| **Agilidad** | Requieren mucho tiempo previo para el análisis y la redacción de plantillas predefinidas. | Se pueden escribir y ajustar en **pocos minutos**. |
| **Comprensión** | Suelen ser de difícil comprensión, incluso para el propio personal técnico. | **Fáciles de leer y comprender** tanto por usuarios como por desarrolladores. |
| **Mantenimiento** | Documentos pesados con cientos de páginas. Muy difíciles y costosos de mantener. | **Muy fáciles de mantener**, reordenar o descartar tras su implementación. |
| **Comunicación** | Modelo textual exhaustivo asociado con diagramas: *todo tiene que estar escrito.* | Basada en la **comunicación verbal** y orientada a la colaboración y discusión para clarificar detalles. |
| **Soporte** | Escrito en documentos extensos con el objetivo de que sean archivados como documentos de referencia. | Escritas en **tarjetas** con el objetivo de que sean manipuladas y usadas directamente por el equipo. |
| **Duración** | Puede ser implementado a lo largo de **varias iteraciones**. | Debe ser implementada y probada íntegramente en **una única iteración**. |

| CONCEPTO | CASOS DE USO (RUP) | HISTORIAS DE USUARIO (XP / Scrum) |
|---|---|---|
| **Autores** | Definidos por analistas de sistemas, consultores, desarrolladores, etc. | Posibilidad real de ser definidas directamente por **usuarios y clientes**. |
| **Pruebas** | La definición de pruebas se redacta en documentación separada e independiente. | Contienen las **pruebas de aceptación en el reverso** de la propia tarjeta. |
| **Contexto** | Proporcionan una visión más general del sistema y de la integración arquitectónica en él. | Proporcionan una visión global menos obvia; por eso las **pruebas continuas y el feedback de los usuarios** son tan importantes en las metodologías ágiles. |
| **Metodología** | Asociado históricamente con **RUP** (*Rational Unified Process*). | Asociado con **Programación Extrema (XP)** y **Scrum**. |

## 1.4 ¿Quién escribe las Historias de Usuario y cómo se definen los Roles?

### 1.4.1 Autoría según el Marco Metodológico (Diapositivas 15–16)

La responsabilidad de redactar las Historias de Usuario varía entre la teoría metodológica pura y la práctica diaria de los equipos:

- **En Programación Extrema (XP):** Las escribe directamente **el cliente** (*On-site Customer*), quien forma parte presencial del equipo.
- **En Scrum:** Las escribe y gestiona el **Product Owner (PO)**, contando con la ayuda activa de clientes, *stakeholders* y el equipo de desarrollo.
- **En la realidad práctica:** Las puede escribir **cualquier miembro del equipo de desarrollo** que posea conocimiento suficiente en el dominio del problema, aunque el Product Owner sigue siendo el responsable de validarlas y priorizarlas en la pila de producto.

**Definición Canónica: El "Customer Team" y los Stakeholders**

Deben escribir y refinar las HU todos aquellos perfiles capaces de **expresar las necesidades reales del sistema** y participar en las reuniones para detallarlas y priorizarlas. Este equipo extendido de cliente (*Customer Team / Stakeholders*) engloba:

1. **Clientes reales (usuarios finales):** Quienes utilizarán directamente la aplicación.
2. **Product Manager / Product Owner:** Quien vela por el valor de negocio y el retorno de inversión.
3. **Probadores (QA / Testers):** Quienes aportan visión crítica sobre casos límite y verificación.
4. **Diseñadores de interacción (UX / UI):** Quienes garantizan la usabilidad y experiencia de uso.

### 1.4.2 Identificación y Organización de los Roles del Sistema (Diapositivas 17–18)

Antes de redactar historias, es imprescindible **identificar y organizar a los usuarios del sistema** en **Roles** o *Personas*, evitando el uso indiscriminado del término genérico "usuario":

- Se recomienda realizar una sesión de ***Brainstorming*** (tormenta de ideas) en equipo para identificar todos los perfiles que interactúan con el software.
- Para cada rol identificado se deben caracterizar:
  - **Frecuencia de uso del software** (diario, semanal, esporádico, puntual).
  - **Objetivos principales** que se plantea al usar el sistema.
  - **Forma habitual de acceso** (escritorio, móvil, automatizado) y nivel de pericia en el dominio.

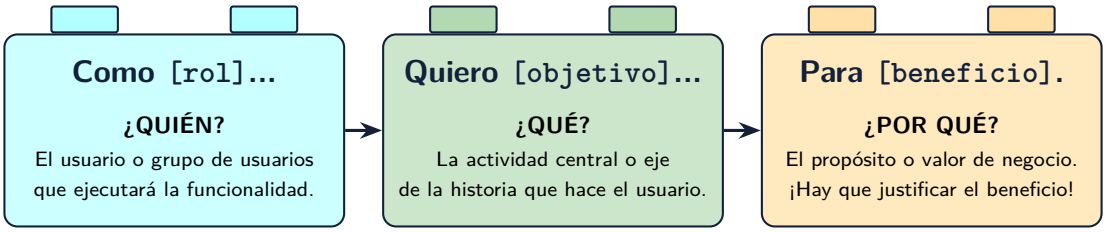
**Ejemplo de Clase / Dominio: Tabla de Roles del Sistema: Agencia de Viajes (Diapositiva 18)**

En el caso práctico del portal web de una **Agencia de Viajes**, una sesión de *Brainstorming* permite distinguir al menos **11 roles diferenciados**, cada uno con necesidades propias:

| Rol Identificado | Objetivos y Comportamiento en el Sistema |
|---|---|
| **Cliente** | Busca alojamientos, compara precios, reserva habitaciones y gestiona cancelaciones. |
| **Socio (Cliente fidelizado)** | Accede a ofertas exclusivas, acumula puntos y utiliza métodos de pago guardados. |
| **Planificador de viaje** | Organiza viajes para terceros; requiere fotos verificadas y opiniones detalladas de calidad. |
| **Comentarista** | Viajero que publica reseñas, valoraciones y fotografías tras concluir su estancia. |
| **Hoteles (Proveedor)** | Da de alta habitaciones, actualiza cupos de disponibilidad y recibe avisos de reservas. |
| **Afiliado** | Enlaza ofertas desde portales externos a cambio de comisiones por venta referida. |
| **Agencias de Viajes** | Gestiona reservas de grupos o paquetes turísticos completos en condiciones mayoristas. |
| **Administrador** | Configura parámetros globales del sistema, gestiona usuarios y audita la seguridad. |
| **Distribuidoras** | Mayoristas que integran inventarios masivos de vuelos, estancias y paquetes. |
| **Social Media Manager** | Conecta campañas, opiniones destacadas y promociones con redes sociales. |
| **Publicista** | Gestiona espacios promocionales, banners y posicionamiento patrocinado de ofertas. |

## 1.5 Formato Canónico de una Historia de Usuario (Diapositivas 19–25)

Para redactar Historias de Usuario consistentes y orientadas al valor, se emplea la plantilla estándar de tres cláusulas (popularizada por Mike Cohn / Connextra):




- `[Como:]` **(Quién):** Identifica al usuario o grupo de usuarios que participan en la historia; normalmente es un **"rol de usuario"**, el que usará directamente la funcionalidad descrita.
- `[Quiero:]` **(Qué):** Es la actividad o característica central, el eje de la historia: qué necesita hacer el usuario con el sistema.
- `[Para:]` **(Por qué):** Es el propósito de la historia, la meta o **valor de negocio/beneficio** que quiere alcanzar el usuario al ejecutar la actividad. Ayuda de forma decisiva a priorizar la historia y a entender su verdadero fin.

**Ejemplo de Clase / Dominio: Catálogo Oficial: Web de Viajes y Librería (Diapositivas 21–25)**

**A) Ejemplos en formato canónico completo (`Como / Quiero / Para`):**

1. *"**Como** cliente, **quiero** reservar una habitación de hotel **para** garantizar mi estancia durante mis fechas de viaje y planificar mi viaje con tranquilidad."*
2. *"**Como** cliente, **quiero** cancelar una reserva previa **para** poder realizar una nueva reserva en fechas o destinos diferentes que se adapten mejor a mi agenda (o evitar cargos si mis planes cambian)."*
3. *"**Como** cliente, **quiero** buscar hoteles cerca del mar **para** disfrutar de acceso rápido a la playa."*
4. *"**Como** cliente, **quiero** comparar las características y precios de dos hoteles **para** evaluar cuál de ellos ofrece la mejor relación calidad-precio según mis necesidades."*
5. *"**Como** cliente, **quiero** filtrar los hoteles en los que admiten perros **para** descartar rápidamente los alojamientos que no acepten animales y no perder tiempo revisando fichas innecesarias."*
6. *"**Como** planificador de viajes, **quiero** ver fotos de un hotel **para** verificar las instalaciones, la calidad de las habitaciones y asegurarme de que cumple con las expectativas de mis clientes antes de realizar la reserva."*
7. *"**Como** planificador de viajes, **quiero** conocer las opiniones de otros viajeros **para** anticipar posibles inconvenientes y recomendar a mis clientes únicamente alojamientos con alta satisfacción comprobada."*

**B) Ejemplos en formato narrativo libre directo (Diapositiva 25):**

- *"Un usuario puede buscar un libro insertando cualquier combinación de título o autor."*

- *"Un usuario puede poner los libros en el carrito y sucesivamente comprarlos."*
- *"Un usuario puede escoger acompañar el envío de un libro con su propia dedicatoria."*
- *"Un supervisor puede ver un informe de las ventas realizadas en el día."*

Bloque 2

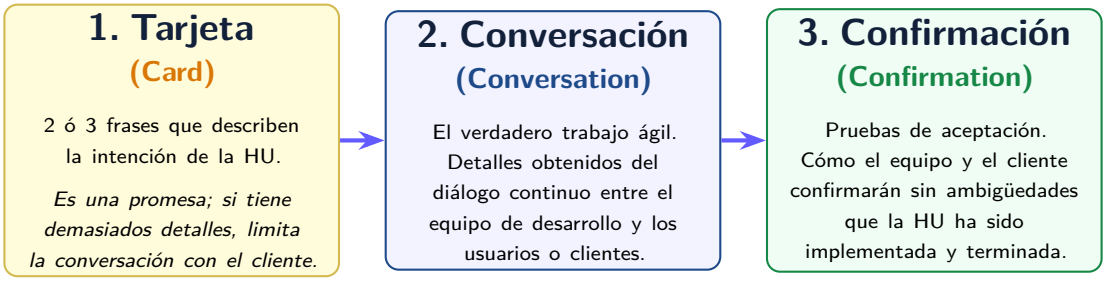
# Anatomía «CCC», Pruebas de Aceptación, Granularidad y Modelo INVEST

**Síntesis del Bloque y Objetivos de Evaluación:**

**Alcance del Bloque:** Diapositivas 26–57 del Seminario 2.
**Conceptos Clave:** Fórmula **CCC** de Ron Jeffries (*Card, Conversation, Confirmation*), anatomía física de la tarjeta (anverso y reverso), búsqueda de detalles en la Conversación, Pruebas de Aceptación de caja negra, formato BDD (`Dado / Cuando / Entonces`), Definición de Terminado (*Definition of Done*), jerarquía de granularidad (*Temas, Épicas, HU, Tareas*), gestión de Requisitos No Funcionales (RNF) y modelo **I.N.V.E.S.T.** de Bill Wake.

## 2.1 Elementos de una Historia de Usuario: El Modelo "CCC" (Ron Jeffries)

En el año 2001, **Ron Jeffries** (uno de los tres fundadores de *Extreme Programming*) formuló la regla de las **Tres "C"** para explicar que una Historia de Usuario es mucho más que el texto escrito en una cartulina:



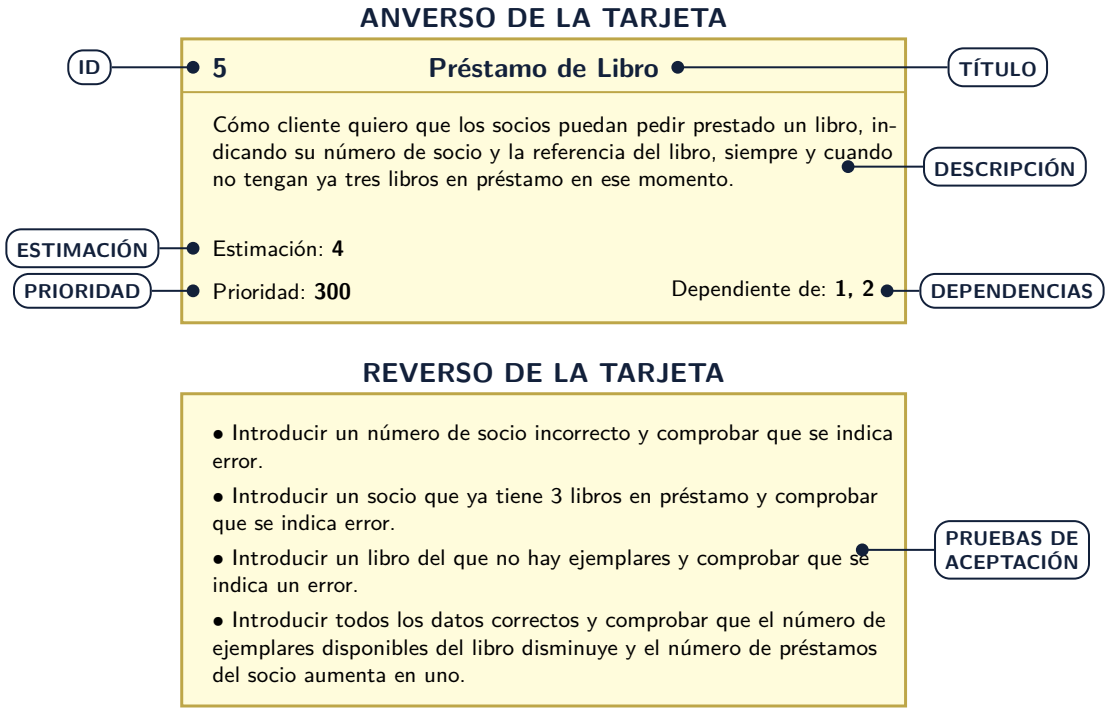

### 2.1.1 Información y Narración en la Tarjeta (Diapositivas 28–30)

- **Descripción corta sin detalles exhaustivos:** La tarjeta presenta una descripción concisa junto con pequeñas notas iniciales (p. ej., *"Un cliente puede pagar un viaje usando una tarjeta de crédito. Nota: Se pueden aceptar Visa, MasterCard y American Express; discutir si se aceptan otras más"*).
- **Peligro del exceso de detalle: Una tarjeta con demasiados detalles limita la conversación con el cliente**, induciendo a los programadores a asumir que todo está decidido y bloqueando soluciones más creativas o simples.

- **Delimitación del alcance:** La parte exacta a abordar por la HU se delimita en la conversación y se plasma usando las **pruebas de aceptación**, las cuales recogen:
  1. Flujos alternativos en la actividad.
  2. Límites de aceptación (condiciones de contorno y restricciones).
  3. Pruebas de aceptación verificables por parte de los clientes.

## 2.2 Anatomía Canónica de la Tarjeta Física de HU (Diapositivas 31–34)

En la diapositiva 31 se establece la estructura oficial de una tarjeta de Historia de Usuario dividida en **Anverso** (identificación, descripción, estimación, prioridad y dependencias) y **Reverso** (pruebas de aceptación):




En herramientas digitales como **Jira** (diapositiva 34) o plantillas corporativas (diapositivas 32–33), estos mismos campos se representan mediante atributos equivalentes: identificador único, título, narrativa `Como/Quiero/Para`, condiciones de completitud, prioridad numérica, coste/puntos estimados, iteración asignada y persona encargada.

## 2.3 Buscar Detalles de la HU: La Conversación (Diapositivas 35–36)

**Importante / Pregunta de Examen: Principio Clave de la Conversación (Diapositiva 35)**

**"La tarjeta no contiene estas respuestas; su propósito es PROVOCAR estas preguntas."**

Por ejemplo, ante la tarjeta mínima *"Como cliente, quiero cancelar una reserva de hotel"*, la etapa de **Conversación** entre desarrolladores y usuarios debe hacer aflorar preguntas como:

- *¿Qué coste de penalización tiene la cancelación?*
- *¿Todos los niveles de clientes tienen el mismo coste o varía si es cliente Premium?*
- *¿Se puede cancelar en cualquier momento o hay límite de horas previo?*
- *¿Se permite una cancelación total y también parcial?*
- *¿A quién (departamento/hotel/cliente) se le notifica automáticamente?*

**¿Por qué son vitales las conversaciones? (Diapositiva 36):**

1. Ayudan al equipo a comprender las necesidades y deseos reales del usuario y a producir **soluciones creativas** a los problemas.
2. Favorecen la **solidez de las relaciones** de confianza entre los miembros del equipo y los usuarios.
3. Deben reflejar la **voz de todos los interesados** (*stakeholders*) en el producto.

## 2.4 Pruebas de Aceptación: La Confirmación y Formato BDD (Diapositivas 37–43)

**Definición Canónica: Criterios o Pruebas de Aceptación**

Los criterios o pruebas de aceptación son las **condiciones objetivas que una Historia de Usuario debe cumplir para considerarse completada** cuando es desarrollada y desplegada por el equipo. Funcionan como una guía que asegura que lo que se construye es realmente lo que el negocio necesita.

### 2.4.1 Reglas y Propiedades de las Pruebas de Aceptación

- **Claridad y precisión:** No deben dejar espacio a interpretaciones subjetivas ni ambigüedades.
- **Independencia de la implementación:** No describen *cómo* programarlo internamente, sino ***qué* resultado observable se espera**.
- **Lenguaje de negocio:** Se escriben siempre en el lenguaje del ámbito de negocios.
- **Propiedad del Product Owner:** Aunque cualquiera en el equipo puede escribir una prueba de aceptación, **el Product Owner es el principal propietario de la prueba**.
- **Pruebas de Caja Negra (*Black-box testing*):** Solo verifican que las salidas del sistema cumplen las condiciones de satisfacción indicadas ante determinadas entradas, sin preocuparse de cómo se consiguen internamente los resultados.
- **Sincronía de iteración:** Se implementan y verifican **en el transcurso de la misma iteración** en la que se implementa la propia Historia de Usuario.
- **Complementariedad con Pruebas Unitarias:** El equipo de desarrollo realiza **pruebas de unidad** (caja blanca) sobre los módulos de código desarrollados, *independientemente* de las pruebas de aceptación definidas para la HU.

**Ejemplo de Clase / Dominio: Criterios de Aceptación Clásicos y Formato BDD (Diapositivas 40–43)**

**Caso 1 (Lista de comprobación – Cancelar reserva previa):**

- Comprobar que un **Cliente Premium** puede cancelar en el mismo día sin coste adicional.
- Comprobar que a un **Cliente No-Premium** se le carga un 10 % al cancelar el mismo día.
- Confirmar que se envía un correo electrónico de confirmación al cliente.
- Confirmar que el hotel recibe notificación automática de la cancelación.

**Caso 2 (Lista de comprobación – Pagar viaje con tarjeta de crédito):**

- Funciona correctamente con tarjetas *Visa, MasterCard y American Express.*
- Falla (no permitida) con tarjeta *Diner's Club.*
- Rechaza adecuadamente una tarjeta cuya fecha haya expirado.
- Maneja correctamente diferentes montos de cargo y falla ante un número de tarjeta inválido o vacío.

**Caso 3 (Formato BDD – Behavior-Driven Development: `Dado / Cuando / Entonces`):**

**Dado** *[contexto inicial / estado previo del sistema]*
⟶ **Cuando** *[acción realizada por el usuario o evento desencadenante]*
⟶ **Entonces** *[resultado esperado / respuesta observable del sistema]*

```
DADO un sistema donde existe un usuario con nombre "idprueba"
CUANDO alguien intenta registrarse con nombre "idprueba"
ENTONCES el sistema muestra un mensaje indicando que el nombre "
    idprueba" ya existe.
```

## 2.5 Características de las HU, "HU Terminada" (Done) y ¿Por qué usar HU?

### 2.5.1 Síntesis de Características y Ventajas de las HU (Diapositivas 44 y 48)

- Son **cortas y fáciles de leer**, entendibles por desarrolladores, interesados y usuarios.
- Representan **incrementos pequeños de funcionalidad valorada**, desarrollables en un período de días o pocas semanas.
- Son **relativamente fáciles de estimar** porque el esfuerzo de implementación puede determinarse rápidamente.
- Se gestionan en **listas organizadas dinámicas** (no en documentos pesados) que pueden reordenarse a medida que emerge nueva información (**desarrollo oportunista / requisitos emergentes**).
- Necesitan **poco o ningún mantenimiento** y se pueden desechar con seguridad después de la implementación: las HU y el propio código creado sirven como documentación

incremental.

- Permiten mantener una **relación cercana con el cliente** y aplazar los detalles para cuando sean realmente necesarios.

### 2.5.2 HU Terminada: *Definition of Done* (DoD) (Diapositivas 45–47)

**Definición Canónica: Definición de "Terminada" (Definition of Done – DoD)**

La definición de "terminado" (*Done*) es un **acuerdo formal de calidad entre los desarrolladores y los clientes / Product Owner**. Se puede establecer una definición de "terminada" global para todas las HU del proyecto o un "terminado" específico adicional para determinadas HU.

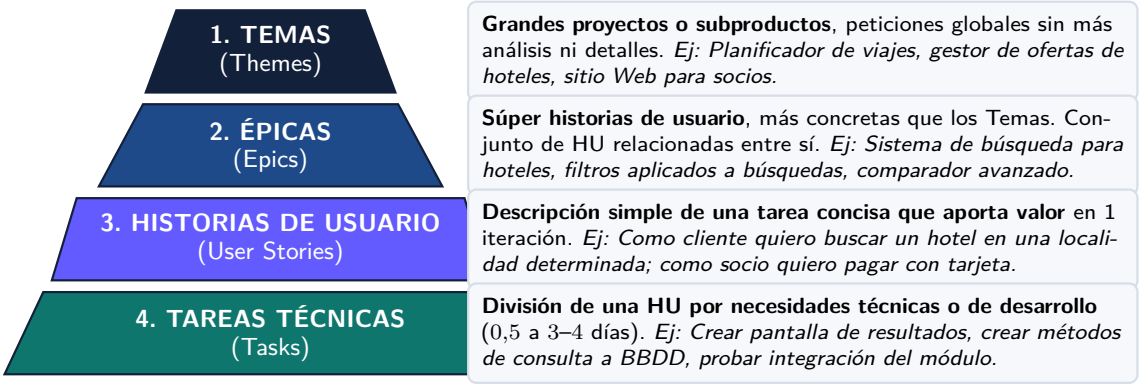
**los 6 aspectos esenciales a considerar en la definición de "terminada" (Diapositiva 47):**

1. **Tipos y grado de prueba:** Superación de pruebas de unidad y pruebas de integración.
2. **Calidad del código:** Código documentado, revisado por pares (*code review*) y sin incidencias críticas.
3. **Entorno de despliegue donde concluye el trabajo:** En preproducción (*prepro*), *testing* o producción.
4. **Nivel de documentación** técnica y de usuario entregada.
5. **Nivel de calidad del producto software:** Cumplimiento de métricas de análisis estático, cobertura, etc.
6. **Criterios de aceptación comprobados y aprobados** formalmente por el Product Owner.

## 2.6 Granularidad de las HU y Requisitos No Funcionales (Diapositivas 49–54)

### 2.6.1 Jerarquía de Granularidad: Temas, Épicas, HU y Tareas

En un proyecto ágil coexisten elementos de distinto tamaño y nivel de abstracción. A medida que una necesidad se acerca a la parte alta de la pila para ser implementada en la siguiente iteración, se descompone desde conceptos amplios hacia unidades más pequeñas:



### 2.6.2 Requisitos No Funcionales (RNF) en Ágil (Diapositivas 51–54)

Los **Requisitos No Funcionales (RNF)** describen cualidades o restricciones del sistema (rendimiento, disponibilidad, seguridad, usabilidad) que no se relacionan de forma directa con un comportamiento funcional concreto, pudiendo aplicarse **al sistema en su conjunto** o **a determinadas funciones específicas**.

**Principio / Patrón Metodológico: Las 4+1 Formas de Tratar los Requisitos No Funcionales (Diapositiva 52)**

En un proyecto ágil basado en Historias de Usuario, los RNF pueden tratarse como:

1. **Restricciones asignadas a cualquiera de los elementos de las pilas** (Temas, Épicas, Historias de Usuario o Tareas) o redactadas directamente con formato de HU.
2. **Criterios de aceptación** asociados a una Historia de Usuario funcional.
3. **Parte de la definición de terminada (*Definition of Done*)** cuando afectan globalmente a todas las historias.
4. **Documento de especificación específico** para los requisitos no funcionales globales al sistema.

**Ejemplo de Clase / Dominio: Ejemplos Oficiales de Requisitos No Funcionales (Diapositivas 53–54)**

**A) RNF expresados directamente con plantilla de Historia de Usuario:**

- *"**Como** comprador, **quiero** que el sitio web esté disponible el **98 % de las veces** que intento acceder a él **para** poder realizar mi compra y no tener que buscar otro lugar para adquirir el producto."* (Disponibilidad).
- *"**Como** cliente de banca online, **quiero** que el saldo de mi cuenta corriente se muestre en **5 segundos** cuando lo solicite **para** no frustrarme y poder ver si tengo fondos para pagar mis facturas."* (Rendimiento / Tiempo de respuesta).

**B) RNF expresados como Criterios de Aceptación dentro de una HU funcional:**

- **HU Funcional:** *"Como analista financiero, quiero ver las transacciones mensuales de mis clientes para poder aconsejarles sobre su salud financiera."*
- **Criterio RNF 1 (Rendimiento):** *El sistema muestra todas las transacciones que cumplen los parámetros de búsqueda en los **10 segundos** siguientes a la recepción de la solicitud.*
- **Criterio RNF 2 (Seguridad):** *Las transacciones de los clientes etiquetados como confidenciales sólo se muestran a los usuarios con **nivel de seguridad** 2.*

## 2.7 ¿Cuándo es buena una HU? El Modelo I.N.V.E.S.T. (Diapositivas 55–57)

Propuesto por **Bill Wake**, el acrónimo **I.N.V.E.S.T.** define las seis cualidades que debe reunir toda Historia de Usuario de calidad antes de entrar en una iteración:

| Letra | Atributo | Definición Oficial y Pregunta de Control (Diapositivas 56–57) |
|---|---|---|
| **I** | **Independiente** (***Independent***) | Una historia debería ser independiente de otras (lo más posible). *¿Puede sostenerse por sí misma sin depender rígidamente de otras historias?* |
| **N** | **Negociable** (***Negotiable***) | Las HU no son un contrato cerrado ni un requerimiento rígido que el software deba implementar literalmente. *¿Es una propuesta abierta a la discusión colaborativa en lugar de un contrato fijo?* |
| **V** | **Valiosa** (***Valuable***) | Tiene que tener valor para el cliente (para el usuario o para el comprador). El cliente las escribe, no son un contrato y son negociables. *¿Aporta un beneficio tangible, demostrable e identificable al usuario o al negocio?* |
| **E** | **Estimable** (***Estimable***) | Debe poder estimarse para permitir que se pueda priorizar y planificar la historia. *¿Tiene el equipo suficiente información y contexto para calcular su complejidad?* |
| **S** | **Suscinta / Pequeña** (***Small***) | Una buena historia debe ser pequeña en esfuerzo (**2–3 personas/semana de trabajo**, o entre **1/10** y **1/6** de la velocidad). *¿Es lo suficientemente compacta para completarse con éxito dentro de un solo Sprint?* |
| **T** | **Testeable** (***Testable***) | Una historia necesita poder probarse objetivamente para que ocurra la etapa de "Confirmación". *¿Posee criterios de aceptación claros que permitan validar su correcto funcionamiento?* |

Bloque 3

# Pila de Producto, Priorización, Estimación, Velocidad, Patrones y SPIKES

**Síntesis del Bloque y Objetivos de Evaluación:**

**Alcance del Bloque:** Diapositivas 58–96 del Seminario 2.
**Conceptos Clave:** *Product Backlog*, Mapa de Historias de Usuario (*User Story Mapping*), flujo de 8 pasos para definir HU, priorización y método **MoSCoW**, Puntos de Historia vs. Días Ideales, Cono de Incertidumbre, *Planning Poker*, Velocidad del equipo y fórmula de cálculo de recursos con factor de dedicación, planificación de iteraciones, descomposición en tareas, los **9 Patrones de División de Richard Lawrence** y gestión de **SPIKES** (técnicos y funcionales).

## 3.1 Pila de Producto (*Product Backlog*) y Mapa de Historias (Diapositivas 58–64)

**Definición Canónica: Pila de Producto (Product Backlog)**

*"Lista priorizada de Historias de Usuario que contiene toda la funcionalidad deseable del producto."*

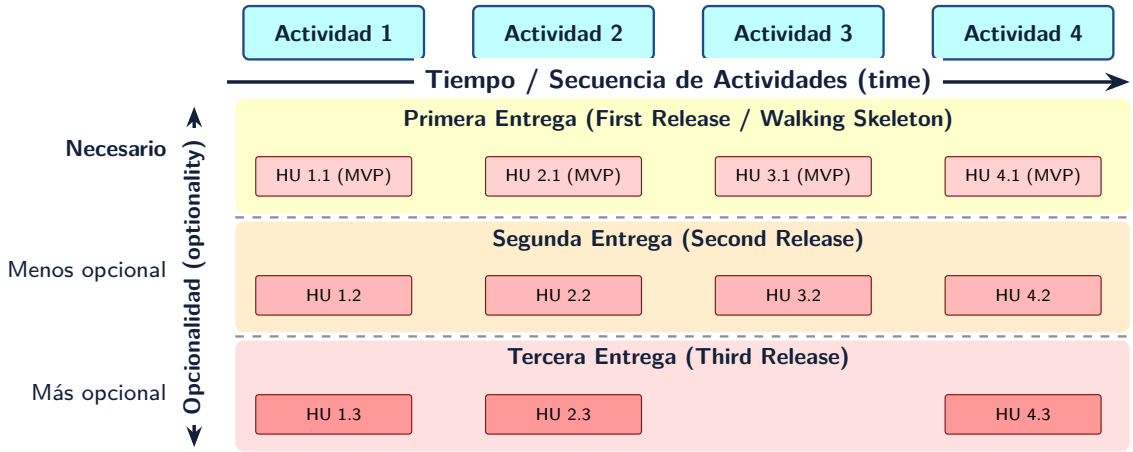
**Propiedades esenciales del Product Backlog (Diapositiva 59):**

1. **Ordenada por prioridad:** El orden refleja el valor que le da el cliente a cada HU (la medida en que permite satisfacer sus necesidades de negocio).
2. **Cobertura global:** Cubre todas las funcionalidades necesarias conocidas hasta el momento.
3. **Viva y dinámica:** Se revisa y actualiza frecuentemente (*Backlog Refinement / Grooming*).
4. **Herramienta de planificación y gestión:** Constituye la base única para planificar entregas e iteraciones.

### 3.1.1 Mapa de Historias de Usuario (*User Story Mapping*) (Diapositivas 60–63)

Propuesto por **Jeff Patton**, el **Mapa de Historias de Usuario** organiza el *Product Backlog* en dos dimensiones (frente a una lista plana unidimensional) para conservar la visión global del flujo de trabajo del usuario:

- **Eje Horizontal (*Time / Flujo cronológico*):** De izquierda a derecha se disponen las actividades principales (*Backbone* o espina dorsal) en el orden temporal en que el usuario interactúa con el sistema.
- **Eje Vertical (*Optionality / Necesidad y prioridad*):** De arriba hacia abajo se ordenan las historias desde las **estrictamente necesarias** (*necessary*) hasta las **más opcionales** (*more optional*).
- **Cortes Horizontales (*Releases / Entregas*):** Trazando líneas horizontales se agrupan las historias en entregas sucesivas (*first release / MVP*, *second release*, *third release*), asegurando que desde la primera entrega el sistema ofrezca un recorrido funcional de extremo a extremo (*Walking Skeleton*).




### 3.1.2 Proceso Completo de Definición de las HU (Diapositiva 64)

El flujo de trabajo ordenado de principio a fin para definir y preparar las Historias de Usuario consta de **8 pasos secuenciales**:

**Paso 1:** **Toma de requisitos** en reuniones con usuarios/clientes (entrevistas, cuestionarios, observación).

**Paso 2:** **Informar de la visión general** transmitida por el cliente a todo el equipo.

**Paso 3:** **Crear las HU** (talleres o reuniones colaborativas de escritura de HU).

**Paso 4:** **Priorizar las HU** según valor de negocio y urgencia.

**Paso 5:** **Actuar sobre cada HU** mediante un estudio funcional y técnico realizado en equipo (**Conversación**).

**Paso 6:** **Definir los criterios de aceptación** (**Confirmación**).

**Paso 7:** **Estimar el tamaño de las HU** (en Puntos de Historia o Días Ideales).

**Paso 8:** **Descomponer las HU en tareas** técnicas individuales para la iteración.

## 3.2 Priorización de las Historias de Usuario (Diapositivas 65–68)

Priorizar las Historias de Usuario es un paso crítico porque determina el orden en que deben desarrollarse para **entregar el máximo valor a los usuarios finales y al negocio lo más rápido posible**.

### 3.2.1 Factores de Priorización (Diapositiva 66)

- **Valor para el cliente (Valor de Negocio):** Retorno económico o estratégico directo.
- **Valor para el usuario:** Combinando tanto las *HU importantes para un número alto de usuarios* como las *HU críticas para un número pequeño de usuarios muy importantes.*
- **Riesgo e incertidumbre:** Priorizar temprano historias con alto riesgo para validar la viabilidad técnica o funcional cuanto antes.
- **Dependencias:** HU estrechamente relacionadas con otras HU priorizadas previamente.
- **Tamaño y esfuerzo:** Relación coste/beneficio de implementar la historia.

**Importante / Pregunta de Examen: Regla de Oro de Examen sobre Prioridad y Tamaño (Diapositiva 66)**

**"La prioridad puede cambiar; el tamaño NO."**
Las Historias de Usuario son negociables con el cliente y su prioridad evoluciona continuamente a lo largo del proyecto según el mercado, pero su tamaño relativo (complejidad técnica y esfuerzo de implementación) permanece invariante mientras no cambie su alcance funcional.

### 3.2.2 Categorías de Criticidad y Método MoSCoW (Diapositivas 67–68)

| Clasificación Clásica | Método MoSCoW | Significado e Impacto en las Entregas del Proyecto |
|---|---|---|
| **1. Requeridas o críticas** | M – Must | Sin las cuales la solución no puede vivir. Se debe completar obligatoriamente este requerimiento para finalizar el proyecto (**primeras entregas**). |
| **2. Importantes** | S – Should | Sin las cuales el sistema puede vivir durante algún tiempo. Se debe completar por todos los medios, pero el éxito del proyecto no depende vitalmente de él (**entregas intermedias**). |
| **3. Opcionales** | C – Could | Si hay tiempo y presupuesto se construyen. Se debería completar si su implementación no afecta a la consecución de los objetivos principales (**entregas finales**). |
| **4. No se construirán** | W – Would / Won't | Por restricciones de presupuesto/tiempo o por carecer de valor actual; se posponen si sobra tiempo o para **futuras versiones** del sistema. |

## 3.3 Estimación del Tamaño y *Planning Poker* (Diapositivas 69–73)

### 3.3.1 Puntos de Historia vs. Días Ideales y Cono de Incertidumbre

La estimación del tamaño de las HU es realizada siempre **por el equipo de desarrollo** empleando una de dos métricas principales:

1. **Puntos de Historia (*Story Points* – PH):** Miden el **tamaño relativo** de una historia comparado con otras historias de referencia estimadas por el mismo equipo:

$$\textbf{PH} = \text{Esfuerzo necesario} + \text{Complejidad} + \text{Riesgo} \;/\; \text{Incertidumbre}$$

2. **Días Ideales (*Ideal Days*):** Representan días de trabajo dedicados al 100 % sin interrupciones, reuniones, correos ni imprevistos.

A través del **Cono de Incertidumbre** (diapositiva 70), una estimación inicial de grano grueso (tallas de camiseta: `XS, S, M, L, XL`) se refina en **Puntos de Historia** (serie de Fibonacci: `0, 1, 2, 3, 5, 8, 13, 20, ?`) y, finalmente, al descomponer la historia dentro del Sprint, se traduce en **Horas** concretas de tarea.

### 3.3.2 *Planning Poker* (El Juego de la Planificación) (Diapositivas 71–73)

**Definición Canónica: Planning Poker**

Técnica colaborativa para realizar una estimación basada en el **consenso entre los estimadores**, utilizando un mazo de cartas con la **Secuencia de Fibonacci**:

- **Mazo Fibonacci puro:** (0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89)
- **Mazo comercial modificado:** (0, $\frac{1}{2}$, 1, 2, 3, 5, 8, 13, 20, 40, 100, ?, café)

**Protocolo paso a paso de una sesión de *Planning Poker* (Diapositiva 72–73):**

1. **Moderador que NO estima:** Existe un moderador (habitualmente el Scrum Master) que gestiona la reunión y controla los tiempos asignados, pero **no emite estimaciones**.
2. **Conversación e introducción:** El desarrollador con mayor conocimiento de una característica proporciona una breve introducción. El equipo hace preguntas y discute para aclarar supuestos y riesgos.
3. **Votación simultánea sin anclaje:** Cada miembro selecciona una carta del mazo y la coloca **boca abajo**. A continuación, **todo el mundo muestra sus cartas a la vez** (evitando que el primer voto condicione a los demás).
4. **Alineación y debate de extremos:** A las personas con estimaciones fuera del promedio (**la más alta y la más baja**, p. ej., un `2` y un `13`) se les concede tiempo para justificar su estimación y continúa la discusión.
5. **Repetición hasta consenso:** Se repite el proceso de votación hasta alcanzar un consenso. *El objetivo real no es solo el número, sino alinear el conocimiento del equipo para descubrir riesgos ocultos.*

## 3.4 Velocidad del Equipo y Planificación de Iteraciones (Diapositivas 74–79)

**Definición Canónica: Velocidad del Equipo (Team Velocity)**

La **Velocidad** es la **cantidad de Puntos de Historia (PH) que un equipo puede completar satisfactoriamente en una única iteración**. Se calcula por observación empírica a lo largo de varias iteraciones y permite planificar qué HU se abordarán en cada Sprint.

### 3.4.1 Los 3 Métodos para Calcular la Velocidad (Diapositiva 76)

1. **Utilizar datos históricos (*"El tiempo que hizo ayer" / Yesterday's Weather*):** Tomar el promedio de puntos completados en iteraciones pasadas del mismo equipo.
2. **Ejecutar una iteración inicial:** Realizar un primer Sprint de prueba y utilizar la velocidad real obtenida en esa iteración.
3. **Hacer un pronóstico (*"El cálculo de recursos"*):** Calcular matemáticamente la capacidad teórica cuando el equipo es nuevo y aplicar un factor de dedicación.

**Ejercicio Resuelto de Examen: Cálculo Oficial de la Velocidad mediante Cálculo de Recursos (Diapositiva 77)**

**Datos del problema:**

- Equipo de desarrollo de **5 miembros** (todos trabajan las mismas horas).
- Iteraciones de **2 semanas** $\Longrightarrow$ **1 iteración = 10 días de trabajo**.
- Equivalencia en el entorno de trabajo: **1 PH = 1 día de trabajo ideal**.
- Factor de dedicación efectiva: **80 %**.

**Fórmula y Resolución Paso a Paso:**

$$\text{Días por iteración} = 5 \text{ programadores} \times 10 \text{ días} = \mathbf{50} \text{ días de trabajo}$$

$$\textbf{Velocidad} = \text{Días totales} \times \text{Factor de dedicación} = 50 \times 80\,\% = \mathbf{40} \text{ PH por iteración}$$

### 3.4.2 Planificación de Iteraciones y Descomposición en Tareas (Diapositivas 78–82)

En la **Planificación de Iteraciones**, se asignan las HU priorizadas a las iteraciones garantizando que la suma de Puntos de Historia de cada iteración no supere la Velocidad del equipo (p. ej., con Velocidad = 13 PH, la Iteración 1 agrupa las historias $A(3) + B(5) + C(5) = 13$ PH; la Iteración 2 agrupa $D(3) + E(1) + F(8) = 12$ PH, etc.).

Posteriormente, cada HU del Sprint se descompone en **Tareas Técnicas** individuales siguiendo cinco reglas (diapositiva 80):

1. Deben poder ser tratadas de forma individual por miembros del equipo de desarrollo.
2. **Duración acotada:** La duración de una tarea debe estar entre **medio día (0.5 días) y 3 ó 4 días de trabajo**.
3. Una vez terminada la tarea se debe generar un **producto o componente entregable** verificable.
4. No se debe dedicar excesivo tiempo previo a estudiar microdetalles de las tareas.
5. Se deben **incluir explícitamente tareas de prueba, documentación y automatización**.

## 3.5 Los 9 Patrones para Partir HU Grandes (Richard Lawrence) (Diapositivas 83–91)

Cuando una Historia de Usuario es demasiado grande (excede entre 1/10 y 1/6 de la velocidad del equipo o no cabe con holgura en una iteración), se divide en historias más pequeñas aplicando los **9 Patrones de Richard Lawrence**:

| Patrón de División | Criterio Metodológico | Ejemplo Oficial de Clase (Diapositivas 83–91) |
|---|---|---|
| **1. División por pasos del flujo de trabajo** (***Workflow Steps***) | Identificar los pasos secuenciales que realiza el usuario para hacer una actividad e implementarlos de forma incremental. | **HU:** *Publicar nuevos programas de precios.*<br>• Publicar precios en la página principal.<br>• Enviar mensaje a cada cliente en el portal.<br>• Enviar lista con descuentos personalizados. |
| **2. División por reglas de negocio** (***Business Rules***) | Si una HU incluye varias reglas o condiciones de negocio, crear historias separadas para cada regla o condición. | **HU:** *Retirar dinero de un cajero.*<br>• Impedir retiro si no hay saldo suficiente.<br>• Limitar retiro a un máximo de 1000 € por día por seguridad. |
| **3. El mayor esfuerzo de desarrollo** (***Major Effort***) | Partir la HU de modo que la primera parte asuma el mayor esfuerzo de infraestructura y las siguientes solo añadan variantes simples. | **HU:** *Cambiar tarifa de electricidad con selector.*<br>• Usar tarifa de medida de uso (infraestructura).<br>• Usar tarifa de prepago.<br>• Usar tarifas de descuento. |
| **4. Sencillo / Complejo** (***Simple / Complex***) | Parar y preguntarse: *"¿Qué es lo más simple que podría funcionar?"* Construir ese núcleo y añadir variaciones y complejidad en nuevas HU. | **HU:** *Tener siempre el precio más bajo.*<br>• Teniendo en cuenta el proveedor actual más barato.<br>• Avisando de ofertas que aparecerán en el futuro próximo. |
| **5. Variaciones en datos** (***Data Variations***) | Las variaciones sobre los tipos de datos, idiomas o formatos aumentan la complejidad; procesar primero un tipo de dato y luego el resto. | **HU:** *Enviar mensajes a los clientes.*<br>• Enviar mensajes en inglés.<br>• Enviar mensajes en español. |
| **6. Métodos de entrada de datos (Vista IU)** (***Interface Variations***) | Cuando la complejidad reside en la IU más que en la lógica, empezar con una interfaz básica y añadir después interacciones o dispositivos más ricos. | **HU:** *Escuchar mis listas de reproducción.*<br>• Escuchar en el navegador de escritorio trabajando.<br>• Escuchar en la aplicación móvil mientras corro. |
| **7. Niveles de calidad del sistema** (***Deferred Performance***) | Hacer que la implementación funcione primero y en HU posteriores optimizar eficiencia, rendimiento o accesibilidad. | **HU:** *Navegar por el catálogo de productos.*<br>• Cargar rápidamente el catálogo sin demoras.<br>• Permitir acceso mediante lectores de pantalla para problemas de visión. |

| Patrón de División | Criterio Metodológico | Ejemplo Oficial de Clase (Diapositivas 83–91) |
|---|---|---|
| **8. Operaciones múltiples** (*CRUD Operations*) | Verbos genéricos como *gestionar, administrar o controlar* se dividen en sus operaciones elementales individuales. | **HU:** *Gestionar mi cuenta.*<br>• Darme de alta como cliente.<br>• Editar parámetros de la cuenta.<br>• Cancelar la cuenta.<br>• Añadir más dispositivos asociados. |
| **9. Escenarios de un caso de uso** (*Use Case Scenarios*) | Si se parte de un estudio previo de casos de uso, dividir cada caso de uso en escenarios individuales independientes. | Separar el escenario principal de éxito de los escenarios alternativos (alta, edición, baja, vinculación). |

## 3.6 SPIKE: Investigación Técnica y Funcional (Diapositivas 92–96)

**Definición Canónica: SPIKE**

Un **SPIKE** es un **tipo especial de Historia de Usuario** utilizada para gestionar una dificultad o incertidumbre. Representa el tiempo acotado (*timebox*) necesario para **reducir un riesgo o una incertidumbre de un problema técnico o funcional**, o para **aumentar la seguridad sobre la estimación de una HU**.

### 3.6.1 Tipos de SPIKE y Causas que los Originan (Diapositivas 92–94)

1. **SPIKE Técnico:** Analiza varias aproximaciones técnicas en el dominio de la solución del problema:
   - Decidir si comprar o desarrollar un componente (*Buy vs. Build*).
   - Evaluar el rendimiento o viabilidad de una solución propuesta.
   - Evaluar el impacto de carga de una nueva HU.
   - *Ejemplo (Diapositiva 94):* Investigar cuánto tiempo se tarda en actualizar una pantalla personalizada con el consumo eléctrico actual y qué requisitos de comunicación y gestión tendrá.
2. **SPIKE Funcional:** Se emplea cuando existe una incertidumbre importante sobre **cómo debe interactuar el usuario con el sistema** para alcanzar sus objetivos:
   - *Ejemplo (Diapositiva 94):* Realizar un prototipo de histograma de gasto eléctrico en el portal web y obtener realimentación (*feedback*) de varios usuarios sobre la forma de presentar la información.

**Importante / Pregunta de Examen: Las 5 Reglas Oficiales sobre SPIKE (Diapositiva 95)**

**Regla 1: Gestión estándar en la pila:** Se tratan como cualquier otra HU (se incluyen en el *Product Backlog*, se priorizan, se estiman/acotan en tiempo, etc.).

**Regla 2: Su resultado NO es código de producción, es INFORMACIÓN útil**

**para otras HU:** Debe generar una decisión fundamentada, un prototipo descartable, una demostración de un diseño o una solución parcial de un problema.

**Regla 3:** **Demostrabilidad:** La solución o conclusión de un SPIKE debe ser **demostrable al equipo de desarrollo**.

**Regla 4:** **Aceptación formal:** Un SPIKE, como cualquier HU, debe ser **aceptado usando criterios de aceptación** definidos previamente en la propia tarjeta del SPIKE.

**Regla 5:** **Separación temporal de iteraciones:** Suele ser recomendable dejar la resolución del SPIKE para una **iteración anterior/diferente** a la de la HU que lo motivó (para poder estimar y construir la HU real una vez despejada la incertidumbre).

Bloque 4

# Algoritmo de División, Cuaderno de Ejercicios y Banco de Examen

**Síntesis del Bloque y Objetivos de Evaluación:**

**Alcance del Bloque:** Diapositivas 96–99 del Seminario 2 y preparación integral de la prueba parcial.
**Contenido Práctico:** Algoritmo en 3 fases de Richard Lawrence (*Prepara* → *Aplica Patrones* → *Evalúa*), resolución detallada de los ejercicios prácticos de las diapositivas oficiales (descomposición de *Gestionar Usuarios 13 PH*, repriorización vertical del buscador de libros y carrito, cálculo de velocidad y asignación de iteraciones), diseño de tarjetas completas, BDD, auditoría INVEST, formulación de Spikes y banco de **20 preguntas cortas de examen** con respuesta razonada.

## 4.1 Algoritmo en 3 Fases para Dividir una Historia (Diapositiva 96)

El póster metodológico de **Richard Lawrence (*Agile For All*)** incluido en la diapositiva 96 sintetiza el procedimiento sistemático en **tres etapas secuenciales** para dividir cualquier Historia de Usuario excesivamente grande:

**Fase 1: PREPARA LA HISTORIA (Filtro de Entrada):**

- *Pregunta 1.1: ¿Cumple la historia los requisitos INVEST (excepto, quizás, Pequeña)?*
  - **NO** ⟶ Combínala con otra historia o reformúlala para tener una buena historia de inicio (que aporte valor real y sea independiente) antes de intentar partirla.
  - **SÍ** ⟶ Pasa a la Pregunta 1.2.
- *Pregunta 1.2: ¿Es la historia de tamaño de* **1/10** *a* **1/6** *de la velocidad del equipo?*
  - **SÍ** ⟶ **¡Lista!** No necesita dividirse más.
  - **NO** ⟶ Continúa a la Fase 2: la historia es demasiado grande y **debe dividirse**.

**Fase 2: APLICA LOS PATRONES DE DIVISIÓN (Núcleo de Partición):**

- **Pasos de un Workflow:** ¿Describe un flujo completo? Haz una pequeña loncha de principio a fin, o bien el inicio y el final del flujo y luego las historias intermedias.

- **Operaciones múltiples:** ¿Incluye múltiples operaciones (p. ej. "gestionar" o "configurar")? Separa cada operación CRUD en su propia historia.
- **Variaciones en las reglas de negocio:** ¿Tiene variedad de reglas o términos como "fechas flexibles"? Implementa primero un subconjunto básico de reglas y mejora con reglas adicionales después.
- **Variaciones en datos:** ¿Hace lo mismo sobre diferentes tipos de datos? Procesa primero un tipo de dato y añade el resto después.
- **Variaciones en la interfaz:** ¿Tiene una interfaz compleja o múltiples canales? Empieza con una versión simple de la interfaz o un solo tipo de interfaz.
- **Esfuerzo importante (*Major Effort*):** Cuando la primera implementación lleva todo el peso de infraestructura, agrúpala y difiere la decisión de qué variante va primero.
- **Simple / Compleja:** ¿Tiene un núcleo simple que provee la mayoría del valor o aprendizaje? Extrae ese núcleo primero y añade el resto posteriormente.
- **Retrasa rendimiento (Calidad):** ¿Debe mucha de su complejidad a RNF (rendimiento, escalabilidad)? Hazla funcionar primero y mejórala después para satisfacer los requisitos no funcionales.
- **Último recurso – Construye un "SPIKE":** ¿Todavía estás confuso sobre cómo dividirla? Si encuentras una pequeña parte que entiendas bien, empieza por ahí; si no, **define las 2 ó 3 preguntas que más te inquietan, escribe un Spike para responderlas, construye lo mínimo para resolverlas y reinicia el proceso**.

**Fase 3: EVALÚA LA DIVISIÓN (Filtro de Calidad de Salida):**

- *¿Son las nuevas historias parecidas en tamaño y cada una representa entre* 1/10 *y* 1/6 *de la velocidad?* (Si no, prueba otro patrón en la historia original o en las subhistorias grandes).
- *¿Es cada historia un corte vertical que cumple INVEST?* (Si no, prueba otro patrón).
- *¿Hay historias resultantes de bajo valor que ahora puedas* ***repriorizar hacia abajo o borrar****?* (Si no, probablemente aún tienes desperdicio oculto).
- *¿Hay una historia obvia por la que empezar que proporcione* ***valor temprano, aprendizaje o mitigación del riesgo****?* Si la respuesta es **SÍ**, la división ha concluido con éxito.

## 4.2 Cuaderno de 8 Ejercicios Prácticos de Examen Resueltos

**Ejercicio Resuelto de Examen: 1. Descomposición de Backlog: «Gestionar Usuarios» (Diap. 97)**

**Enunciado:** En la Pila de Producto inicial figura una historia de usuario de tamaño **13** PH titulada *"Gestionar Usuarios"*, seguida de:

- *Generar factura en HTML, PDF o Excel.*
- *Como operador quiero saber qué usuarios están conectados en el sistema.*
- *Como administrador quiero poder tener 100 usuarios simultáneos.*
- *Administración manual del sistema de usuarios.*

Explica cómo se aplica la división de historias y la reordenación del backlog mostrada en las tres columnas de la diapositiva 97.

**Resolución Paso a Paso:**

1. **Detección de Épica / Historia demasiado grande:** La historia *"Gestionar Usuarios"* tiene un tamaño de **13** PH y utiliza un verbo genérico (*"Gestionar"*) que oculta múltiples operaciones independientes.
2. **Aplicación del Patrón 8 (Operaciones Múltiples / CRUD):** Se parte *"Gestionar Usuarios (13 PH)"* en 4 Historias de Usuario independientes y más pequeñas:
   - **Registrar un nuevo Usuario** $\longrightarrow$ Estimación: **3** PH.
   - **Editar un Usuario existente** $\longrightarrow$ Estimación: **5** PH.
   - **Buscar un Usuario** $\longrightarrow$ Estimación: **3** PH.
   - **Borrar un Usuario** $\longrightarrow$ Estimación: **5** PH.
3. **Repriorización de la Pila (Tercera columna de la diapositiva 97):** Una vez dividida la historia en piezas pequeñas, el Product Owner no está obligado a construir las 4 piezas seguidas antes de abordar otras historias valiosas. Así, puede priorizar arriba *Registrar Usuario (3 PH)* y *Editar Usuario (5 PH)*, intercalar a continuación *Generar factura en HTML, PDF o Excel* y *Saber qué usuarios están conectados*, posponer *Buscar un Usuario (3 PH)* más abajo y postergar *Borrar un Usuario (5 PH)* para entregas posteriores.

## Ejercicio Resuelto de Examen: 2. Partición Vertical: Librería Online (Diapositiva 98)

**Enunciado:** Analiza las dos tablas de priorización de una tienda de libros online (*Bookstore*) de la diapositiva 98. Explica qué patrón de división se ha aplicado sobre la característica de prioridad 1 y qué ventaja estratégica obtiene el negocio con la nueva ordenación.

| Tabla Inicial (Antes de dividir) | | Tabla Final (Dividida y repriorizada) | |
|---|---|---|---|
| **Pri.** | **Feature** | **Pri.** | **Feature** |
| 1 | Book search (title, author, subject) | 1 | **Book Search by Title** |
| 2 | Online shopping cart | 2 | Online shopping cart |
| 3 | Credit card processing integration | 3 | **Book Search by Author** |
| 4 | Book details | 4 | Credit card processing integration |
| 5 | Order tracking | 5 | **Book Search by Subject** |
| 6 | Book preview | 6 | Book details |
| | | 7 | Order tracking |
| | | 8 | Book preview |
| | | 9 | **Book Search by ISBN** |

**Resolución Paso a Paso:**

1. **Problema inicial:** La funcionalidad 1 (*Capacidad de búsqueda de libros por título, autor y materia*) es demasiado amplia. Construir todos los motores de búsqueda al principio retrasa la entrega del carrito de compra (*Online shopping cart*) y del pago con tarjeta (*Credit card processing*).

2. **Patrón aplicado:** Se aplica el **Patrón 4 (Sencillo / Complejo)** combinado con el **Patrón 5 (Variaciones en datos / parámetros de búsqueda)**, separando la búsqueda en 4 historias independientes:
   - *Búsqueda por Título* (Prioridad 1 – el caso más frecuente y sencillo).
   - *Búsqueda por Autor* (Prioridad 3).
   - *Búsqueda por Materia/Subject* (Prioridad 5).
   - *Búsqueda por ISBN* (Prioridad 9 – caso muy específico que se pospone al final).
3. **Ventaja de negocio (*Walking Skeleton*):** Con solo implementar la *Búsqueda por Título* (Prioridad 1) y el *Carrito de compra* (Prioridad 2), el cliente ya dispone de un flujo de compra funcional muy temprano, adelantando el retorno de inversión y posponiendo la búsqueda por ISBN al final de la pila.

**Ejercicio Resuelto de Examen: 3. Descomposición en Tareas Técnicas (Diaps. 81–82)**

**Enunciado:** Dadas las siguientes dos Historias de Usuario del seminario:

- **HU A:** *"Como administrador quiero realizar un mantenimiento de los clientes de la empresa para tenerlos actualizados."*
- **HU B:** *"Como cliente, quiero buscar un hotel para alojarme."*

Realiza la descomposición en tareas de desarrollo para cada una cumpliendo las reglas del seminario.

**Resolución (Desglose Oficial de las Diapositivas 81 y 82):**

**Tareas para HU A (Mantenimiento de clientes):**

1. Construir la Interfaz de Usuario (IU).
2. Construir la lógica de negocio.
3. Construir la capa de acceso a datos.
4. Implementar la operación *añadir usuario*.
5. Implementar la operación *modificar usuario*.
6. Implementar la operación *eliminar usuario*.
7. Realizar las pruebas de *añadir/modificar/eliminar usuario*.

**Tareas para HU B (Buscar un hotel):**

1. Codificar la pantalla de búsqueda básica.
2. Codificar la pantalla de búsqueda avanzada.
3. Codificar las pantallas de resultados.
4. Escribir las consultas SQL para las búsquedas.
5. Documentar la nueva funcionalidad en manuales y ayudas.
6. Realizar las pruebas de aceptación.

*Nota metodológica:* Cada tarea tiene una duración de entre **medio día y 3 ó 4 días de trabajo**, genera un entregable verificable e incluye explícitamente tareas de prueba y documentación.

### Ejercicio Resuelto de Examen: 4. Cálculo de Velocidad y Planificación Completa de Iteraciones

**Enunciado:** Un equipo ágil está formado por **6 desarrolladores**. Las iteraciones duran **2 semanas laborales** (5 días por semana). El equipo estima en Puntos de Historia sabiendo que 1 PH equivale a 1 día de trabajo ideal. Debido a reuniones de Scrum, mantenimiento y gestiones internas, el **factor de dedicación es del 75 %**.

**a)** Calcula razonadamente la **Velocidad del equipo** por iteración.

**b)** Si el Product Backlog tiene las siguientes historias priorizadas de mayor a menor prioridad: $H_1$(15 PH), $H_2$(12 PH), $H_3$(8 PH), $H_4$(15 PH), $H_5$(10 PH), $H_6$(12 PH) y $H_7$(5 PH), asigna las historias a las iteraciones necesarias sin sobrepasar nunca la velocidad del equipo.

**Resolución Paso a Paso:**

- **Apartado a) Cálculo de la Velocidad:**

$$\text{Días laborables por iteración} = 2 \text{ semanas} \times 5 \text{ días/semana} = 10 \text{ días}$$

$$\text{Capacidad bruta del equipo} = 6 \text{ desarrolladores} \times 10 \text{ días} = 60 \text{ días de trabajo}$$

$$\textbf{Velocidad} = 60 \text{ días} \times 0{,}75 = \textbf{45} \text{ PH por iteración}$$

- **Apartado b) Planificación de Iteraciones (límite máximo = 45 PH por iteración):**
  - **Iteración 1:** $H_1(15) + H_2(12) + H_3(8) + H_5(10) = \mathbf{45}$ PH (o bien respetando orden estricto: $H_1(15)+H_2(12)+H_3(8) = \mathbf{35}$ PH, ya que $35+H_4(15) = 50 > 45$; si $H_5(10)$ no depende de $H_4$, puede adelantarse para completar exactamente $35 + 10 = 45$ PH).
  - **Iteración 2 (orden estricto):** $H_4(15) + H_5(10) + H_6(12) + H_7(5) = \mathbf{42}$ PH $\leq$ 45 PH.
  - En tan solo **2 iteraciones** (35 PH + 42 PH = 77 PH totales) queda completado todo el bloque de historias.

### Ejercicio Resuelto de Examen: 5. Diseño Completo de Tarjeta de HU (Anverso + Reverso en BDD)

**Enunciado:** En el sistema de una clínica médica privada, un paciente registrado necesita poder cancelar una cita médica programada desde la aplicación web. Durante la conversación con el cliente se acuerda que: (1) si cancela con más de 24 horas de antelación no tiene coste; (2) si cancela con menos de 24 horas se le aplica una penalización de 15 €, salvo que tenga seguro "Oro"; y (3) el hueco libre debe volver a mostrarse disponible inmediatamente en la agenda del médico. Redacta la tarjeta completa (Anverso y Reverso con pruebas de aceptación tradicionales y en BDD).

**Resolución:**

**Tarjeta de Historia de Usuario: ID: HU-08 – Cancelación de Cita Médica Online**

**ANVERSO DE LA TARJETA:**

- **ID:** `HU-08` **TÍTULO:** `Cancelación de cita médica programada`
- **DESCRIPCIÓN: Como** paciente registrado de la clínica, **quiero** cancelar una cita médica previamente reservada desde mi panel web **para** liberar mi compromiso si no puedo asistir y permitir que otro paciente aproveche ese horario.
- **ESTIMACIÓN: 5** PH **PRIORIDAD: Alta (250) DEPENDENCIAS:** `HU-03 (Reserva de cita)`

**REVERSO DE LA TARJETA (PRUEBAS DE ACEPTACIÓN):**

1. Comprobar que si un paciente cancela una cita con **más de 24 horas** de antelación, la cita queda anulada con coste 0 € y recibe correo de confirmación.
2. Comprobar que si un paciente con seguro **Estándar** cancela con **menos de 24 horas**, se genera un cargo de penalización de 15 €.
3. Comprobar que si un paciente con seguro **Oro** cancela con **menos de 24 horas**, la cancelación es gratuita (0 €).
4. Verificar que el tramo horario cancelado pasa inmediatamente al estado *"Disponible"* en el calendario del facultativo.

**Especificación en formato BDD (Gherkin):**

```
DADO un paciente con seguro "Estandar" y una cita programada
   para dentro de 12 horas
CUANDO el paciente solicita cancelar dicha cita en el sistema
ENTONCES el sistema anula la cita, aplica un cargo de 15 euros,
    libera el hueco en la agenda del medico y envia un email
   justificante al paciente.
```

**Ejercicio Resuelto de Examen: 6. Auditoría Crítica de Historias Defectuosas mediante INVEST**

**Enunciado:** Evalúa por qué las siguientes tres propuestas NO son buenas Historias de Usuario indicando qué criterio(s) del modelo **I.N.V.E.S.T.** incumplen y cómo corregirlas:

**a)** *"Como desarrollador backend, quiero migrar las tablas de PostgreSQL a MongoDB usando índices compuestos B-Tree para mejorar la base de datos."*

**b)** *"Como cliente del portal de viajes, quiero un buscador de vuelos y hoteles que sea muy intuitivo, moderno, rápido y fácil de usar."*

**c)** *"Como gerente del banco, quiero automatizar toda la contabilidad, el análisis de riesgo hipotecario y la pasarela internacional SWIFT para modernizar el banco."*

**Resolución Razonada:**

- **Análisis de a):** Incumple **V (Valiosa)** y **N (Negociable)**. Está escrita desde la perspectiva técnica de un programador e impone una tecnología concreta (*MongoDB,*

*índices B-Tree*) que no aporta valor directo perceptible al cliente ni deja margen de negociación. **Corrección:** Transformarla en una tarea técnica interna o asociarla a un beneficio real del usuario (*"Como comprador, quiero que las búsquedas de catálogo respondan en menos de 2 segundos para no abandonar la compra"*).

- **Análisis de b):** Incumple **T (Testeable)** y **E (Estimable)**. Adjetivos subjetivos como *"muy intuitivo, moderno y fácil de usar"* son imposibles de verificar mediante una prueba de aceptación determinista de caja negra. **Corrección:** Definir criterios medibles (*"completar la reserva en un máximo de 3 clics y mostrar resultados en menos de 3 segundos"*).
- **Análisis de c):** Incumple **S (Small / Suscinta)** e **I (Independiente)**. No es una Historia de Usuario, sino un **Tema o conjunto de Épicas** que requeriría meses o años de trabajo. **Corrección:** Desglosarla jerárquicamente en Épicas y partirla en historias pequeñas (de 2–3 personas/semana) aplicando los patrones de Richard Lawrence.

## 4.3 Banco de 15 Preguntas Cortas de Examen con Respuesta Modelo

**P1. ¿Cuál es el objetivo principal de las Historias de Usuario según el seminario?**
*Respuesta:* Lograr la **interacción y comunicación verbal** con el equipo de desarrollo y con el usuario/cliente, situando esta colaboración muy por encima de documentar exhaustivamente los requisitos del sistema.

**P2. ¿Por qué se dice que las Historias de Usuario son "promesas de una conversación" y qué ocurre si incluimos demasiados detalles en la tarjeta?**
*Respuesta:* Porque la tarjeta solo contiene una declaración corta de intención cuyo propósito es provocar preguntas que se resolverán dialogando en la etapa de *Conversación*. Si una tarjeta incluye demasiados detalles, **limita la conversación con el cliente**, reduce la creatividad del equipo y la convierte en una especificación rígida.

**P3. ¿Qué diferencia fundamental existe entre Casos de Uso e Historias de Usuario respecto a la planificación y la duración?**
*Respuesta:* Los **Casos de Uso** (RUP) no se utilizan para planificar y su implementación puede extenderse durante varias iteraciones. Las **Historias de Usuario** (XP/Scrum) sí se utilizan directamente para planificar y estimar, y deben ser implementadas y probadas íntegramente dentro de **una única iteración**.

**P4. Explica qué elementos componen el modelo "CCC" de Ron Jeffries.**
*Respuesta:* **1) Card (Tarjeta):** las 2–3 frases que describen la intención de la HU en lenguaje de negocio; **2) Conversation (Conversación):** el diálogo entre equipo y cliente donde emergen los detalles; y **3) Confirmation (Confirmación):** las pruebas de aceptación que verifican que la HU ha sido correctamente implementada.

**P5. ¿Por qué se afirma que las Pruebas de Aceptación son pruebas de "Caja Negra" y quién es su propietario principal?**
*Respuesta:* Son de **caja negra** porque solo verifican que las salidas observables del sistema cumplen las condiciones de satisfacción pactadas ante ciertas entradas, sin preocuparse de cómo están programadas internamente. Aunque cualquiera puede escribirlas, su propietario principal es el **Product Owner**.

**P6. ¿Qué diferencia existe entre los Criterios de Aceptación y la Definición de**

**Terminado (*Definition of Done*)?**
*Respuesta:* Los **Criterios de Aceptación** son condiciones funcionales o de negocio *específicas de una Historia de Usuario concreta*. La **Definición de Terminado (DoD)** es un acuerdo formal de calidad *global/transversal para todas las HU* (pruebas unitarias e integración superadas, código revisado y documentado, entorno de despliegue, métricas de calidad y aprobación del PO).

**P7. ¿De qué cuatro formas pueden tratarse los Requisitos No Funcionales (RNF) en un proyecto ágil?**
*Respuesta:* 1) Como restricciones asignadas a cualquier elemento de las pilas (Temas, Épicas, HU o Tareas); 2) Como criterios de aceptación de una HU; 3) Como parte de la definición de terminada (*Done*); y 4) En un documento de especificación específico para RNF globales al sistema.

**P8. ¿Qué significan las siglas del modelo I.N.V.E.S.T. de Bill Wake?**
*Respuesta:* **I**ndependiente (*Independent*), **N**egociable (*Negotiable*), **V**aliosa (*Valuable*), **E**stimable (*Estimable*), **S**uscinta/Pequeña (*Small*: 2–3 personas/semana o 1/10 a 1/6 de la velocidad) y **T**esteable (*Testable*).

**P9. ¿Qué representa un Mapa de Historias de Usuario (*User Story Mapping*) y cuáles son sus dos ejes?**
*Respuesta:* Es una representación bidimensional del *Product Backlog*. Su **eje horizontal** representa el tiempo o secuencia de actividades del usuario (*Backbone*), y su **eje vertical** representa el grado de necesidad u opcionalidad (arriba las críticas para el *first release*, abajo las opcionales para entregas posteriores).

**P10. En relación con la priorización y estimación de una HU, ¿puede cambiar su prioridad a lo largo del proyecto? ¿Y su tamaño?**
*Respuesta:* **La prioridad SÍ puede cambiar** en cualquier momento según evolucionen las necesidades del negocio y del cliente; en cambio, **el tamaño NO cambia** por el hecho de que varíe su prioridad.

**P11. ¿Qué diferencia hay entre Puntos de Historia y Días Ideales?**
*Respuesta:* Los **Puntos de Historia (PH)** miden el tamaño relativo de una HU comparada con otras del mismo equipo combinando esfuerzo, complejidad y riesgo. Los **Días Ideales** miden el tiempo de trabajo puro asumiendo dedicación continua sin ninguna interrupción.

**P12. En *Planning Poker*, ¿por qué todos muestran sus cartas a la vez y qué se hace cuando hay disparidad en las votaciones?**
*Respuesta:* Se muestran a la vez para evitar el **sesgo de anclaje** (que la opinión de un miembro influya en los demás). Cuando hay disparidad, las personas con las estimaciones **más alta y más baja** exponen sus argumentos al equipo, se discuten los riesgos ocultos y se vuelve a votar hasta lograr consenso.

**P13. ¿Cuáles son las tres formas de obtener la Velocidad de un equipo de desarrollo?**
*Respuesta:* 1) Utilizar datos históricos de iteraciones pasadas (*"el tiempo que hizo ayer"*); 2) Ejecutar una iteración inicial de prueba y observar su velocidad; y 3) Hacer un pronóstico mediante el **cálculo de recursos** afectado por un factor de dedicación.

**P14. ¿Qué diferencia hay entre descomponer una HU en Tareas y partir una HU grande mediante los patrones de Richard Lawrence?**
*Respuesta:* Al **partir una HU grande** (con los 9 patrones) se obtienen *varias Historias de Usuario más pequeñas*, donde cada una sigue siendo un corte vertical que entrega valor al usuario. Al **descomponer en tareas**, una única HU se divide en actividades técnicas

horizontales (IU, BBDD, lógica, pruebas) de 0,5 a 3–4 días para organizar el trabajo diario de los programadores.

**P15. Define qué es un SPIKE, sus dos tipos y cuál es su entregable principal.**
*Respuesta:* Un **SPIKE** es un tipo especial de HU acotada en tiempo destinada a investigar una dificultad, reducir un riesgo/incertidumbre o ganar confianza para estimar otra HU. Puede ser **Técnico** (evaluar arquitectura, rendimiento, comprar vs. construir) o **Funcional** (prototipar interacción con el usuario). Su resultado **NO es código de producción**, sino **información útil** (una decisión, prototipo o demostración) aceptada mediante criterios de aceptación.

## 4.4 Bibliografía Oficial del Seminario (Diapositiva 99)


- **Dean Leffingwell, Pete Behrens (2009):** *"A User Story Primer"*.
- **Dean Leffingwell (2009):** *"A Lean and Scalable Requirements Information Model for the Agile Enterprise"*.
- **Mike Cohn (2004):** *"User Stories Applied for Agile Software Development"*, Addison-Wesley.
- **Mike Cohn (2005):** *"Agile Estimating and Planning"*, Prentice Hall.
- **Kent Beck, Martin Fowler (2000):** *"Planning Extreme Programming"*, Addison-Wesley.
- **Richard Lawrence (2009):** *"Patterns for Splitting User Stories"*, Agile For All.