Ingeniería de Software

Requisitos e historias de usuario

Entrevistas, criterios de aceptación y priorización

Gerson Garrido

Semana 3

UNIVERSIDAD NACIONAL DE INGENIERÍA

Hoy

1. Qué es un requisito: funcionales, no funcionales y restricciones
2. El problema detrás de la solución que trae el cliente
3. Elicitación: preparar, conducir y documentar una entrevista
4. Historias de usuario y criterios *dado, cuando, entonces*
5. INVEST, dividir historias y priorizar con MoSCoW
6. Con IA: encontrar huecos sin inventar requisitos

Lectura

Sesión A: Nielsen Norman Group, *User Interviews 101*, *Interviewing Users* y *Open-Ended vs. Closed Questions in User Research*. Sesión B: Mountain Goat Software, *User Stories*; Agile Alliance, *INVEST*; Martin Fowler, *GivenWhenThen*.

SECCIÓN 01

Qué es un requisito

01

De dónde salen los ítems del backlog

- La semana pasada: backlog, story points, sprint.
- Dimos por hecho que cada ítem dice lo que el cliente necesita.
- 737 MAX: el MCAS hizo lo especificado. Falló la especificación.

La idea de hoy

Un requisito tiene que poder **rastrearse** hasta algo que el cliente dijo o aprobó, y tiene que poder **comprobarse**.

Requisito

> **Requisito** ***(requirement)***
>
> Una condición que el sistema tiene que cumplir para que el cliente resuelva su problema o logre su objetivo, **y que está escrita**.

- **Condición**: se cumple o no. “Que funcione bien” no lo es.
- **Sistema**: visto desde fuera. “Usar SQLite” no es algo que el cliente necesite.
- **Problema del cliente**: si nadie sabe qué problema resuelve, sobra.

En Kallpa están en el README, *Reglas del negocio*: “Si el código hace otra cosa, el código está mal”.

IEEE 610.12 (1990), glosario de ingeniería de software.

Funcionales y no funcionales

Funcional

Qué hace el sistema.

- Se cancela hasta 3 horas antes.
- Una reserva tiene de 1 a 3 horas.
- Los socios pagan 10 % menos.

No funcional

Cómo de bien lo hace: una cualidad medible.

- La disponibilidad aparece en menos de 1 s en un celular con 4G.
- La página se usa entera solo con el teclado.
- Disponible de 7:00 a 22:00.

La frontera no siempre es nítida. Lo que importa: que esté escrito y se pueda comprobar.

Sin número todavía no es un requisito

Vago	Medible
Es rápida.	Desde que se elige la fecha hasta que aparecen las horas, menos de 1 s, con la red limitada a 4G en el navegador, en 9 de cada 10 intentos.
Es fácil.	Un vecino nuevo hace su primera reserva en menos de 2 minutos, sin ayuda (prueba con 5 vecinos).
Es accesible.	Todo se puede hacer solo con el teclado (WCAG 2.2, criterio 2.1.1).
Es confiable.	Ninguna reserva confirmada se pierde si el servidor se reinicia.

Cuatro cosas

Qué se mide, **dónde**, **cuánto** es aceptable y **cómo** se mide. “En promedio” esconde a quien espera 5 s.

No todo se mide con un número

Error común

Creer que todo requisito no funcional tiene un percentil.

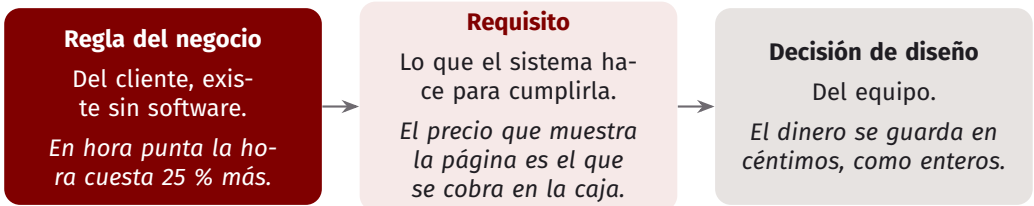
Restricción *(constraint)*

Una condición que se respeta o no, sin grados.

- Los datos personales de los vecinos se tratan según la Ley N.° 29733.
- La aplicación corre en la computadora de la oficina, con Windows 10.

La regla del número vale para las **cualidades** (velocidad, facilidad, disponibilidad).

Tres cosas que salen mezcladas



Una decisión escrita como requisito ya no se puede cambiar; una regla tomada como decisión del equipo se “optimiza”.

SECCIÓN 02

El problema y la solución

02

“¿Pueden poner un botón para bloquear la cancha?”

—¿Para qué la bloquearía? ¿Qué pasó la última vez?

—El martes vino el técnico a cambiar los reflectores de la cancha de vóley. Llegó a las cuatro y Las Matadoras ya tenían reservado de cuatro a seis. Tuve que llamarlas para que no vinieran y se molestaron, con razón.

—¿Cada cuánto viene el técnico?

—Más o menos una vez al mes. Avisa con dos o tres días.

El problema

El mantenimiento choca con reservas ya hechas y los equipos se enteran tarde.

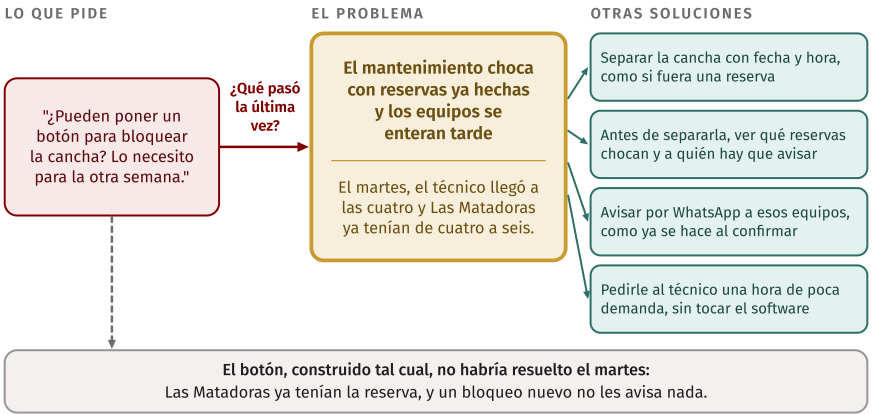
Con el problema aparecen otras soluciones

- Separar la cancha **con fecha y hora**, como una reserva.
- Ver **qué reservas chocan** antes de separarla.
- Avisar por WhatsApp a esos equipos.
- O pedirle al técnico que venga en una hora de poca demanda, y no tocar el software.

El botón, construido tal cual, no habría resuelto el martes: Las Matadoras ya tenían la reserva y un bloqueo nuevo no les avisa nada.

> **La pregunta que más ayuda**
> “¿Qué pasó la última vez?”

Del pedido al problema



Por qué el cliente no lo dice de entrada

- Es experto en su trabajo, no en software: pide lo que imagina posible.
- Lo obvio no se dice.
- **Lo que dice no es lo que hace**: recuerda mal y predice peor.
- *Query effect*: la gente fabrica una opinión si se le pregunta.

Una anécdota

Antes de Office 2007, Microsoft pidió ideas a sus clientes. La mayoría de los “pedidos nuevos” ya existían en Office 2003: el problema era encontrarlos.

Lo que pide el cliente es un dato, no la conclusión.

Jakob Nielsen, *Interviewing Users*, Nielsen Norman Group.

SECCIÓN 03

Elicitación

03

Cada técnica ve cosas distintas

Técnica	Encuentra	Se le escapa
Entrevista	Objetivos, problemas, reglas, vocabulario	Lo que hace sin darse cuenta
Observación	Lo que de verdad pasa: la caja, el cuaderno, los atajos	Por qué lo hace
Documentos	Reglamento, acuerdos de la junta	Lo que nadie escribió
Prototipo	Si la solución tiene sentido	El problema de fondo
Encuesta	Cuántos piensan algo	Lo inesperado

En una entrevista la gente cuenta lo que cree que hace; observando se ve lo que hace.

A quién preguntarle: los stakeholders de Kallpa



Los intereses chocan: el vecino quiere cancha libre en hora punta, el equipo quiere su hora asegurada, la junta quiere la hora punta alquilada.

¿A cuántas personas?

- Griffin y Hauser (1993): entrevistas a clientes de una categoría de productos de consumo.
- Estimaron que **20 a 30 entrevistas** encontraban el **90 a 95 %** de las necesidades.
- Límites: una categoría de productos, muchos clientes parecidos, una cifra que sale de un modelo.

Lo que sí se lleva

Cada entrevista nueva encuentra menos cosas nuevas, pero la primera sola encuentra bastante menos que todas juntas.

A. Griffin y J. R. Hauser, *The Voice of the Customer*, Marketing Science, 1993.

En la industria

> **GOV.UK Service Manual**
>
> Las opiniones o sugerencias que no vienen de los usuarios se tratan como **supuestos que hay que comprobar** con investigación.

- Entender a todos los usuarios, no solo a los “típicos”.
- Incluir a quienes prestan el servicio.
- Una necesidad se enfoca en el problema: “necesito un recordatorio”, no “necesito un correo”.

“Los vecinos quieren pagar por Yape” es un supuesto hasta que alguien lo averigüe con vecinos.

GOV.UK Service Manual, *Start by learning user needs.*

SECCIÓN 04

La entrevista

04

Antes: un objetivo

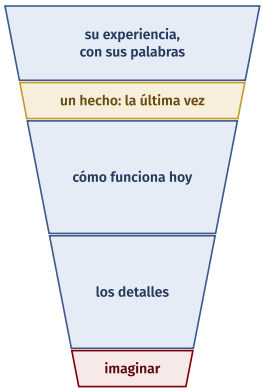
- Mal: “conocer a Doris”.
- Mal: “ver qué funciones quiere en la página”.
- **Bien**: “entender cómo se organiza hoy el mantenimiento de las canchas, cada cuánto choca con reservas y qué se hace cuando pasa”.

El objetivo no menciona la solución

Si dijera “validar el botón de bloqueo”, nadie preguntaría por el técnico.

Nielsen Norman Group, *User Interviews 101*: una entrevista es un estudio, no una venta.

La guía en embudo



PREGUNTAS DE LA GUÍA

1. Cuénteme cómo es una semana normal en la administración del complejo.
2. ¿Qué trabajos de mantenimiento se hacen en las canchas?
3. **¿Cómo fue la última vez que vino el técnico? ¿Qué pasó, paso a paso?**
4. ¿Cómo se entera usted de que va a venir? ¿Con cuánta anticipación?
5. Cuando el trabajo choca con una reserva, ¿qué hace hoy?
6. ¿Cómo avisa a los equipos? ¿Qué le dicen?
7. ¿Cuántas veces pasó en los últimos meses? ¿Lo anota en algún lado?
8. ¿Hay trabajos que cierran la cancha un día entero?
9. ¿Quién más está involucrado: la junta, la caja, el técnico?
10. **Si pudiera cambiar una sola cosa de cómo funciona esto hoy, ¿cuál sería?**

No menciona botones, pantallas ni la página.

Los roles

Pregunta

Conduce, sigue la guía sin leer-la, repregunta.

Anotan (dos)

Con las palabras del cliente, sin traducir.

Controla el tiempo

Avisa cuando quedan 10 minutos.

Error común

Ir sin guía “para que fluya”. Fluye hacia lo que el cliente quiere contar. La guía no es un guion: se sale de ella y se vuelve.

Abiertas, después cerradas

Pregunta	Respuesta	Qué se aprendió
¿Le sirve poder bloquear la cancha?	“Sí, claro.”	Nada.
¿El técnico viene en la tarde?	“A veces.”	Casi nada.
Cuénteme cómo fue la última vez que vino el técnico.	“El martes llegó a las cuatro, las de vóley ya estaban…”	El choque, el aviso tardío, el costo, que llama por teléfono.

Cuándo cada una

Las abiertas traen lo inesperado; las cerradas aclaran un dato que ya salió.

Nielsen Norman Group, *Open-Ended vs. Closed Questions in User Research*.

“¿Cómo fue la última vez que…?”

Pregunta general

“¿Cómo manejan los choques con el mantenimiento?”

→ “Llamamos a los equipos con anticipación.” *(la versión ideal)*

Caso concreto

“¿Cómo fue la última vez?”

→ “Me enteré el mismo día y las llamé a las tres.” *(lo que pasó)*

Después: “¿y la vez anterior?”, “¿siempre es así?”. Así se separa lo que pasa siempre de lo que pasó una vez.

Preguntas que inducen la respuesta

Inductora	Mejor
¿No sería mejor que pudieran pagar por Yape?	¿Cómo pagan hoy los vecinos? ¿Qué problemas hay con eso?
El cuaderno debe ser muy incómodo, ¿no?	¿Cómo usa el cuaderno? Muéstreme la última página.
¿Le molesta cuando el técnico llega sin avisar?	¿Cómo se entera de que va a venir el técnico?
¿Quiere que avisemos por WhatsApp a los equipos?	Cuando cancela una hora, ¿cómo se enteran los equipos?

Regla práctica

Si se puede responder “sí, claro” y esa respuesta haría feliz al equipo, está mal formulada.

Repreguntar, repetir, anotar lo que no sabe

- **Repreguntar**: “cuénteme más”, “¿por qué eso es importante?”, “¿qué pasó después?”.
- **Repetir lo entendido**: “A ver si entendí: el técnico avisa con dos o tres días, usted revisa la página y llama a quien reservó. ¿Es así?”
- **Lo que no sabe**: se anota qué y a quién preguntarle. Es la agenda de la próxima entrevista.

Error común

Hablar más que el cliente. Un silencio de cinco segundos incomoda, y casi siempre el cliente lo llena con lo que no había dicho.

Después: las notas, el mismo día

Lo que dijo

“Llegó a las cuatro y Las Matadoras ya tenían reservado de cuatro a seis.”

Viene “más o menos una vez al mes”.

“A veces no tengo el número.”

Lo que suponemos

Que un mantenimiento dura unas dos horas (un solo caso).

Que los equipos preferirían WhatsApp.

Lo que falta preguntar

¿Cuántos choques en tres meses? (el cuaderno)

¿Quién decide cuándo viene el técnico? (la junta)

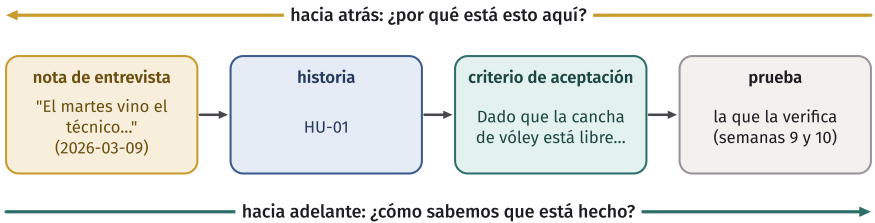
Un mes después alguien pregunta “¿de dónde sacamos que dura dos horas?”. Las notas tienen que poder responder.

El glosario

Término	Qué quiere decir para el complejo
Mantenimiento	Trabajo del técnico durante el cual no se puede jugar. Dura unas horas.
Separar la cancha	Que nadie pueda reservarla en un horario. Doris dice “bloquear”.
Socio	Vecino que paga la cuota mensual; 10 % de descuento.
La caja	El lugar y la persona que cobra, y el dinero del día.

- Las palabras del cliente no significan lo que el equipo cree.
- Si el equipo usa otras palabras, el cliente no reconoce su problema ni puede validar.

Rastrear cada requisito



- Hacia atrás: ¿por qué está esto aquí? Sin nota, lo inventó alguien.
- Hacia adelante: ¿cómo sabemos que está hecho? Un criterio sin prueba es una promesa.

Con IA: preparar y ordenar

Puede

- Proponer preguntas que no se les ocurrieron.
- Marcar preguntas inductoras en la guía.
- Resumir una transcripción, si el cliente aceptó.

No puede

- Saber qué quiere *este* cliente.
- Si se le piden “los requisitos”, devuelve una lista plausible y segura que nadie del complejo dijo.

Cómo se comprueba

Todo lo que entra a las historias se rastrea a una nota. Un resumen se coteja con la transcripción: puede omitir un “a veces” o volver dato un “no sé”.

SECCIÓN 05

Historias de usuario

05

Una promesa de conversación

Historia de usuario *(user story)*

Como **<rol>**, quiero **<acción>** para **<beneficio>**.

Como vecino que reserva, quiero ver cuánto voy a pagar antes de confirmar para no llevarme sorpresas en la caja.

- **Rol**: quién gana. Nunca “el usuario”.
- **Acción**: qué quiere hacer, sin decir cómo se construye.
- **Beneficio**: para qué. Si cuesta escribirlo, ¿hace falta la función?

Kent Beck introdujo el término en XP. La oración es un recordatorio: el requisito sale de la conversación.

Las tres C



- La tarjeta identifica y sirve para planificar: estimación y prioridad. Hoy, un issue.
- La confirmación: ejemplos, mejor si se pueden ejecutar.

Ron Jeffries, *Essential XP: Card, Conversation, Confirmation*, 2001.

Historias buenas y malas

Historia	Problema
Como usuario, quiero un botón para bloquear la cancha.	Rol genérico, una solución, sin beneficio.
Como administradora, quiero un CRUD de mantenimientos.	Implementación; no dice para qué.
Como administradora, quiero bloquear la cancha para bloquearla.	El beneficio repite la acción.
Como administradora del complejo, quiero separar una cancha para mantenimiento en un horario para que ningún vecino la reserve mientras trabaja el técnico.	**Bien**: sale de la nota del martes.

Error común: una historia por pantalla, por tabla o por endpoint.

No todo trabajo es una historia

- Actualizar una dependencia, configurar el pipeline.
- Investigar si `node:sqlite` aguanta dos procesos a la vez.

Spike

Una investigación con tiempo limitado. No entrega una función: entrega lo que el equipo necesita saber para estimar.

“Como developer, quiero actualizar Node para…” no aporta nada. Va al backlog como lo que es, y se explica qué riesgo reduce.

SECCIÓN 06

Criterios de aceptación

06

Dado, cuando, entonces

Criterio de aceptación *(acceptance criterion)*

Una condición concreta que la historia tiene que cumplir, comprobable con un sí o un no.

- **Dado**: el estado antes de la acción.
- **Cuando**: la acción. Una sola.
- **Entonces**: el resultado observable.

Ejemplo

Dado que hoy es lunes 9 de marzo a las 10:00, **cuando** Rosa cancela su reserva del martes 10 a las 9:00, **entonces** la respuesta es `200` y la hora vuelve a estar libre.

Terhorst-North y Matts, en BDD. Es la estructura de una prueba: *Arrange, Act, Assert* (Martin Fowler, *GivenWhenThen*, 2013).

Verificable: dos personas harían la misma prueba

Entonces…	**¿La misma prueba?**
la reserva es fácil de hacer.	No. ¿Fácil para quién?
el sistema responde bien.	No. ¿Qué es “bien”?
se muestra un error.	Casi. ¿Cuál? ¿Con qué código?
la respuesta es `422` con `TOO_LATE_TO_CANCEL` y la reserva sigue confirmada.	Sí.

Datos concretos: Rosa (1, regular), Luis (2, socio), cancha 1 de fulbito, cancha 3 de vóley, reloj fijo en el lunes 9 de marzo de 2026 a las 10:00.

El caso normal, el error y el límite

1. **Dado** que hoy es lunes 9 a las 10:00, **cuando** Rosa cancela su reserva del martes 10 a las 9:00, **entonces** `200` y la hora vuelve a estar libre.
2. **Dado** lo mismo, **cuando** cancela su reserva de hoy a las 12:00, **entonces** `422 TOO_LATE_TO_CANCEL` y sigue confirmada.
3. ¿Y si faltan exactamente 3 horas? “Hasta 3 horas antes”: ¿incluye el borde?

Los errores viven en los bordes

Justo a las 3 horas, justo a las 18:00, justo 14 días después. Si el criterio no dice qué pasa en el borde, cada developer decide por su cuenta. La decisión es del cliente.

Comportamiento, no implementación

Mal

Dado un mantenimiento nuevo, cuando se guarda, entonces se agrega una fila a la tabla `blocks` con la columna `reason`.

Bien

Dado ese mantenimiento, cuando Rosa intenta reservar la cancha de vóley el martes 10 a las 17:00, entonces la respuesta es `409` con el código `SLOT_TAKEN`.

- Doris no sabe qué es una columna; el segundo lo puede validar ella.
- Un código de error de la API sí es comportamiento: es el contrato con la página.

El criterio, ejecutado

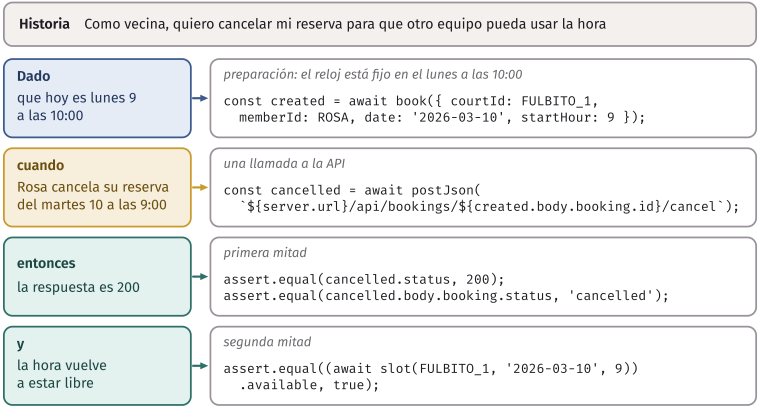
criterios.test.ts

```ts
test('dado que hoy es lunes 9 a las 10:00, cuando Rosa cancela su reserva del martes 10 a las 9:00, entonces la respuesta es 200 y la hora vuelve a estar libre', async () => {
  const created = await book({ courtId: FULBITO_1, memberId: ROSA, date: '2026-03-10', startHour: 9 });
  assert.equal(created.status, 200);
  assert.equal((await slot(FULBITO_1, '2026-03-10', 9)).available, false);

  const cancelled = await postJson(`${server.url}/api/bookings/${created.body.booking.id}/cancel`);
  assert.equal(cancelled.status, 200);
  assert.equal(cancelled.body.booking.status, 'cancelled');
  assert.equal((await slot(FULBITO_1, '2026-03-10', 9)).available, true);
});
```

Dado → preparar **cuando** → una llamada a la API **entonces** → las verificaciones, las dos mitades.

Cada parte del criterio, en el código



Si la prueba olvidara la segunda mitad, una implementación que responde 200 sin cancelar nada pasaría.

Doce criterios, doce pruebas

node —test criterios.test.ts

```
$ node --test criterios.test.ts
...
✓ Como vecino, quiero saber cuánto voy a pagar para no llevarme sorpresas en la caja (6.972125ms)
ℹ tests 12
ℹ suites 3
ℹ pass 12
ℹ fail 0
ℹ cancelled 0
ℹ skipped 0
ℹ todo 0
ℹ duration_ms 171.108208
```

- Pasan porque Kallpa *ya* cumple esas reglas. Con una historia nueva, los criterios van antes del código y fallan hasta que alguien la implementa.
- La historia del precio pide verlo antes de confirmar; Kallpa lo muestra después. Un criterio verificable puede no cumplirse.

Requisitos no funcionales como criterios

Como criterio

Dado un celular con la red limitada a 4G en el navegador, **cuando** se abre la disponibilidad de la cancha 1, **entonces** las horas aparecen en menos de 1 s.

O en la Definition of Done

Si vale para todas las historias: “toda pantalla nueva se puede usar solo con el teclado”.

Un revisor automático

revisar-historias.ts

```
$ node revisar-historias.ts historias-con-problemas.md
HU-01: el rol es genérico: «usuario»
HU-01 · criterio 1: palabra vaga «rápido»: ¿cómo se comprueba?
HU-01 · criterio 1: palabra vaga «bien»: ¿cómo se comprueba?
HU-01 · criterio 1: no tiene datos concretos: ni una hora, ni una fecha, ni un monto
HU-01 · criterio 2: habla de la implementación («tabla»), no del comportamiento
HU-01 · criterio 2: habla de la implementación («columna»), no del comportamiento
HU-01 · criterio 2: no tiene datos concretos: ni una hora, ni una fecha, ni un monto
HU-02: no tiene criterios de aceptación
HU-03 · criterio 1: le falta el "entonces"
3 historias, 3 criterios, 9 observaciones
$ node revisar-historias.ts historias-ejemplo.md
3 historias, 8 criterios, 0 observaciones
```

Lo que ningún revisor ve

- “Quiero bloquear la cancha para bloquearla” pasa **sin observaciones** (hay una prueba que lo comprueba).
- Tampoco ve si la historia resuelve el problema de Doris, si el monto es el correcto o si falta un caso.

Cero observaciones no es una buena historia

Esas preguntas solo las responde alguien que estuvo en la entrevista. La revisión automática de P1 “no juzga si las historias son buenas: eso lo revisan el docente y el cliente”.

SECCIÓN 07

INVEST

07

INVEST

I	N	V	E	S	T
Independiente en cualquier orden	**Negociable** no es un contrato	**Valiosa** para alguien; vertical	**Estimable** se puede estimar	**Pequeña** cabe en un sprint	**Testeable** en principio

Bill Wake (2003); Mike Cohn, *User Stories Applied* (2004). Es una lista de revisión, no una ley: dividir para que sea pequeña crea dependencias.

Agile Alliance, *INVEST*.

INVEST sobre Kallpa

Historia	Falla	Por qué
Ver con qué reservas choca un mantenimiento	I	Depende de HU-01: se ordena primero.
Pagar por Yape al reservar	E	Nadie sabe cómo se conectaría: primero un spike.
Que la página sea más moderna	V, T	¿Para qué? ¿Cuándo está hecho?
Manejar los torneos del barrio	S, E	Es una épica: hay que dividirla.
Separar una cancha con el ID de la tabla `courts`	N	Fija un detalle de diseño.

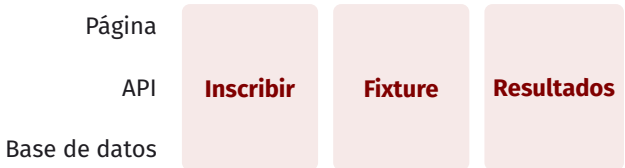
Lo útil es la conversación: “esta no la podemos estimar, ¿qué nos falta saber?”.

SECCIÓN 08

Dividir historias

08

Cortes verticales, no capas



Cada corte atraviesa todo y se puede usar

Una historia que no cabe en un sprint es una cascada dentro de un ítem del backlog. De 13 puntos o más, se divide.

SPIDR (Mike Cohn), sobre una épica

Como organizador de un campeonato del barrio, quiero inscribir equipos y programar los partidos para no armar el fixture a mano en un cuaderno.

Spike	Averiguar en un día cómo se genera un fixture todos contra todos para 8 equipos.
Path	Primero el camino feliz; después el equipo que se retira o el partido reprogramado.
Interface	Primero una lista desde la oficina; después una página para el celular.
Data	Primero solo fulbito, todos contra todos; después vóley, básquet y eliminatorias.
Rules	Primero sin cobro de inscripción; después con cobro y descuento de socio.

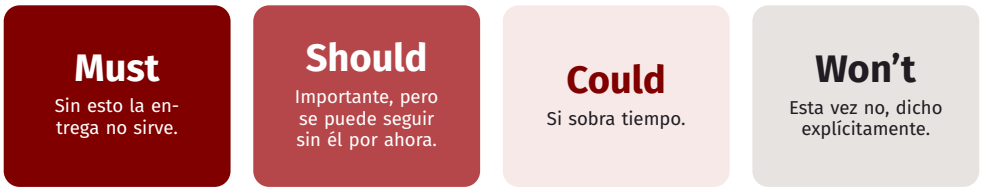
Se divide lo necesario para que quepa y para aprender pronto, no hasta llegar a un punto.

SECCIÓN 09

Priorizar

09

MoSCoW



Dai Clegg, 1994 (RAD; después DSDM). Siempre **respecto de una entrega**. Límites: no ordena dentro de una categoría, no justifica, y el Won't es ambiguo (¿esta vez o nunca?).

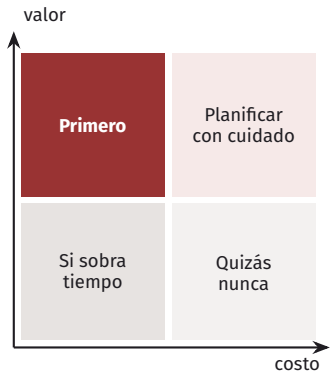
La primera entrega del mantenimiento (una semana)

	Historia	Por qué
Must	Separar una cancha para mantenimiento	Es el problema del martes.
Should	Ver con qué reservas choca	Doris puede revisar a mano, como hoy.
Could	Avisar por WhatsApp a los afectados	Doris ya llama; hoy Kallpa escribe el aviso en la consola.
Won't	Mantenimientos que se repiten cada mes	El técnico no tiene fecha fija. Se revisa si cambia.

Error común: todo es Must

"Si mañana solo pudiéramos entregar esto, ¿el cliente lo usaría?". Lo que no pasa esa pregunta no es Must.

Valor, costo y riesgo



- **Valor**: lo dice el cliente.
- **Costo**: lo dice el equipo.
- **Riesgo**: lo incierto, temprano. Es donde más se aprende.

Si el WhatsApp de verdad es esencial, un spike en el primer sprint.

El product owner ordena el backlog **con el cliente**. Uno priorizado solo por el equipo es una hipótesis.

SECCIÓN 10

Con IA

10

Una herramienta revisa HU-01

Una respuesta como las que suelen dar estas herramientas, **escrita para la clase**:

Sugerencia	Qué es
¿Qué pasa si choca con una reserva confirmada?	Ya respondida: HU-02.
¿Puede durar más de 3 horas, el máximo de una reserva?	Buena pregunta para Doris.
Los afectados reciben un reembolso del 100 % automático.	Regla inventada.
Se programa con hasta 60 días de anticipación.	Número inventado.
¿Y si cae en un feriado?	Buena pregunta para Doris.
Un calendario mensual para el técnico.	Historia nueva sin fuente.

Una pregunta para el cliente, no un requisito

¿A qué nota se puede rastrear? Si a ninguna, no entra.

El registro de uso de IA

En P1, `docs/registro-ia.md`: una fila por cada uso que influyó en la entrega.

Fecha	Qué se pidió	Qué se aceptó	Qué se rechazó y por qué
2026-03-10	Revisar HU-01 y proponer casos que falten	Preguntas 2 y 5, para la siguiente entrevista	Reembolso automático y 60 días: nadie del complejo los dijo

(Más la columna *Herramienta*.) Si una historia salió de una herramienta, que se note en el registro y que el cliente la haya confirmado.

SECCIÓN 11

Laboratorio

11

Entrevista simulada: los turnos fijos

> **El pedido de Doris**
>
> “Los equipos de siempre quieren tener su hora fija todas las semanas, como la liga de vóley de los jueves. ¿Se puede hacer en la página?”

1. Preparar: objetivo y guía de 10 a 12 preguntas abiertas; roles.
2. Entrevistar: una sola entrevista, por turnos; todos anotan todo.
3. Ordenar las notas: dijo, suponemos, falta preguntar; glosario.
4. Escribir de 6 a 10 historias con criterios; un RNF con número.
5. Revisar con INVEST y dividir con SPIDR.
6. Priorizar con MoSCoW y dibujar el prototipo en papel.
7. Validar con Doris.
8. Crear los issues con la etiqueta `turnos-fijos`. **No se implementan.**

Recomendaciones

- Al menos tres preguntas “¿cómo fue la última vez que…?”.
- Doris responde “sí, claro” a las preguntas inductoras. Como un cliente real.
- Escuchen las respuestas a los otros grupos: la mejor pregunta suele ser una repregunta.
- En el glosario: qué es para ella un *turno fijo* y una *temporada*.
- ¿Cómo encaja lo que dijo con las reglas que Kallpa ya tiene?
- Un Won’t bien justificado vale más que un Must que no cabe.

Semana 4: el design doc de esta funcionalidad. Semana 5: ¿un turno fijo es un concepto nuevo o una forma de usar la reserva?

Para la próxima clase

- **P0**: el pull request de su issue, revisado y con `make check ISSUE=NN` en verde.
- **P1** publicado: propongan su cliente al docente esta semana.
- Semana 4: Meyer, *Applying "Design by Contract"* (1992); Alexis King, *Parse, don't validate* (2019); TypeScript Handbook, *Narrowing*.

Pregunta

Una de sus historias pasa el revisor sin observaciones y Doris la aprueba. ¿Qué podría estar mal todavía?