Cartas de Slicing

Las cartas de Slicing te ayudarán a encontrar nuevos criterios para **subdividir** las **Historias de Usuario (HU)** grandes o épicas y complejas en otras más pequeñas y simples.

*Conoce más de las Cartas de Slicing escaneando este QR o entrando a **kl.la/slicing-kards** y cuéntanos cómo las usaste con **#SlicingKards** en las redes sociales.*

El alcance del juego es dividir historias de usuario o épicas en historias más pequeñas en el contexto del desarrollo de software, aplicado a los Ítems de un Backlog de Producto (PBI), aunque **se puede extrapolar** a otros contextos.

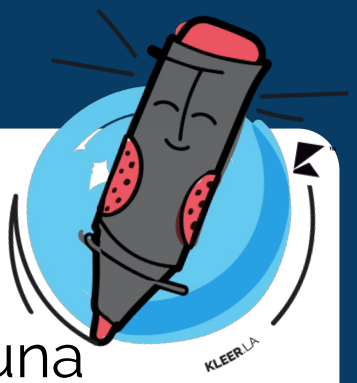
Inspirada por How to Split a User Story de R. Lawrence y Fifty Quick Ideas to Improve your User Stories por G. Adzic y D. Evans.

Puedes utilizarlas en sesiones de **Refinamiento** y **Planning** para trabajar de manera **colaborativa** con el equipo de desarrollo y el Product Owner.

kleer™

kl.la/slicing-kards
v6.0

¿Qué es Slicing?



Slicing o **splitting** (dividir, rebanar) es una práctica que permite reducir la complejidad y esfuerzo requerido de las Historias de Usuario, y que se aplica a todos los ítems sobre los cuales trabaja el equipo. Cuánto más pequeñas sean, mayores serán las chances de completarlas dentro de un Sprint y obtener feedback frecuente y temprano.

Este juego te ayudará a tener un Product Backlog más refinado y manejable para entregar valor de manera iterativa e incremental.

Un proceso **iterativo e incremental** es aquel que progresa a través del refinamiento sucesivo. **Iteramos** para encontrar o mejorar el valor entregado. Lo hacemos de forma **incremental** para construir gradualmente y así obtener el valor de negocio buscado.

Es un **compromiso de todo el equipo** que se puede mejorar con el tiempo.

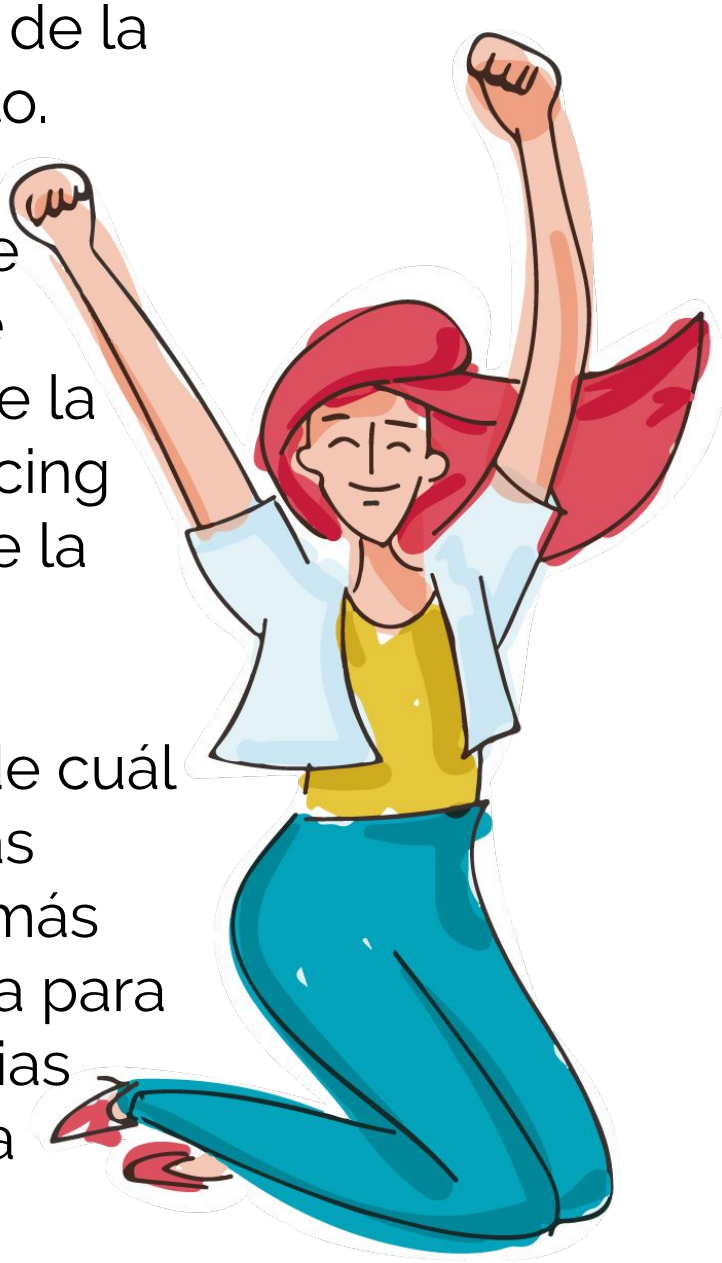
¿Cómo jugar?

Dada una historia de usuario aparentemente muy grande o compleja:

1. Repartimos todas las cartas entre los participantes de la reunión de refinamiento.

2. Cada participante elige entre sus cartas la que considera que contiene la mejor estrategia de slicing posible y la pone sobre la mesa.

3. Por consenso se decide cuál de todas las estrategias sobre la mesa será la más conveniente y se aplica para crear las nuevas historias derivadas de la historia original.



¡A jugar colaborativamente con todo el equipo!

1

Proceso o *workflow*

Cuando la historia de usuario que necesitamos dividir está relacionada con un proceso o workflow que tiene varios pasos, podemos intentar hacer un corte que incluya el primer y último paso (los de más valor) y luego ir agregando los pasos intermedios.

Si eso no es posible, buscar hacer un fino corte transversal de todo el proceso priorizando por valor (ver también carta 6 y 18).

2

Poner el foco en aprender

Si el alcance de la historia no está bien entendido por el equipo, o consideran que les falta conocimiento técnico se puede comenzar con una fase exploratoria tipo “spike” con:

- objetivo de aprendizaje
- entregable concreto
- tiempo fijo que vamos a dedicarle a la exploración o investigación

3

Diferir la Arquitectura

Si lo que queremos es entregar un producto funcionando rápidamente para recibir feedback temprano podemos:

- realizar un *working prototype* con total foco en la funcionalidad

- diferir los trabajos de arquitectura, backend y diseño de la experiencia de usuario

4

Sólo las operaciones necesarias

Si una historia incluye varias operaciones tales como Altas, Bajas y Modificaciones, comenzamos con lo estrictamente necesario.

Dejamos para más adelante a las operaciones menos utilizadas, las cuales quedarán temporariamente soportadas por procedimientos manuales.

5

Diferir la performance

Si los criterios de performance agregan mucha complejidad, crear dos historias que incluirán:

- el desarrollo de la funcionalidad
- el cumplimiento de los requerimientos de performance

6

¡Pareticemos!

Si la historia de usuario es muy grande o compleja podemos recurrir a aplicar el Principio de Pareto:

- enfocarse primero en el 20% del trabajo que aportará el 80% del valor
- dejar el resto para futuras iteraciones luego de recibir feedback

7

Tener en cuenta a un usuario por vez

Si una historia incluye funcionalidad para diversos usuarios finales o Personas entonces:

- dividir la historia creando una por cada rol o Persona y

- comenzar por aquellas que aportan más valor de negocio o que tengan mayor riesgo y necesitemos validar tempranamente

8

Diferir la Interfaz de Usuario (UI) para después

Evaluar si tiene sentido dividir la historia de usuario en:

- una historia con la funcionalidad pedida pero con una interfaz de usuario simplificada

- otra historia para completar o refinar la UI/UX

9

Por regla de negocio

Si las reglas de negocio a implementar son muy complejas o distintas:

- priorizarlas por valor de negocio, creando una historia de usuario para cada una de ellas

10

Una interfaz por vez

Si se requieren datos similares en diferentes formatos y provenientes de diferentes fuentes de información:

- trabajar con una interfaz o fuente de información cada vez

11

Validar antes de construir o desarrollar

Si la funcionalidad tiene mucho riesgo o incertidumbre (valor para el cliente, usabilidad, factibilidad técnica y valor de negocio) o está basada en una corazonada y no en datos, se transforma en una hipótesis a testear. Diferir el desarrollo y adelantar experimentos de Product Discovery:

- entrevistas con clientes/Persona
- experimentos cuantitativos
- prototipos para testear o validar

12

División por datos

Considere dividir la historia de usuario de acuerdo al significado de los datos, agregando inicialmente sólo la funcionalidad necesaria para los datos seleccionados.

Ej. procesar primero sólo la información de la Ciudad X (con mayores ventas) con sus impuestos locales, y que para el resto de ciudades se siga haciendo manualmente.

13

El camino feliz

Dividir la historia de usuario en:

- una parte que dará soporte al caso de prueba perfecto o “happy path”

- otras historias que agreguen validaciones y soporte a las excepciones (“sad path”)

14

Un escenario de prueba o caso por vez

Si una historia incluye varios escenarios o casos de prueba:

- reemplazarla por historias que resuelvan cada escenario o caso específico

- asignar mayor prioridad a los escenarios o casos de prueba que aportan más valor

15

Un browser o plataforma por vez

Si una historia o el producto completo exigen compatibilidad con varios navegadores o plataformas

- dividir la historia creando una para cada navegador o plataforma

- dar mayor prioridad a la historia que aporte mayor valor

16

Historias con servicios externos

Si una historia consume servicios externos, podemos:

- primero simular la dependencia (*mocking*) para aislarla y simular su comportamiento o

- comenzar con una utilización básica del servicio externo que luego se siga mejorando de forma incremental

17

Por usabilidad

Si una historia incluye varios escenarios de usabilidad que requieren diferentes formas de navegación, podemos:

- Crear una historia con el escenario más relevante

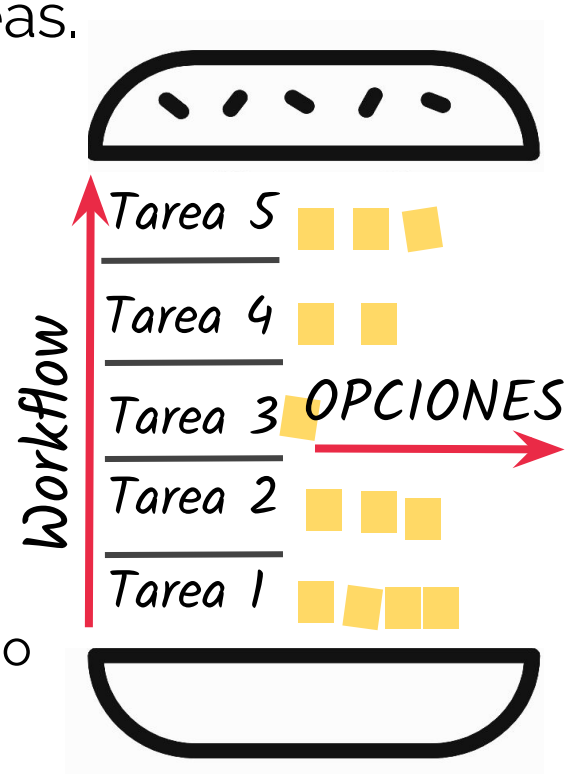
- Dejar los demás escenarios y detalles incluidos en otras historias

18

Método de la hamburguesa

Para una historia con workflow complejo que exigen “todo o nada”, podemos seguir un proceso similar a un User Story Map con planes de entrega, que incluya todos los pasos o tareas.

1. Identificar los pasos o tareas del proceso
2. Identificar sus atributos de calidad
3. Identificar opciones de cada paso o tarea
4. Podar o ajustar para tomar lo mínimo indispensable de cada paso o tarea del proceso
5. Escoger la rebanada vertical de funcionalidades end-to-end



Inspirado de Fifty Quick Ideas to Improve your User Stories por G. Adzic y D. Evans.

19

20

kleer™

kl.la/slicing-kards
v6.0

Cartas de Slicing

más en kl.la/slicing-kards

kleer™

AGILE COACHING
& TRAINING