**Tarea 3**

**Ciclo de vida del software (Relación 2)**

1.- ¿Qué cuatro principios rigen el desarrollo ágil expresados en el Manifiesto Ágil?

* Individuos e interacciones sobre procesos y herramientas.
* Software funcionando sobre documentación extensiva.
* Colaboración con el cliente sobre negociación contractual.
* Respuesta ante el cambio sobre seguir un plan.

2.- ¿Qué es una historia de usuario? Consulta el siguiente enlace: <https://es.wikipedia.org/wiki/Historias_de_usuario>

Una historia de usuario es una representación de un [requisito](https://es.wikipedia.org/wiki/Requerimiento_(sistemas)) escrito en una o dos frases, utilizando el lenguaje común del usuario.

Las historias de usuario son utilizadas en las [metodologías de desarrollo ágiles](https://es.wikipedia.org/wiki/Metodolog%C3%ADa_%C3%A1gil) para la especificación de requisitos, acompañadas de la opinión y la discusión de los usuarios y las pruebas de validación.

Las historias de usuario son una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales y sin requerir de mucho tiempo para administrarlos. Las historias de usuario permiten responder rápidamente a los requisitos cambiantes.

Las historias de usuario deben ser:

-Negociables.

-Independientes unas de otras.

-Estimables.

-Valoradas por los clientes y los usuarios, los intereses de los clientes y de los usuarios no siempre coinciden.

-Pequeñas, las historias muy largas son difíciles de estimar.

-Verificables, la verificación debe automatizarse, de manera que pueda ser verificada en cada entrega del proyecto.

Al ser tan corta, representa requisitos del modelo de negocio que pueden implementarse rápidamente y necesitan poco mantenimiento.

Es ideal para proyectos con requisitos volátiles o no muy claros.

3.- Haz un resumen sobre qué se entiende por Lean software y qué principios lo rigen. Consulta el siguiente enlace: <https://es.wikipedia.org/wiki/Lean_software_development>

La metodología de desarrollo de software Lean es una traducción de los principios y las prácticas de la forma de producir hacia el área del desarrollo del software.

El desarrollo puede resumirse en siete principios:

* Eliminar los desperdicios.

Todo lo que no añade valor al cliente se considera un desperdicio, códigos y funcionalidades innecesarias, burocracia, comunicación interna lenta, retraso en el desarrollo del software y unos requisitos poco claros.

* Amplificar el aprendizaje.

El desarrollo de software es un proceso de aprendizaje continuo, a ello se le suman los retos de los equipos de desarrollo y el tamaño del producto final.

El mejor enfoque para mejorar el ambiente de desarrollo de software es amplificar el aprendizaje.

* Decidir lo más tarde posible.

Los mejores resultados se alcanzan con un enfoque basado en opciones, por lo que se pueden retrasar las decisiones tanto como sea posible hasta que se basen en hechos y no en suposiciones y pronósticos inciertos.

Cuanto más complejo es un proyecto, más capacidad para el cambio debe incluirse, así que debe permitirse el retraso de los compromisos importantes.

* Entregar lo más rápido posible.

Cuanto antes se entrega el producto final, más pronto se pueden recibir comentarios y se incorporan en la siguiente iteración.

* Capacitar al equipo.
* Construir integridad intrínseca.

Significa que los componentes separados del sistema funcionan bien juntos, logrando equilibrio entre la flexibilidad, sostenibilidad, eficiencia y capacidad de respuesta.

Esto podría lograrse mediante la comprensión del dominio del problema, resolviéndolo al mismo tiempo.

4. KANBAN. Estudia las ventajas e inconvenientes de tener una pizarra web digital para la metodología Kanban. Puedes consultar los siguientes enlaces:

* <https://leankit.com/learn/kanban/kanban-board/>
* <https://trello.com/es>
* <https://taiga.io/>
* <https://kanbantool.com/es/>



Kanban gestiona de manera general el desarrollo de las tareas, gracias a su visualización del trabajo por fases, permitiendo así evitar la sobrecarga, a la vez que mide el tiempo estimado en el que se debería completar cada tarea.

Kanban nos ofrece un esquema visual donde poder ver fácilmente el estado del trabajo por medio de un tablero donde cada tarjeta representa una tarea. Esta disposición permite mantener a todos los miembros del equipo al día y establecer unas metas asequibles equilibrando el flujo de trabajo.

* Estas son las principales ventajas:
  + Medición del rendimiento: Con el método Kanban podemos medir el rendimiento de los trabajadores o equipos, así como detectar cualquier problema que se genere
  + Organización del flujo de trabajo: El flujo de trabajo queda claro en cada fase del proyecto gracias a la organización de las tareas.
  + Cumplir los tiempos de entrega.
  + Distribución de tareas: Poder ver las tareas que se han hecho, las que se están haciendo y las que quedan por hacer ayuda a una mejor distribución.
  + Mejorar la calidad del producto: Con buena organización y con un buen control de los tiempos se podrán detectar posibles problemas y, por lo tanto, encontrar la solución.
  + Evitar la acumulación de trabajo.
* Por otro lado, los inconvenientes:
  + Coste: Si se usa este método para unidades muy grandes, el almacenamiento del sistema de Kanban será muy costoso.
  + Kanban no es óptimo para todo tipo de proyectos.
  + Es limitado.
  + No permite anticiparse a grandes aumentos de la demanda: Con Kanban resulta difícil manejar cambios de gestión provocados por la acumulación de nuevas tareas, lo que podría provocar un cúmulo de trabajo.

5. KANBAN. Haz un resumen de la metodología Kanban e indica sus diferencias frente a SCRUM. Puedes consultar el siguiente enlace: <https://es.atlassian.com/agile/kanban>

Kanban es un marco de trabajo muy popular a la hora de implementar un desarrollo de software [ágil.](https://www.atlassian.com/es/agile) Los elementos de trabajo se representan visualmente en un [tablero de kanban](https://www.atlassian.com/es/software/jira/templates/kanban), lo que permite a los miembros del equipo ver el estado de cada uno en cualquier momento.

**DIFERENCIAS KANBAM Y SCRUM**

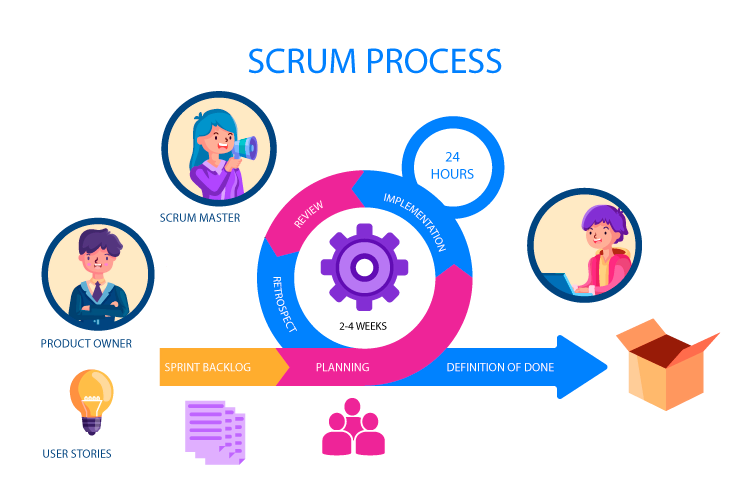
1. Scrum es un marco de trabajo o framework pensado para maximizar el valor entregado en el desarrollo de un producto complejo. En cambio, Kanban está diseñado para optimizar el flujo de trabajo.
2. En Scrum se trabaja en base a un [sprint](https://donetonic.com/que-son-los-sprints-en-scrum/), una interacción de tiempo, mientras que en Kanban hay un trabajo continuo.
3. En Scrum existen los roles, como [Scrum Master](https://donetonic.com/es/que-es-scrum-master/), Product Owner, mientras que en Kanban no existe ningún tipo de rol.
4. En la metodología Scrum no se presentan cambios en el proyecto durante el desarrollo del sprint, en cambio, en Kanban se pueden producir cambios en cualquier momento.
5. En Scrum, se crea un tablero nuevo para cada sprint, mientras que, en Kanban, estos tableros no tienen fecha de inicio ni de fin.
6. Al trabajar con la metodología Scrum, se debe realizar una reunión diaria, el [Daily Sprint](https://donetonic.com/es/que-es-el-daily-scrum-meeting/) para conocer diariamente el estado del desarrollo y tener una visión global de las tareas acabadas y pendientes de todo el equipo de desarrollo, en cambio, en Kanban no existen esas reuniones.
7. Los equipos de trabajo también pueden ser diferentes: en Scrum se exige equipos multidisciplinares, mientras que en Kanban los equipos pueden estar formados por especialistas.

6. SCRUM. Explica cómo funciona Scrum. Consulta los siguientes enlaces:

* <https://proyectosagiles.org/que-es-scrum/>
* <https://proyectosagiles.org/como-funciona-scrum/>

Scrum es un proceso en el que se aplican de manera regular [un conjunto de prácticas](https://proyectosagiles.org/fundamentos-de-scrum) para trabajar en equipo y obtener [el mejor resultado posible](https://proyectosagiles.org/beneficios-de-scrum) de un proyecto.

Se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto.



**CÓMO FUNCIONA**

# La Planificación de la iteración (Sprint Planning)

Se divide en dos partes:

* QUÉ

La primera parte de la reunión, el [cliente](https://proyectosagiles.org/cliente-product-owner) presenta al equipo la lista de requisitos del producto o proyecto, se pone nombre a la iteración, de manera que ayude a tomar decisiones durante su ejecución, y propone los requisitos más prioritarios para este.

El [equipo](https://proyectosagiles.org/equipo-team) examina la lista, pregunta al cliente las dudas que le surgen y añaden más [condiciones de satisfacción](https://proyectosagiles.org/introduccion-estimacion-planificacion-agil#historias-usuario) y selecciona los requisitos más prioritarios.

* CÓMO

La segunda parte de la reunión, el equipo planifica la iteración, elabora la táctica que le permitirá conseguir el mejor resultado posible con el mínimo esfuerzo. Esta actividad la realiza el equipo dado que es el responsable de organizar su trabajo.

Los miembros del equipo se autoasignan las tareas que pueden realizar, se [autoorganizan](https://proyectosagiles.org/2018/10/09/auto-organizacion-fundamentos-y-relacion-con-la-motivacion-intrinseca/) para trabajar.

# Ejecución de la iteración (Sprint)

Cada iteración tiene que proporcionar un resultado completo, un incremento de producto que sea potencialmente entregable, de manera que cuando el cliente lo solicite sólo sea necesario un esfuerzo mínimo para que este esté disponible.

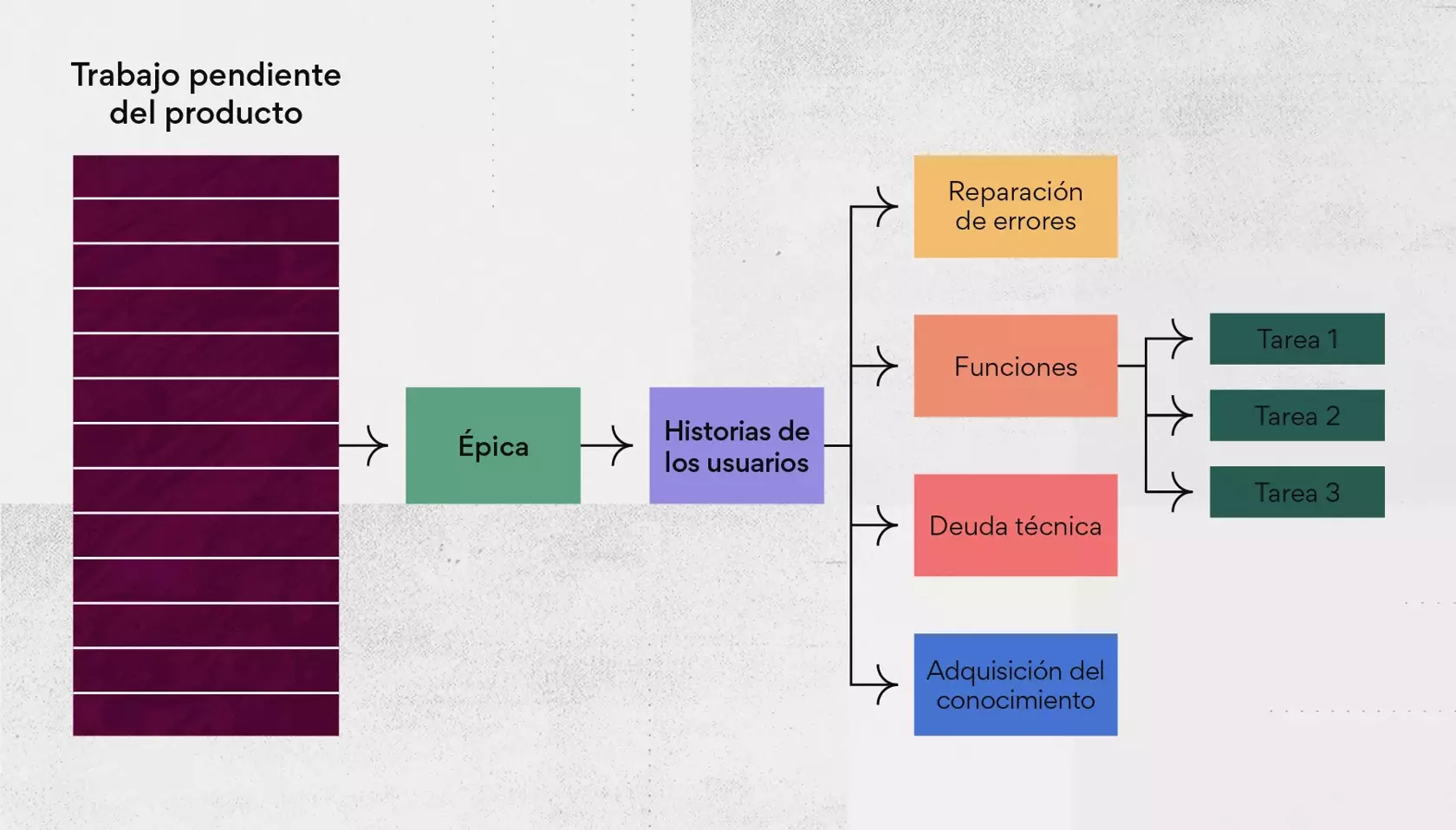
Cada día el [equipo](https://proyectosagiles.org/equipo-team) realiza una reunión, donde cada miembro inspecciona el trabajo de los otros para poder hacer las adaptaciones necesarias, comunica cuales son los impedimentos y actualiza el estado de la [lista de tareas de la iteración.](https://proyectosagiles.org/lista-tareas-iteracion-sprint-backlog)

7. SCRUM. Define los siguientes términos:

* Product backlog.

Es una lista de funciones y elementos, ordenados según las prioridades, que son necesarios para cumplir con los objetivos y las expectativas del proyecto. La regla general es desarrollar uno de estos conjuntos de trabajo pendiente del producto para cada producto y asignarle a un equipo ese trabajo pendiente en particular.

A veces hay muchos conjuntos y varios equipos que trabajan en un solo producto más amplio.



* Sprint backlog.

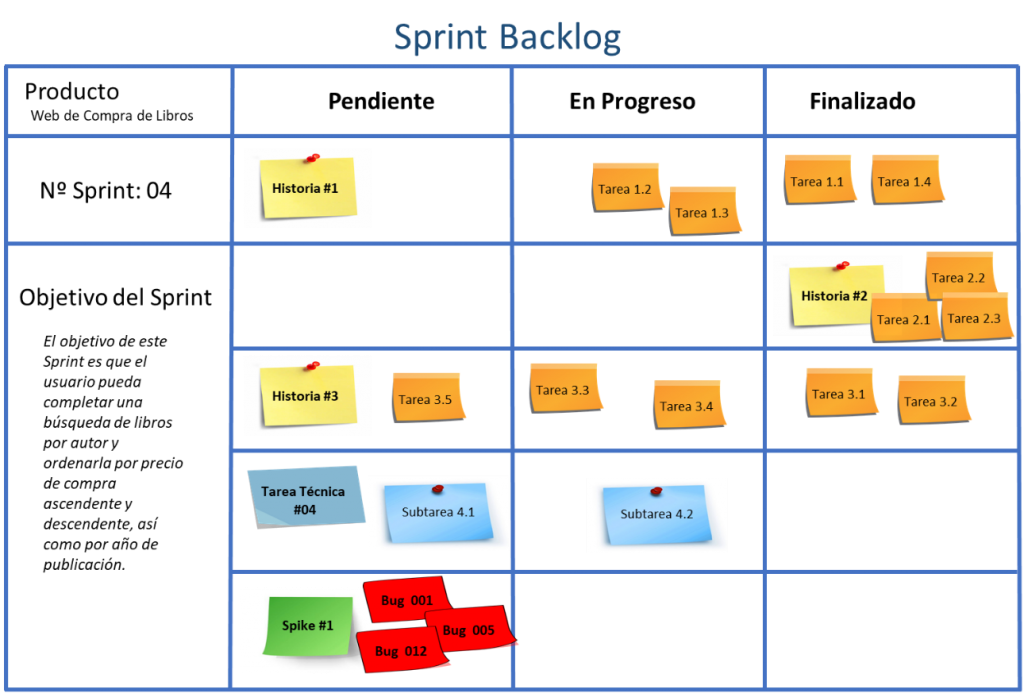
El Sprint Backlog es la suma del Objetivo del Sprint, los elementos del [Product Backlog](https://ittude-agile.com/b/scrum/que-es-el-product-backlog/) elegidos para el [Sprint](https://ittude-agile.com/b/scrum/que-es-un-sprint/) y un plan de acción de cómo crear el [Incremento de Producto](https://ittude-agile.com/b/scrum/que-es-el-incremento-de-producto/).

Permite ver las tareas donde el equipo está teniendo problemas y no avanza, para tomar decisiones al respecto.

Para cada uno de los objetivos y requisitos del proyecto, se muestran las tareas a cubrir, el esfuerzo pendiente para finalizarlas, y la auto asignación que han hecho los miembros del equipo.

La práctica habitual es gestionar el sprint backlog mediante un tablero con post its.

Al lado de cada objetivo se colocan las tareas necesarias para completarlo y se van moviendo para cambiarlas de estado: pendiente, en progreso y hechas.



8. SCRUM. En la terminología Scrum qué términos se utilizan como sinónimo de:

* Jefe de proyecto.

[Scrum Master,](https://donetonic.com/es/que-es-scrum-master/) es el líder del equipo Scrum, aunque sin autoridad jerárquica. Protegerá al equipo y les ayudará a eliminar impedimentos a la vez que colaborará con el Product Owner para priorizar el Product Backlog. Su función principal es asegurar que se sigue metodología Scrum durante todo el proceso de desarrollo.

* Cliente.

### Product Owner, es el representante del cliente y responsable de la gestión del product backlog, es decir, es el responsable de la gestión de la cartera de los productos. Se encargará de maximizar el valor del producto desarrollado por el Equipo Scrum.

Recogerá los requisitos del proyecto, decidirá que se desarrolla y que no, priorizará el Product Backlog y hará que esté visible para todo el mundo.

* Equipo de desarrollo.

Development Team, está compuesto por entre 3 y 9 profesionales, con capacidad de autogestión y multidisciplinares y su tarea es conseguir un incremento en el desarrollo del producto a partir de los elementos del Product Backlog seleccionados durante el [Sprint Planning](https://donetonic.com/que-son-los-sprints-en-scrum/).

9. SCRUM. Haz un resumen de los requisitos para poder utilizar Scrum. Consulta el siguiente enlace:

<https://proyectosagiles.org/requisitos-de-scrum/>

* Equipo, delegación, creatividad y mejora continua.
* [Compromiso del cliente](https://proyectosagiles.org/requisitos-de-scrum#compromiso-cliente) en la dirección de los resultados del proyecto y disponibilidad para poder colaborar.
* [Compromiso de la Dirección](https://proyectosagiles.org/requisitos-de-scrum#compromiso-direccion) de la organización para resolver problemas y realizar cambios.
* [Compromiso conjunto y colaboración de los miembros del equipo](https://proyectosagiles.org/requisitos-de-scrum#compromiso-equipo).
* [Facilidad para realizar cambios](https://proyectosagiles.org/requisitos-de-scrum#facilidad-cambios) en el proyecto.
* Equipo trabajando en un mismo [espacio común](https://proyectosagiles.org/requisitos-de-scrum#espacio-comun) para maximizar la comunicación.
* Dedicación del [equipo a tiempo completo](https://proyectosagiles.org/requisitos-de-scrum#equipo-tiempo-completo).

10. XP. Explica los 5 valores de la Programación Extrema.

* Simplicidad:

Se simplifica el diseño para agilizar el desarrollo y facilitar el mantenimiento.

Un diseño complejo del código junto a sucesivas modificaciones por parte de diferentes desarrolladores hace que la complejidad aumente bastante.

Para mantener la simplicidad es necesaria la refactorización del código, esta es la manera de mantener el código simple a medida que crece.

* Comunicación:

La comunicación se realiza de diferentes formas, para los programadores, el código comunica mejor cuanto más simple sea, si el código es complejo hay que esforzarse para hacerlo legible.

* Retroalimentación:

Al estar el cliente integrado en el proyecto, su opinión sobre el estado del proyecto se conoce en tiempo real.

Al realizarse ciclos muy cortos, se minimiza el tener que rehacer partes que no cumplen con los requisitos y ayuda a los programadores a centrarse en lo más importante.

* Valentía o coraje:

Muchas de las prácticas implican valentía, y una de ellas es siempre diseñar y programar para hoy y no para mañana.

La valentía les permite a los desarrolladores que se sientan cómodos con reconstruir su código cuando sea necesario. Esto significa revisar el sistema existente y modificarlo si con ello los cambios futuros se implementarán más fácilmente.

Otro ejemplo de valentía es saber cuándo desechar un código, la valentía para quitar el código fuente obsoleto, sin importar cuanto esfuerzo y tiempo se invirtió.

* Respeto o humildad:

Los miembros del equipo se respetan los unos a otros, porque los programadores no pueden realizar cambios que hacen que las pruebas existentes fallen o que demore el trabajo de sus compañeros.

Los miembros respetan su trabajo porque siempre están luchando por la alta calidad en el producto y buscando el diseño óptimo.

11. XP. ¿Cuáles son las características distintivas de XP frente a otras metodologías ágiles? Explícalas. Puedes consultar el siguiente enlace: <http://www.davidvalverde.com/blog/introduccion-a-la-programacion-extrema-xp/>

* Desarrollo iterativo e incremental.
* [Pruebas unitarias](https://es.wikipedia.org/wiki/Prueba_unitaria) continuas.
* Frecuente integración del equipo de programación con el cliente o usuario.

Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.

* Corrección de todos los [errores](https://es.wikipedia.org/wiki/Error_de_programaci%C3%B3n) antes de añadir nueva funcionalidad.
* [Refactorización](https://es.wikipedia.org/wiki/Refactorizaci%C3%B3n) del código.

Reescribir ciertas partes del código para aumentar su legibilidad y [mantenibilidad](https://es.wikipedia.org/wiki/Mantenibilidad) pero sin modificar su comportamiento.

* Simplicidad en el código.

La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.