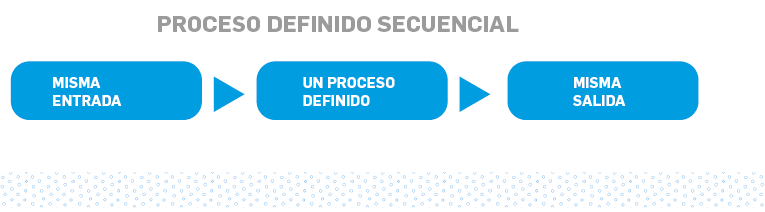
¿Qué es el desarrollo de software?

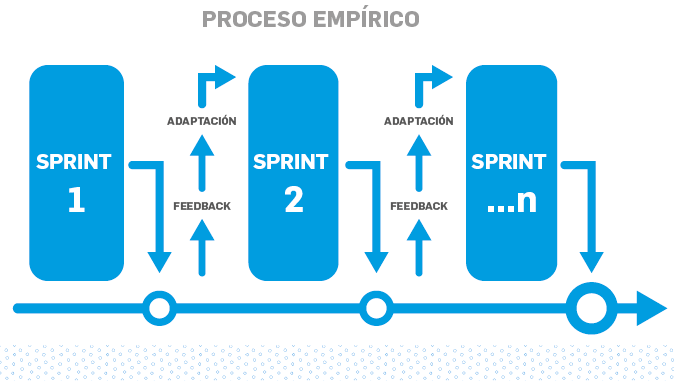
**Desarrollo de Software**

Según la [IEEE](https://www.ieee.org/) , el software es el “ *conjunto de programas de cómputo, procedimientos, reglas, documentación y datos asociados, que forman parte de las operaciones de un sistema de computación* ”. Es la parte de un sistema que se puede codificar para ejecutarse en una computadora como un conjunto de instrucciones, e incluye la documentación asociada necesaria para comprender, transformar y usar esa solución. Estos documentos describen la organización del sistema, explican al usuario cómo utilizarlo y obtener eventuales actualizaciones del producto.

El software es un producto intangible, de alto contenido intelectual, que no sufre desgaste alguno y que puede ser potencialmente modificado de forma permanente. No se manufactura sino que se desarrolla a través de proyectos que son realizados por equipos de personas altamente formadas. A pesar de la tendencia a desarrollar componentes que puedan ser reutilizados y adaptados a diferentes necesidades, la gran mayoría del software se construye a medida.

El proceso de desarrollo de software es un conjunto de actividades que da como resultado un producto que responde a las necesidades de un usuario. Este proceso define todas aquellas actividades necesarias para transformar los requisitos de un usuario en un producto. Estas actividades se las puede organizar con diferentes criterios y, como es un proceso intelectual-creativo, depende de las personas que formen parte del equipo y de sus decisiones y juicios. Existen varios procesos o modelos de desarrollo y podríamos catalogarlos en Secuenciales o Definidos, y en Empíricos. Los primeros responden siempre de la misma forma ante una determinada entrada, mientras que los segundos dependen de la evolución y adaptación de etapas anteriores.





El conjunto de fases (o procesos) por las que pasa el software desde que se concibe y se desarrolla hasta que finaliza su uso (retiro de servicio) se conoce como **ciclo de vida del software** . Estas fases están estandarizadas, existiendo un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, explotación y mantenimiento de un producto software, abarcando la vida del sistema, desde la definición de sus requerimientos hasta la finalización de su uso.

##### **Metodología en Cascada**

El modelo de desarrollo en cascada fue presentado por primera vez en 1970 por [Royce](https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/el-modelo-en-cascada/) . Se lo conoce también como modelo lineal secuencial. Su principal característica es que cada fase del desarrollo está bien definida y separada, siguiendo un orden secuencial en el que cada etapa dependiente de la finalización de la anterior y que esta última haya pasado un proceso de validación que apruebe el paso a la siguiente.

Comúnmente las etapas del modelo son:

1. Análisis y definición de los requerimientos.
2. Diseño del sistema y del software.
3. Implementación y pruebas unitarias.
4. Integración y pruebas de sistema.
5. Funcionamiento y mantenimiento.



Imaginemos que somos un equipo de desarrollo de software y recibimos un pedido de la empresa “ArgProg”. Se trata de una PyME nacional del rubro farmacéutico que, en los últimos años, experimentó un importante crecimiento gracias a su capacidad de idear soluciones innovadoras a problemas de la industria. Como resultado, la empresa incrementó tanto su plantilla de empleados como su cartera de proyectos. En este nuevo contexto, “ArgProg” nos demanda el desarrollo de un software que mantiene conectadas a las personas y a los proyectos, ya que esa interacción es la base de la innovación permanente.

Si bien es un gran desafío, conocemos el recorrido tradicional para responder a esta demanda:

* 1.       nos reunimos internamente para decidir si tomamos el proyecto,
* 2.      invertimos tiempo y recursos en diseñar hasta el último detalle de la solución que vamos a ofrecer a la empresa (alcance del proyecto),
* 3.       listamos y ordenamos todas las tareas que serán necesarias para construir esa solución (descomposición de la estructura de trabajo)
* 4.       estimamos cuánto tiempo y recursos implicará la realización de esas tareas (cronograma y presupuesto del proyecto). Tomamos como base para esta estimación los proyectos similares que hicimos en el pasado.

Una vez que plasmamos toda esta información en documentos (planificación), se lo presentamos a la empresa y, si logramos un acuerdo, entonces sí:

* 5.       Ejecutamos las tareas planificadas (ejecución)
* 6.       Monitoreamos el progreso para evitar desvíos respecto de lo que habíamos planificado en los documentos (monitoreo y evaluación)
* 7.       Completamos el desarrollo del software, se lo entregamos a nuestro cliente y ¡finalizamos el proyecto! (cierre).

Esta metodología para gestionar proyectos, denominada “metodología tradicional”, parece clara y relativamente sencilla de implementar ¿no?   
Sin embargo, según datos del [Standish Group](https://www.standishgroup.com/" \t "_blank), más del 70% de los proyectos de desarrollo de software gestionados bajo esta metodología no tienen éxito.

Las principales razones:

* Suponen que los usuarios del software tienen completamente clara su necesidad o deseo y esto muchas veces no ocurre en la realidad. En el ejemplo, “ArgProg” busca “mantener conectados a las personas y a los proyectos”, sin embargo ¿qué implica estar “conectados”? es algo difuso... ¿Alcanza con que sepan qué están desarrollando los otros equipos? ¿Lo relevante es que estén al tanto de los desafíos que enfrentan sus compañeros? ¿Aportar sus experiencias o saberes para ayudarlos a resolverlos? A esta falta de claridad sobre la necesidad hay que sumarle otro factor: la multiplicidad de actores involucrados. La empresa está conformada por un directorio, gerentes, mandos medios, trabajadores de línea, cada uno de ellos con sus propios intereses.
* Suponen que el equipo de desarrollo puede crear la solución ideal y describirla de forma precisa y detallada antes de empezar a construirla. En el caso de “ArgProg” implicaría tener absoluta certeza sobre la arquitectura que tendrá el software, cada aspecto de su funcionamiento, cada una de las funcionalidades que incluirá, cada detalle del “look and feel”, etcétera, antes de escribir el primer código. Idear una solución y plasmarla en documentos con este nivel de detalle es, en la mayoría de los casos, inviable y lleva a que se desarrollen más funcionalidades de las deseadas o muchas menos de las necesarias, a que se sobreestimen aspectos que no agregan valor al cliente y se descarten aquellos que fueron realmente valiosos.
* Dada la claridad de las necesidades y la capacidad de idear una solución que las satisfaga, las metodologías tradicionales suponen que es posible planificar todos los aspectos del proyecto (actividades, cronograma y presupuesto) y ajustarse a ese plan. El problema es que esto sólo es viable cuando contamos con experiencias anteriores en desarrollos similares, las cuales podemos usar como referencia para estimar el esfuerzo que demandará el proyecto actual. Esto no es viable cuando apuntamos a generar soluciones innovadoras para las cuales no contamos con antecedentes para planificar con certeza.

## Metodología iterativa e incremental

#### La metodología iterativa e incremental deriva del proceso de desarrollo en cascada pero, con la diferencia de que aquí se admite que las etapas se solapan en tiempo con la finalidad de flexibilizar el tiempo de desarrollo total y así poder alcanzar resultados funcionales de manera temprana.

#### C:\Users\54381\Downloads\Recurso 9.png

Esta metodología involucra dos procesos fundamentales:

* ●       El proceso **incremental** : con esto se busca desarrollar una parte del producto que se pueda integrar al conjunto a medida que se alcanza un grado de completitud.
* ●       El proceso **iterativo** : se realiza en ciclos donde se revisa y mejora el producto. De manera que la calidad del producto aumente, y no siempre implica la integración de nuevas funcionalidades.

##### **Procesos Ágiles**

A fin de superar las dificultades o límites de las metodologías tradicionales, surgieron las metodologías ágiles para la gestión de proyectos. Los métodos o procesos ágiles consideran un enfoque iterativo para las etapas de especificación, desarrollo y entrega del software. Estos métodos fueron pensados ​​para el desarrollo de aplicaciones donde los requerimientos del sistema cambian rápidamente y también están enfocados en poder entregar software funcional de forma rápida a los usuarios, quienes pueden evaluar y proponer nuevos requerimientos o cambios para iteraciones próximas.

Estos procesos se basan en los valores y los principios definidos en el [Manifiesto Ágil](https://agilemanifesto.org/iso/es/principles.html) elaborado en 2001 por un grupo de profesionales críticos de los modelos de desarrollo de software propuestos hasta el momento a los que consideraban excesivamente pesados ​​y rígidos por su carácter normativo y fuerte dependencia de planificaciones detalladas previas al desarrollo.

Los valores que propone este manifiesto son:

* ●       Valorar más a los individuos y sus interacciones que a los procesos y las herramientas
* ●       Valorar más el software funcionando que la documentación exhaustiva
* ●       Valorar más la colaboración con el cliente que la negociación contractual
* ●       Valorar más la respuesta ante el cambio que seguir un plan

También se ven los siguientes 12 principios:

1. Nuestra principal prioridad es satisfacer al cliente a través de la entrega temprana y continua de software de valor.
2. Son bienvenidos los requisitos cambiantes, incluso si llegan tarde al desarrollo. Los procesos ágiles se doblegan al cambio como ventaja competitiva para el cliente.
3. Entregar con frecuencia software que funcione, en periodos de un par de semanas hasta un par de meses, con preferencia en los períodos breves.
4. Las personas del negocio y los desarrolladores deben trabajar juntos de forma cotidiana a través del proyecto.
5. Construcción de proyectos en torno a individuos motivados, dándoles la oportunidad y el respaldo que necesitan y procurándoles confianza para que realicen la tarea.
6. La forma más eficiente y efectiva de comunicar información de ida y vuelta dentro de un equipo de desarrollo es mediante la conversación cara a cara.
7. El software que funciona es la principal medida del progreso.
8. Los procesos ágiles promueven el desarrollo sostenido. Los patrocinadores, desarrolladores y usuarios deben mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica enaltece la agilidad.
10. La simplicidad como arte de maximizar la cantidad de trabajo que no se hace, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos que se auto-organizan.
12. En intervalos regulares, el equipo reflexiona sobre la forma de ser más efectivo y ajusta su conducta en consecuencia.

Si bien hay diferentes métodos ágiles, todos comparten algunos **principios elementales** :

* ●       **Participación del cliente**: Los clientes deben estar implicados en todo el proceso de desarrollo, su rol está enfocado en proporcionar y dar prioridad a nuevos requerimientos del sistema, como así también participar en la evaluación de los entregables de cada iteración.
* ●       **Entrega incremental**: El software se desarrolla en incrementos y el cliente es quien determina qué necesita incluir en cada incremento.
* ●       **Enfocado en personas, no en procesos**: Hay que identificar y explotar las habilidades de cada una de las personas del equipo. Cada equipo debe desarrollar sus propias formas de trabajar, es decir, sin imponer procesos formales.
* ●       **Aceptar el cambio** : Siempre se debe tener en cuenta que los requerimientos del sistema pueden cambiar, y por lo tanto el diseño debe contemplar esto.
* ●       **Mantener la simplicidad** : Se debe enfocar la simplicidad tanto en el software que se desarrolla como en el proceso de desarrollo.

Existen diferentes metodologías ágiles que si bien comparten los mismos principios, cada una posee sus propias características que las diferencian entre sí. Algunas de ellas son la [Programación Extrema](https://openwebinars.net/blog/extreme-programming-que-es-y-como-aplicarlo/) (XP, Extreme Programming), el [Desarrollo Rápido de Aplicaciones](http://metodologiarad.weebly.com/) , el Desarrollo de prototipo, y Scrum.

¿Qué es Scrum?

**SCRUM**

SCRUM fue creado por [Ken Schwaber](https://www.scrum.org/team/ken-schwaber) y [Jeff Sutherland](https://scrumguides.org/jeff.html) a partir de la reflexión sobre su propia práctica y la de sus colegas, y actualmente es el enfoque más utilizado a nivel mundial para el desarrollo de software. Es especialmente útil para trabajar en entornos complejos en los cuales tanto la necesidad a cubrir como la solución que la satisfaga no son claras de antemano y sólo pueden conocerse a medida que se construye y se pone a prueba el software.

Scrum es un marco de trabajo a través del cual las personas pueden abordar problemas complejos adaptativos, a la vez que se entregan productos de forma eficiente y creativa con el máximo valor. En scrum se aplica un conjunto de buenas prácticas para trabajar de manera colaborativa y obtener así el mejor resultado posible en proyectos complejos que demandan constantes cambios y adaptaciones.

La palabra scrum tiene su origen en un ámbito alejado al de la gestión de proyectos: el deporte. En rugby, “Scrum” es el término que define a la formación en la que ambos equipos empujan de forma coordinada para obtener la pelota. Esta formación requiere a cada equipo empujar no sólo en la misma dirección sino al mismo tiempo para lograr la suma de las fuerzas individuales. Representa una claro ejemplo de trabajo en equipo y coordinación entre todos los miembros de un mismo equipo.

Uno de los pilares de Scrum es la **transparencia**. Los aspectos significativos del proceso deben ser visibles para aquellos que son responsables del resultado. Esto requiere que dichos aspectos sean definidos en base a un estándar común, de tal modo que los observadores compartan un entendimiento común de lo que están viendo. Por ejemplo:

* Deben compartir un lenguaje común todos los participantes para referirse al proceso; y,
* Aquellos que desempeñan el trabajo y quienes inspeccionan el incremento deben compartir una definición común de “Terminado”.

Otro de los pilares es la **inspección**. Los usuarios de Scrum deben inspeccionar frecuentemente los artefactos de Scrum y el progreso hacia un objetivo para detectar variaciones indeseadas. Su inspección no debe ser tan frecuente como para que pueda interferir en el trabajo. Las inspecciones son más beneficiosas cuando se realizan de forma diligente por inspectores expertos en el mismo lugar de trabajo.

Finalmente, otro pilar de la metodología es su **adaptación**. Si un inspector determina que uno o más aspectos de un proceso se desvían de los límites aceptables y que el producto resulta inaceptable, el proceso o el material que está siendo procesado deben ajustarse. Dicho ajuste deberá realizase cuanto antes para minimizar desviaciones mayores.

Desde la perspectiva de SCRUM, un proyecto se considera exitoso si entrega el mayor valor posible para el usuario en un tiempo dado. ¿Cómo sabemos qué aporta valor al o los usuarios? A medida que se va construyendo y testeando el software se va comprendiendo más profundamente la problemática y la necesidad de los usuarios y las posibles formas de satisfacerlo (incluso los mismos usuarios al interactuar con el producto van terminando de comprender su propia necesidad).

Los proyectos se llevan a cabo en equipos que trabajan en ciclos temporales cortos y duración fija. Cada ciclo se conoce como Sprint y tiene que proporcionar un resultado completo, un incremento del producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando este lo solicite. Los componentes del ciclo son:

* **Eventos**: Planificación del sprint, Scrum diario, Revisión del sprint, y Retrospectiva del sprint
* **Roles**: Desarrollador, Propietario del producto, y Scrum master
* **Artefactos**: Pila del producto, Pila del sprint, e Incremento.



*Ciclo de desarrollo (Sprint) siguiendo metodología Scrum*

## Eventos - Planificación en Scrum

#### ****Eventos****

El núcleo de esta metodología son los Eventos. Cada uno de estos, se enmarca en el ciclo iterativo denominado Sprint y hace que sea posible llegar a un entregable útil al final de cada iteración. **Los *Sprints*** son el corazón de la metodología, **son períodos de tiempo fijos**, generalmente de un mes o menos, y comienzan inmediatamente después de finalizado el Sprint anterior. Durante un Sprint no se pueden hacer cambios que perjudiquen los objetivos del mismo, se refina la pila de producto y eventualmente se podría clarificar o renegociar el alcance con el propietario del producto a medida que se aprende sobre el avance. Solo si el objetivo se vuelve obsoleto, el propietario del producto tiene la autoridad para cancelar el Sprint.

##### **Planificación de Sprint**

La planificación es el evento que da inicio al Sprint y es donde se lleva a cabo la organización del trabajo a realizar. Este evento no es necesariamente exclusivo del equipo de Scrum, sino que se puede invitar a otras personas para que aconsejen en los temas pendientes.

Durante la planificación se tratan los siguientes temas:

* ●       ¿Por qué es valioso este Sprint ?

El propietario del producto propone cómo el va a aumentar su valor y utilidad con el Sprint. Todo el equipo colabora para definir el objetivo del Sprint que transmite el por qué es valioso para los interesados ​​en el producto. Al final de este evento el objetivo debe estar bien definido.

* ●       ¿Qué se puede hacer en este Sprint ?

A lo largo de la discusión, los desarrolladores eligen ítems de la pila de producto, para incluirlos en el Sprint en curso. Durante este proceso, el equipo puede refinar estos ítems mejorando el entendimiento sobre cada uno. Este tema, de conocer cuánto se puede hacer, los desarrolladores lo entrenan teniendo en cuenta su rendimiento pasado, la capacidad y su definición de Terminado.

* ●       ¿Cómo se llevará a cabo el trabajo seleccionado?

Para cada uno de los elementos seleccionados, los desarrolladores planean el trabajo necesario para crear un incremento que cumpla con la definición de Terminado. La manera en que estos lo ejecutan queda a su discreción y suelen descomponer los ítems de la pila de producto en tareas más pequeñas, de un día o menos.

##### **Scrum Diaria**

Las Scrum diarias (Daily Scrum) son reuniones de no más de 15 minutos, en las cuales se sincroniza el trabajo desarrollado y se establece el plan para el día en curso. Generalmente se recomienda hacerlas viendo el tablero con la pila de sprint.  Se comparten los avances, los problemas y las ideas, también si es necesario se actualiza la pila de Sprint. Es importante tener en cuenta que no es una reunión de inspección o control, sino que de comunicación entre el equipo.

##### **Revisión de Sprint**

Sobre el final del Sprint se hacen las reuniones de revisión de Sprint para comprobar el incremento. Estas suelen durar entre una y dos horas, en caso de que el incremento tenga una complejidad muy alta, podrían extenderse aún más. Estas pueden permitir al equipo y al propietario del producto tomar sensibilidad del ritmo de construcción y la trayectoria que cobra la visión del producto. Son instancias en las que se reajusta la pila de producto para incluir nuevas oportunidades.

##### **Retrospectiva de Sprint**

El propósito de las reuniones de Retrospectiva de Sprint es planear nuevas formas de mejorar la calidad y efectividad. Se analiza lo que sucedió en la última Sprint con respecto a los individuos, sus interacciones, los procesos, lo que fue bien y los problemas que se encontraron, los que se resolvieron y los que quedaron pendientes.

## Roles - equipo SCRUM

#### ****Roles / Equipo****

La unidad fundamental de Scrum es un equipo de personas (Scrum Team). Cada equipo consiste en un Scrum Master, un propietario del producto (Product Owner)  y desarrolladores (Developers).

El equipo debe ser suficientemente pequeño como para ser ágil y grande como para completar una cantidad considerable de trabajo dentro de un Sprint. Normalmente los equipos están conformados por 10 personas o menos, ya que en general, los equipos pequeños se comunican mejor y son más productivos.

El equipo es responsable de todas las actividades relacionadas al producto desde la colaboración con las partes interesadas, verificación, mantenimiento, experimentación, desarrollo y cualquier otra cosa que pueda ser requerida.

##### **Desarrolladores**

Los desarrolladores son aquellos que producen cada uno de los incrementos de cada sprint y siempre son responsables de:

* ●       Crear un plan para el Sprint, la pila de Sprint (Sprint Backlog).
* ●       Asegurar la calidad mediante la adopción de una Definición de Terminado.
* ●       Adaptar el Plan de Cada Día para conseguir el Objetivo del Sprint (Sprint Meta).

##### **Propietario del Producto**

El Propietario del Producto es el responsable de Gestionar Efectivamente la Pila de Producto (Product Backlog), Lo que incluye:

* ●       Desarrollar y Comunicar el objetivo m del Producto (Meta del producto).
* ●       Crear y comunicar cada uno de los artículos de la pila de producto.
* ●       Ordenar los artículos de la pila de producto siguiendo alguna prioridad.
* ●       Asegurar que la pila de producto sea transparente, visible y comprensible.

Es importante resaltar que el propietario del producto es una única persona, y por tanto, se deben respetar sus decisiones. Estas, deben ser reflejadas en el contenido y orden en la pila de producto, como así también en el incremento y la revisión del Sprint.

##### **Scrum Master**

El Scrum Master es el encargado de asegurar que la metodología se lleve a cabo tal y como está definida. Esto se logra haciendo que cada persona del equipo entienda la teoría y práctica de la metodología.

La persona con el rol de Scrum Master sirve al equipo de Scrum de varias formas:

* ●       Orienta a los miembros del equipo en la autogestión y la multifuncionalidad.
* ●       Ayuda a los equipos a enfocarse en crear incrementos de alto valor que cumplan con la definición de Terminado.
* ●       Remueve los impedimentos de progreso que pudiera tener el equipo.
* ●       Asegura que todos los eventos de Scrum se lleven a cabo y que sean positivos, productivos y dentro de los tiempos establecidos.

El Scrum Master también colabora con el propietario del producto de las siguientes maneras:

* ●       Ayuda a encontrar técnicas para definir efectivamente el objetivo del producto y la gestión de la pila de producto.
* ●       Ayuda a que los miembros del equipo entiendan los artículos de la pila de producto.
* ●       Ayuda a establecer un planeamiento del producto de manera empírica para entornos que sean complejos.

Finalmente, también colabora con el resto de la organización con actividades como:

* ●       Liderar, planear y entrenar a los miembros de la organización en lo que respeta a la adopción de Scrum.
* ●       Planear y aconsejar cómo implementar Scrum dentro de la organización.
* ●       Ayudar a los empleados a comprender y llevar a cabo la práctica de la metodología, mediante un enfoque empírico para trabajos complejos.

Durante la reunión de planificación de Sprint, el Scrum Master deberá desempeñar las siguientes funciones:

* ●       Realizar esta reunión antes de cada Sprint.
* ●       Asegurar que se cuenta con una pila de producto preparada por el propietario del producto.
* ●       Ayudar a mantener el diálogo entre el propietario del producto y los desarrolladores.
* ●       Asegurar que se llegue a un acuerdo entre el propietario del producto y los desarrolladores con respecto a lo que incluirá el incremento.
* ●       Ayudar a comprender la visión y las necesidades de negocio del cliente.
* ●       Asegurar que se ha realizado una descomposición y estimación del trabajo realistas.
* ●       Asegurar que al final de la reunión estén determinados los siguientes puntos:
  + 1. Los ítems de la pila de producto que se van a ejecutar
    2. El objetivo del Sprint.
    3. La pila de Sprint con todas las tareas estimadas.
    4. La duración del Sprint y la fecha de reunión de revisión.
    5. La definición de Terminado que determinará que el incremento está listo.

## Artefactos SCRUM

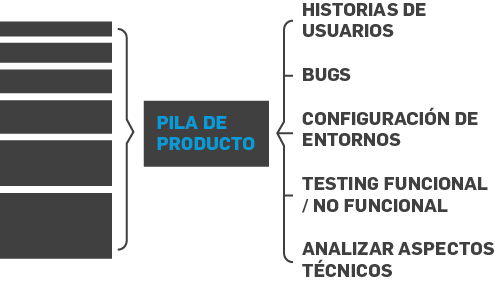
#### ****Artefactos****

##### 

##### **Pila de Producto**

La Pila de Producto (Product Backlog) es una lista ordenada. Cada una de las entradas de esta lista son posibles trabajos a seleccionarse para su realización durante una reunión de planificación de Sprint . El refinamiento de los ítems de la pila de producto es el acto de convertir esos ítems en elementos más detallados y precisos, en este proceso también se les asigna una prioridad y un “tamaño”.

La pila de producto puede incluir ítems para explorar las necesidades del cliente, analizar opciones técnicas, y otros ítems de trabajo tales como la corrección de errores (bugs) o la  configuración del entorno. Todo lo que esté en la pila de producto representa o aporta a conseguir el objetivo del producto (Product Goal) y este último es el objetivo que el equipo tiene que conseguir cumplir a largo plazo.



Fuente: Tipos de elementos en la pila de producto

##### **Pila de Sprint**

Este artefacto, pila de Sprint (Sprint Backlog), delimita el trabajo necesario para alcanzar determinado incremento, y sirve para marcar el avance. También sirve como herramienta de comunicación del equipo. Esto es visible para todos y representa una imagen en tiempo real del trabajo que llevan los desarrolladores para alcanzar el Objetivo del Sprint (Sprint Meta). Este último se crea durante la reunión de planificación de Sprint y a lo largo de todo el ciclo, los desarrolladores lo tienen presente.

##### **Incremento**

Un incremento de producto es un paso concreto que acerca el desarrollo al objetivo del producto. Cada incremento agrega valor al incremento anterior y bajo un proceso de verificación, se asegura que todos los incrementos trabajen bien juntos.

Existe un compromiso entre el incremento y la Definición de Terminado (Definición de Hecho). Esta definición es una descripción formal del estado que debe alcanzar el incremento para cumplir con el nivel de calidad requerido para el producto. Si por alguna razón algún ítem de la pila de producto no cumple con la definición de terminado, no debe formar parte del entregable y ni siquiera ser presentado en la reunión de revisión de Sprint. En lugar de esto, debe volver a la pila de producto para ser considerado en un futuro.

