# *Product Manager*

## **Clase 6 - Requerimientos y técnicas**

Objetivo de la clase:

* Entender cómo nacen los requerimientos y la importancia de un buen relevamiento de información antes de comenzar un proyecto

## **Requerimientos**

Desarrollar un producto nuevo puede ser una de las tareas más exigentes de un Product Manager, para tener éxito debemos poner en marcha algunos procesos de recolección, análisis, verificación y validación de información que nos provea de una base sólida al momento de crear nuestro producto.

Un requerimiento puede definirse de diversas maneras:

* Una condición o capacidad requerida por un interesado para resolver un problema o alcanzar un objetivo (IEEE 610.12-1990).
* Una declaración de una necesidad u objetivo, o una condición o capacidad que un producto debe poseer para satisfacer tal necesidad u objetivo (“Software Requirements,” by Karl Wiegers).
* Algo que el producto debe de hacer o una cualidad que debe de poseer (“Mastering the Requirements Process,” by Suzanne & James Robertson).

Es evidente múltiples definiciones de lo que es un requisito o requerimiento. Resumiendo: es una capacidad que un producto debe de poseer o algo que un producto debe hacer a fin de satisfacer una necesidad.

A la vez, existen tipos de requerimientos. En un macronivel podríamos observar los siguientes:

### **REQUERIMIENTOS DE NEGOCIO**

Definen los objetivos y problemas (y oportunidades) que la empresa quiere resolver con el producto. Deben estar basados en una necesidad real del usuario, sea conocida o no por él. Si queremos verlo en profundidad, el trabajo previo a los requerimientos de negocio nace analizando:

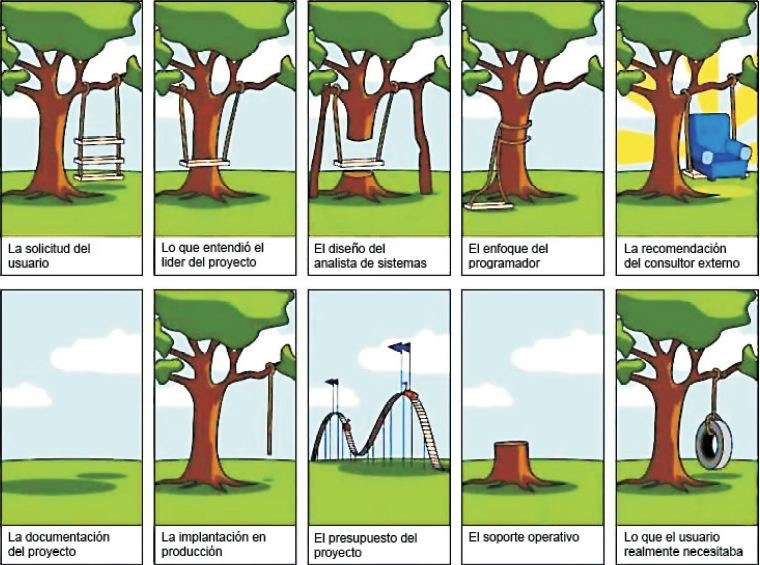
* Panorama competitivo: ¿Qué está haciendo la competencia y cómo debemos ofertarnos para diferenciarnos de ella? ¿Dónde está fallando la competencia?
* Panorama del cliente: ¿Cuáles son las necesidades de nuestros clientes? ¿Qué huecos podemos llenar?
* Panorama del usuario: ¿Cuáles son las necesidades de los usuarios? ¿Qué capacidades pueden mejorar su habilidad para terminar sus trabajos?
* Deuda técnica: ¿Cuánto debes invertir para sobrepasar los límites del código heredado? No es habitual encontrarse con un producto bien construido. La velocidad en la que emergen nuevas tecnologías de desarrollo es muy rápida, y un producto con poca dedicación al mantenimiento suele quedar atrasado u obsoleto con el paso del tiempo. Esto generalmente aplica al desarrollo de software, sin embargo, también ocurre con los procesos de trabajo, una comunicación fluida y clara entre el Squad y la Organización es primordial para reducir la incidencia de la deuda técnica en el desarrollo de nuestro producto.

### **REQUERIMIENTOS DE USUARIO**

Responde a la pregunta: ¿Qué necesita hacer el usuario con el sistema? Estos requisitos describen las necesidades de los interesados, para ayudarles a lograr los objetivos. Describen los requerimientos funcionales y no funcionales de tal forma que sean comprensibles por los usuarios del sistema. ¿Por qué debemos levantar requerimientos? 👀 Nos permiten vincular los requerimientos del negocio con los de la solución. Algunos de los beneficios:

* Diseñar productos de mayor calidad.
* Facilitar la estimación de tiempos de desarrollo.
* Contribuir a la priorización de tareas.
* Testear de manera más eficiente.

Lo que solicitamos no siempre es lo que necesitamos; y lo solicitado no es siempre lo que se imaginaba el solicitador. Veamos el ejemplo 🙋‍♀️:



### **REQUERIMIENTOS FUNCIONALES Y NO FUNCIONALES**

* **Funcionales**: Son aquellos que describen cualquier actividad que se debe realizar. En otras palabras, el comportamiento o función particular de un sistema o software cuando se cumplen ciertas condiciones. Por ejemplo: El sistema debe realizar transferencias bancarias 💸.
* **No funcionales:** Son los que especifican criterios para evaluar la operación de un servicio, en contraste con los requerimientos funcionales que especifican los comportamientos específicos de la acción y respuesta. Por ejemplo: El sistema, luego de hacer una transferencia, podría mostrar una ventana preguntando si deseamos realizar otra operación 🏁.

Para resumir, los Requerimientos Funcionales describen lo que el producto debe hacer, mientras que los No Funcionales indican atributos, características o propiedades que el mismo debe tener.

## **Técnicas de relevamiento**

Resulta imperativo realizar un abordaje múltiple al momento de relevar requerimientos, tanto de negocio como de usuarios, para englobar las distintas etapas del proceso de creación y desarrollo del producto que tenemos a cargo como PM. Para ello, existen las técnicas de relevamiento. Las siguientes técnicas son recomendadas para reducir problemas antes, durante y después del desarrollo:

* **Análisis de documentación:** Se obtiene información sobre los requerimientos funcionales y no funcionales a partir de documentación existente. Debemos incluir reportes de errores y manuales de usuario. Es útil cuando los expertos del sistema no están disponibles para guiarnos.
* **Observación:** Implica estudiar el entorno de trabajo de los usuarios, clientes e interesados del proyecto (Stakeholders). Se toman notas sobre eventos, acciones y experiencias usuarias con todos los factores incidentes. Es una técnica útil cuando se está documentando la situación actual de procesos de negocio.
* **Entrevistas:** Se realizan con los usuarios y Product Owner. Se dirige al usuario hacia aspectos específicos del requerimiento en cuestión, luego de haber manifestado un problema o una idea. El éxito de las entrevistas puede depender del grado de conocimiento entre las partes, disposición del entrevistado de suministrar información, buena documentación y buena relación entre las partes. Algunos tips:
  + Mantente enfocado en los objetivos de la entrevista.
  + Las preguntas abiertas son útiles para identificar información faltante.
  + Las preguntas cerradas son útiles para confirmar y validar información.
* **Encuestas:** Es una técnica útil para recopilar eficientemente los requerimientos de muchas personas. La clave para el éxito es que tengan un propósito y audiencia claramente definida, establecer fechas topes para llenar la encuesta, con preguntas claras y concisas.

## **Desarrollo tradicional VS. Desarrollo ágil**

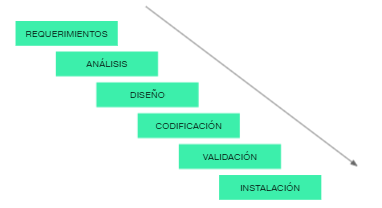
### **DESARROLLO TRADICIONAL (DESARROLLO CASCADA)**

Cada etapa comienza cuando termina la anterior. Esta metodología no contempla la posibilidad de medir los resultados hasta el final del proyecto por dos motivos:

* Porque recién allí dispondrá de un entregable
* Porque no se considera necesaria una medición antes del lanzamiento.

Los proyectos son divididos en una secuencia de tareas agrupadas en fases progresivas.

Cada fase necesita ser completada cumpliendo ciertos parámetros para poder avanzar a la siguiente. Deviene del famoso modelo fordista.



Las etapas son:

* **Requerimientos**: Son reunidos al inicio del proyecto para plantear cada fase sin necesidad de volver a realizar el levantamiento de información.
* **Análisis**: Es la interpretación y análisis sobre el documento de requerimientos para indicar qué cambios necesita. El desarrollo inicia recién cuando el documento de requerimientos está completamente terminado con todos los cambios.
* **Diseño**: En base al resultado anterior, se genera un diseño visual y de arquitectura de software para poder construir el producto.
* **Codificación**: Implica escribir el código del producto en base a toda la documentación anterior. Ingeniería trabaja aislado del/la PM.
* **Validación**: Es hora de validar con el negocio el resultado obtenido. El cliente no interviene en el testeo. Generalmente, se siguen muchas modificaciones hasta alcanzar un resultado que satisfaga a todos los integrantes del proyecto.
* **Instalación**: Es el lanzamiento propiamente dicho. Aquí se da el primer contacto del cliente con el producto.

Quizá la mayor desventaja del desarrollo tradicional sea el riesgo. Si una etapa se extiende demasiado o el producto no funciona con los clientes, solucionarlo supondrá mucho tiempo, y con ello mayores costos.

### **DESARROLLO ÁGIL**

Posee tantas instancias de relevamiento como iteraciones y refinamientos. En cada iteración se podrá revisar lo relevado en la anterior, a partir de los hallazgos en las pruebas realizadas 🚀.

‘Agile’ es mucho más que una metodología rápida y flexible. Es una filosofía que supone una forma distinta de trabajar y organizarse, donde cada objetivo se divide en pequeñas partes, las cuales tienen que completarse y entregarse en pocas semanas. **Iteración** es el recorrido de un tiempo construyendo un producto. Suele ser de aproximadamente 2 semanas. Esto es, cada por ejemplo 2 semanas, voy a ver con mi equipo por qué camino le conviene al usuario que siga; podría tener una planificación anual, pero si me di cuenta a las 2 semanas que esta planificación cambió según las necesidades de las personas, es correcto cambiarla.

**Algunas ventajas del desarrollo ágil:**

* Calidad: Minimiza los errores en los entregables y mejora la experiencia y las funcionalidades para el cliente.
* Rapidez: Acorta los ciclos de producción y minimiza los tiempos de reacción, así como la toma de decisiones.
* Compromiso: Mejora la satisfacción de las personas que integran el squad de producto y genera conciencia de equipo.
* Productividad: Al asignar mejor los recursos y de forma más dinámica, mejora la producción según las prioridades que tenga la empresa.

**Algunas desventajas del desarrollo ágil:**

* Planificación: Es complicado predecir los costos y tiempos finales que el desarrollo tendrá, entre otros factores involucrados.
* Documentación: Es limitada. La documentación no es prioritaria por lo que puede terminar siendo poco detallada y en algunos casos inexistente.
* Resultados: A pesar de lograr que la producción sea más rápida e incremental los resultados también se pueden percibir incoherentes.
* Producto final: La naturaleza iterativa de Agile puede significar que se tenga una idea poco clara de lo que el producto final podría ser.

Agile es una filosofía y existen varias formas de implementarla. De allí vienen las conocidas metodologías ágiles:

* Scrum
* Kanban
* LESS
* XP

### **INTRODUCCIÓN A SCRUM**

Scrum es una metodología ágil basada en equipos auto-organizados y multifuncionales (squads) que buscan entregar incrementos en funcionalidades en ciclos de desarrollo llamados “sprints” (que duran entre 2 y 4 semanas) 🤝. El objetivo es minimizar los riesgos durante la realización de un proyecto, pero de manera colaborativa.

*“Básicamente, el Scrum se basa en una idea sencilla: cada vez que iniciamos un proyecto, ¿por qué no comprobamos cómo va cada cierto tiempo, vemos si lo que estamos haciendo apunta en la buena dirección y si es lo que la gente realmente quiere? ¿Y por qué no comprobar si existen maneras de mejorar lo que estamos haciendo, de hacerlo mejor y más rápido, y qué es lo que puede estar impidiendo que sea así?”*. Jeff Sutherland “Scrum”.

### **INTRODUCCIÓN A KANBAN**

Kanban es otra metodología ágil basada en pocos momentos fundamentales. Se elabora un cuadro o diagrama, en el que se reflejan estados básicos, por ejemplo: pendientes, en proceso o finalizadas. Usamos Tablero Kanban donde representamos visualmente el proyecto permitiendo a todo el Squad saber en qué estado se encuentra cada ítem.

Un ejemplo básico de un tablero de desarrollo puede tener columnas que asignen diferentes momentos: “En backlog”, “Programando”, “En testeo”, “Listo”. Los ítems irán moviéndose en el tablero conforme se vaya cumpliendo cada etapa.

### **INTRODUCCIÓN A SCRUM**

Los proyectos Scrum y Kanban suelen tener similitudes con respecto a las columnas. Sin embargo, a los proyectos Kanban se les suele agregar una columna al principio llamada “Backlog” donde tendremos todos los requerimientos que se vayan creando. Son distintas estrategias para implementar desarrollo Ágil. Kanban es más fluido y continuo, mientras que Scrum está basado en Sprints cortos y estructurados.

La diferencia con un proyecto Scrum es que se puede iniciar un Sprint y dentro del mismo no se verán todos los ítems del Backlog. El Sprint es un compromiso de alcance que mantenemos con el cliente: “Vamos a realizar todo este trabajo en X periodo de tiempo”. Otra diferencia son los reportes:

* Al trabajar sprint a sprint, podemos obtener los llamados “Sprint Metrics”, algunos ejemplos podría ser la Velocidad del Equipo. (¿Cuántos Story Points se resuelven por Sprint?)
* Analizar estadísticas de este estilo nos ayuda a entender a nuestro equipo de trabajo, por ejemplo para analizar Por qué pasamos de tener 50 Story Points por Sprint a 32.

**¡Atención! Información importante para la próxima clase:** La próxima clase trabajaremos tableros con Asana. Les recomendamos descargarlo y/o registrarse en <https://asana.com/es> antes de iniciar la clase para poder realizar la actividad sin problemas 🤗 Nota importante: ¡Utilizaremos la opción Gratuita!

Para el desafío entregable de la próxima clase, deberás configurar tus requerimientos a partir de user stories y épicas (conceptos que veremos próximamente) a través de encuestas realizadas a tus potenciales usuarios. Te recomendamos iniciar el research esta semana con el objetivo de:

* Validar y re-ajustar lo trabajado hasta el momento con perspectiva de Customer Centricity: problema, objetivo, solución, pov y arquetipos de usuario y key features.
* Identificar los requerimientos de usuarios.
* Puedes ayudarte con estas buenas prácticas para realizar encuestas.

**Resumen de lo visto en la clase de hoy:**

* Requerimientos de negocio, funcionales y de usuario.
* Técnicas de relevamiento.
* Desarrollo tradicional y desarrollo ágil.

#DemocratizandoLaEducación