**Scrum**

Agilidad

El entorno de trabajo de las empresas del conocimiento se parece muy poco al que originó la gestión de proyectos predictiva. Ahora se necesitan estrategias para el lanzamiento de productos orientadas a la entrega temprana de resultados tangibles, y a la respuesta ágil y flexible, necesaria para trabajar en mercados de evolución rápida.

Ahora se construye el producto al mismo tiempo que se modifican e introducen nuevos requisitos. El cliente parte de una visión medianamente clara, pero el nivel de innovación que requiere, y la velocidad a la que se mueve el entorno de su negocio, no le permiten prever con detalle cómo será el resultado final.

Quizá ya no hay “productos finales”, sino productos en continua evolución y mejora.

***La gestión de proyectos ágil no se formula sobre la necesidad de anticipación, sino sobre la de adaptación continua.***

Hay clientes que necesitan disponer de una primera versión con funcionalidades mínimas en cuestión de semanas, y no un producto completo dentro de uno o dos años. Clientes cuyo interés es poner en el mercado rápidamente un concepto nuevo, y desarrollar de forma continua su valor.

Hay proyectos que no necesitan gestionar el seguimiento de un plan, y cuyo fracaso puede ser la consecuencia de un modelo de gestión inapropiado.

La mayoría de los fiascos se producen por aplicar ingeniería secuencial y gestión predictiva tanto en el proceso de adquisición (requisitos, contratación, seguimiento y entrega) como en la gestión del proyecto, en productos que no necesitan tanto garantías de previsibilidad en la ejecución, como respuesta rápida y flexibilidad para funcionar en entornos de negocio que cambian y evolucionan rápidamente.

Manifiesto Ágil (Valores)

* Valorar a los individuos por encima de los procesos.

-Este es el postulado más importante del manifiesto.

Por supuesto que los procesos ayudan al trabajo. Son una guía de operación. Las herramientas mejoran la eficiencia, pero hay tareas que requieren talento y necesitan personas que lo aporten y trabajen con una actitud adecuada.

La producción basada en procesos persigue que la calidad del resultado sea consecuencia del *know-how* “explicitado” en los procesos, más que en el conocimiento aportado por las personas que los ejecutan. Sin embargo, en el desarrollo ágil los procesos son una ayuda. Un soporte para guiar el trabajo. La defensa a ultranza de los procesos lleva a afirmar que con ellos se pueden conseguir resultados extraordinarios con personas mediocres, y lo cierto es que este principio no es cierto cuando se necesita creatividad e innovación.

* El software que funciona por encima de la documentación exhaustiva.

-Poder anticipar cómo será el funcionamiento del producto final, observando prototipos previos, o partes ya elaboradas ofrece un “*feedback*” estimulante y enriquecedor, que genera ideas imposibles de concebir en un primer momento, y que difícilmente se podrían incluir al redactar un documento de requisitos detallado en el comienzo del proyecto.

El manifiesto ágil no considera inútil la documentación, sólo la innecesaria. Los documentos son soporte de hechos, permiten la transferencia del conocimiento, registran información histórica, y en muchas cuestiones legales o normativas son obligatorios, pero su relevancia debe ser mucho menor que el producto final.

A la comunicación a través de documentos le falta la riqueza y producción de valor que logra la comunicación directa entre las personas y a través de la interacción con prototipos del producto.

Por eso, siempre que sea posible se debe reducir al mínimo indispensable el uso de documentación que consume trabajo sin aportar un valor directo al producto.

Si la organización y los equipos se comunican a través de documentos, además de ocultar la riqueza de la interacción con el producto, forman barreras de burocracia entre departamentos o entre personas.

* La colaboración del cliente por encima de la negociación contractual.

-Las prácticas ágiles están indicadas para productos de evolución continua. En este tipo de productos no es posible definir en un documento de requisitos cerrado cómo debería ser el producto final, y resulta más apropiado tomar feedback de forma continua, y en paralelo al desarrollo del producto redefinir y mejorar en consecuencia los propios requisitos de las partes aún no desarrolladas.

El objetivo de un proyecto ágil no es controlar la ejecución para garantizar el cumplimiento de la planificación, sino proporcionar de forma continua el mayor valor posible al producto.

Resulta por tanto más adecuada una relación de implicación y colaboración continua con el cliente, que una contractual de delimitación de responsabilidades.

* La respuesta al cambio por encima del seguimiento de un plan.

-Para desarrollar productos de requisitos inestables, en los que es inherente el cambio y la evolución rápida y continua, resulta mucho más valiosa la capacidad de respuesta que la de seguimiento y aseguramiento de planes. Los principales valores de la gestión ágil son la anticipación y la adaptación, diferentes a los de la gestión de proyectos ortodoxa: planificación y control para garantizar el cumplimiento del plan.

Scrum

* Es un enfoque ágil para la gestión de un proyecto. Más que una metodología o proceso, es un **Marco de Trabajo**.
* Utiliza *procesos interactivos/incrementales*.
* Orientado a resultados y compromisos.
* No está restringido a proyectos de software solamente.
* Su visión es opuesta a la propuesta por la metodología en cascada.

Pilares

* El ciclo de vida iterativo e incremental (Sprint)
* La comunicación.

- “La forma más eficiente y efectiva de comunicar información de ida y vuelta dentro de un equipo de desarrollo es mediante la comunicación cara a cara”

* Transparencia

-Aspectos significativos del proceso deben ser visibles

-Definidos en base a un estándar común

* Inspección

-Deben inspeccionar frecuentemente los Artefactos de Scrum y el progreso hacia un objetivo para detectar variaciones indeseadas.

* Adaptación

-Si se determina que uno o más aspectos de un proceso se desvían de los límites aceptables y que el producto resultante será inaceptable, el proceso o el material que está siendo procesado deben ajustarse. Dicho ajuste deberá realizarse cuanto antes para minimizar desviaciones mayores.

Sprint

El Sprint es un período de corta duración que debe finalizar con un prototipo operativo o producto parcialmente entregable. El mismo se repite n veces a lo largo del proyecto y permite hacer entregas de producto en partes, donde cada entrega, es un incremento de funcionalidad respecto al anterior. Esto difiere del conocido ciclo de vida en cascada muy utilizado en el campo de desarrollo de software en que, las fases del ciclo de vida (requisitos, análisis, diseño, codificación, testing, etc.) se realizan una única vez y, el inicio de cada fase no comienza hasta que termina la que precede.

Durante el Sprint:

• No se realizan cambios que puedan afectar al objetivo del Sprint (*Sprint Goal*);

• Los objetivos de calidad no disminuyen;

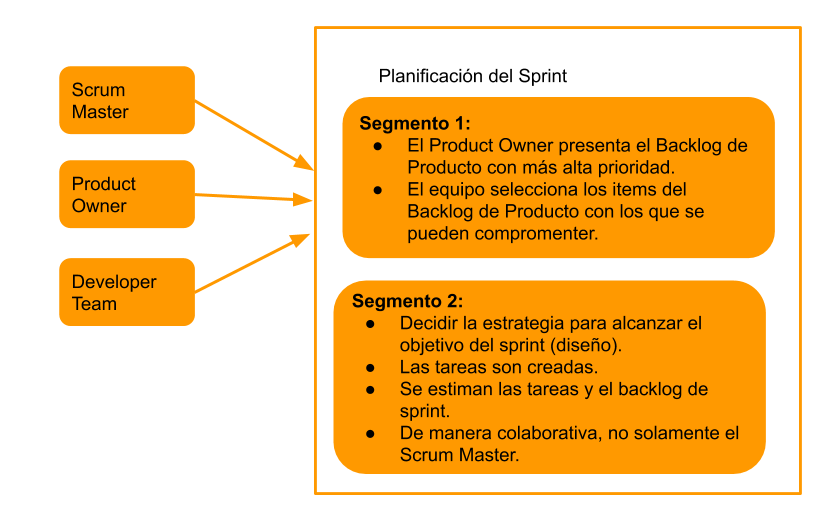
• El alcance puede clarificarse y renegociarse entre el Propietario del Producto (*Product Owner*) y el Equipo de Desarrollo a medida que se va aprendiendo más.

Ceremonias que se realizan a lo largo del proyecto:

* Sprint Planning Meeting
* Daily Scrum Meeting
* Sprint Review Meeting
* Sprint Retrospective Meeting

Planificación del Sprint

*(Sprint Planning)*



Daily Scrum *(Scrum Diario)*

1 ¿Qué hiciste ayer?

2 ¿Qué vas a hacer hoy?

3 ¿Hay algo que te impida hacer tu trabajo?

-No son reuniones de reporte de estado al Scrum Master.

-Son compromisos frente a los demás miembros del equipo.

Sprint Review

* El equipo presenta la funcionalidad terminada al Product Owner y demás stakeholders.
* Los miembros del equipo responden preguntas de los stakeholders en relación a la demostración, y toman nota de los cambios propuestos.
* Al finalizar la presentación, los stackholders dan su impresión acerca del producto, cambios deseados y prioridad de esos cambios.
* No es una reunión de seguimiento, la presentación del incremento tiene por objeto fomentar la retroalimentación.

Sprint Retrospective

* Provee una visión de qué está funcionando y qué no está funcionando.
* Se realiza al finalizar el sprint.
* Participa todo el equipo.

Los Artefactos en Scrum

*(Scrum Artifacts)*

*-Pila del Producto (Product Backlog)*

La **pila del producto (product backlog)** es la lista ordenada de todo aquello que el propietario de producto cree que necesita el producto.

Es el inventario de funcionalidades, mejoras, tecnología y corrección de errores que deben incorporarse al producto a través de los sucesivos sprints. Representa todo aquello que esperan el cliente, los usuarios, y en general los interesados. Todo lo que suponga un trabajo que debe realizar el equipo debe estar reflejado en esta pila.

La pila del producto nunca se da por completada; está en continuo crecimiento y evolución. Al comenzar el proyecto incluye los requisitos inicialmente conocidos y mejor entendidos, y evoluciona conforme avanza el desarrollo.

Gracias a su carácter dinámico refleja aquello que el producto necesita incorporar para adecuarse a las circunstancias, en todo momento.

*-Pila del Sprint (Sprint Backlog)*

La pila del sprint (*sprint Backlog*) es la lista de las tareas necesarias para construir las historias de usuario que se van a realizar en un sprint.

La confecciona el equipo en la reunión de planificación del sprint, indicando para cada tarea el esfuerzo previsto para realizarla. Para calcular el esfuerzo de cada tarea es habitual emplear técnicas de estimación.

La pila del sprint descompone las historias de usuario en unidades de tamaño adecuado para monitorizar el avance a diario, e identificar riesgos y problemas sin necesidad de procesos de gestión complejos.

Es también una herramienta para la comunicación visual directa del equipo.

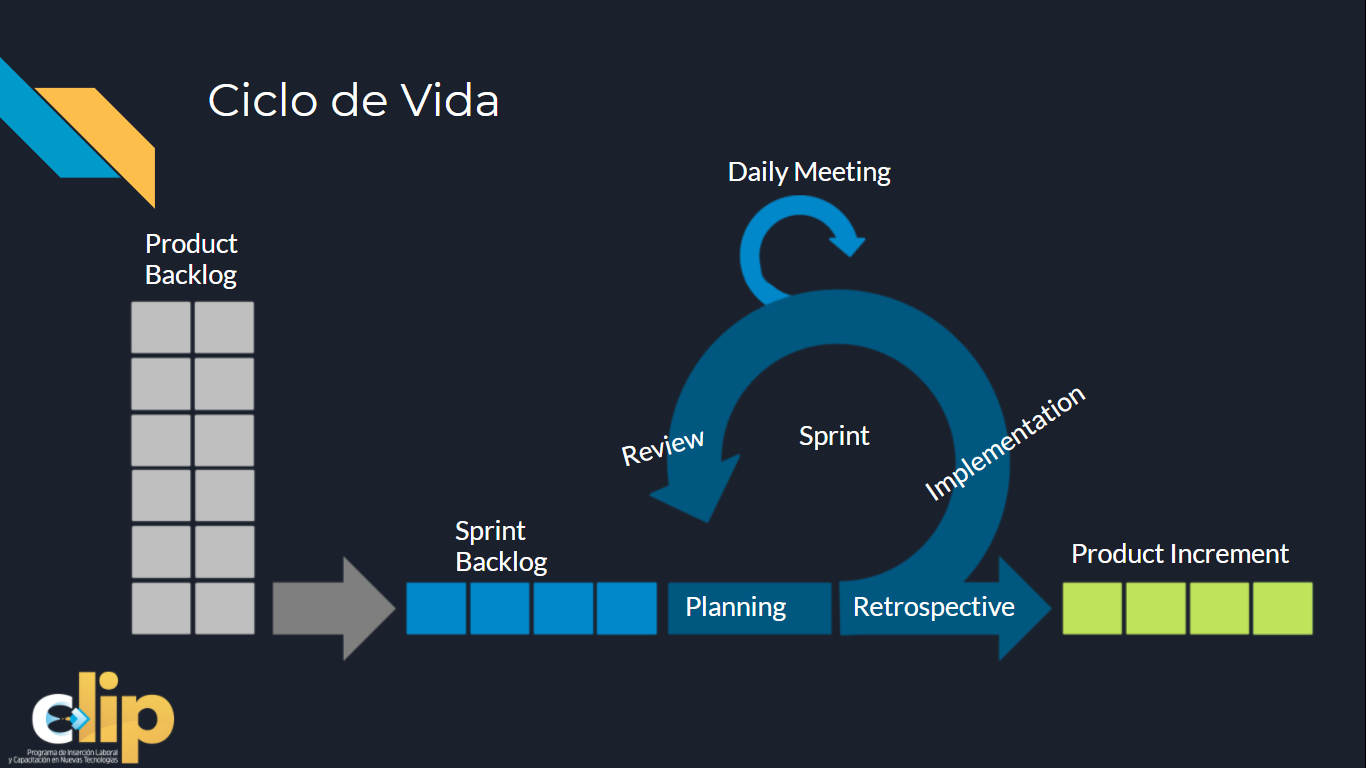
*Incremento (Increment)*

El incremento es la parte de producto producida en un sprint, y tiene como característica el estar completamente terminada y operativa, en condiciones de ser entregada al cliente.

No se deben considerar como Incremento a prototipos, módulos o sub-módulos, ni partes pendientes de pruebas o integración.

Si el proyecto o el sistema requiere documentación, o procesos de validación y verificación documentados, o procesos con niveles de independencia que implican procesos con terceros, éstos también tienen que estar realizados para considerar que el incremento está “hecho”.

*Ciclo de vida*



* Roles
  + Equipo Scrum

La unidad fundamental de Scrum es un pequeño equipo de personas, un equipo Scrum. El equipo Scrum consta de un Scrum Master, un propietario de producto (Product Owner) y desarrolladores. Dentro de un equipo de Scrum, no hay sub-equipos ni jerarquías. Es una unidad cohesionada de profesionales enfocada en un objetivo a la vez, el objetivo del producto.

Los equipos de Scrum son multifuncionales, lo que significa que los miembros tienen todas las habilidades necesarias para crear valor en cada Sprint. También son autogestionados, lo que significa que internamente deciden quién hace qué, cuándo y cómo.

El equipo de Scrum es lo suficientemente pequeño como para permanecer ágil y lo suficientemente grande como para completar un trabajo significativo dentro de un Sprint, por lo general 10 o menos personas. En general, hemos descubierto que los equipos más pequeños se comunican mejor y son más productivos. El equipo Scrum es responsable de todas las actividades relacionadas con los productos, desde la colaboración, verificación, mantenimiento, operación, experimentación, investigación y desarrollo, y cualquier otra cosa que pueda ser necesaria.

Scrum Master

El Scrum Master es responsable de establecer Scrum tal como se define en la Guía de Scrum. Lo consigue ayudando a todos a comprender la teoría y la práctica de Scrum, tanto dentro del Equipo como en toda la organización.

El Scrum Master es responsable de la efectividad del Scrum Team. Lo logra al permitir que el equipo Scrum mejore sus prácticas, dentro del marco de Scrum.

Los Scrum Masters son verdaderos líderes que sirven al equipo Scrum y a toda la organización.

Development Team

Los desarrolladores son las personas del equipo Scrum que se comprometen a crear cualquier aspecto útil (funcional) de un Incremento en cada Sprint.

Las habilidades específicas que necesitan los desarrolladores son a menudo amplias y variarán con el dominio del trabajo. Sin embargo, los desarrolladores siempre son responsables de:

● Crear un plan para el Sprint, el Sprint Backlog;

● Inculcar la calidad adhiriéndose a una definición de Hecho;

● Adaptar su plan cada día hacia el Objetivo Sprint;

● Responsabilizarse mutuamente como profesionales.

Product owner

El Propietario del Producto es responsable de maximizar el valor del producto resultante del trabajo del equipo de Scrum. La forma en que se hace esto puede variar ampliamente entre organizaciones, equipos Scrum e individuos.

El Propietario del Producto también es responsable de la gestión eficaz de la pila del producto (Product Backlog).

El Propietario del Producto puede hacer el trabajo anterior o puede delegar la responsabilidad a otros. En cualquier caso, el propietario del producto sigue siendo responsable.

Para que los Propietarios de Productos tengan éxito, toda la organización debe respetar sus decisiones. Estas decisiones son visibles en el contenido y el orden del trabajo pendiente del producto, y a través del Incremento inspeccionable en la revisión de Sprint.

El Propietario del Producto es una persona, no un comité. El Propietario del Producto puede representar las necesidades de muchas partes interesadas en el trabajo pendiente del producto. Aquellos que deseen cambiar el trabajo pendiente del producto pueden hacerlo tratando de negociar con criterio con el Product Owner.

Requerimientos

Requerimiento: es una petición de algo que se considere necesario.

Requisito: es una condición necesaria a cumplir.

Requisito=requerimiento

Especificación: documento que define de forma completa, precisa y verificable los requisitos, el diseño y el comportamiento u otras características de un sistema o componente de un sistema.

Especificación de un requerimiento:

● Declaración abstracta, de alto nivel, de un servicio que debe proporcionar el sistema o una restricción del mismo

* Requerimientos Funcionales

Declaraciones de los servicios que debe proporcionar el sistema

Cómo debe reaccionar el sistema a entradas particulares

Cómo debe comportarse el sistema en determinadas situaciones

* Requerimientos No Funcionales

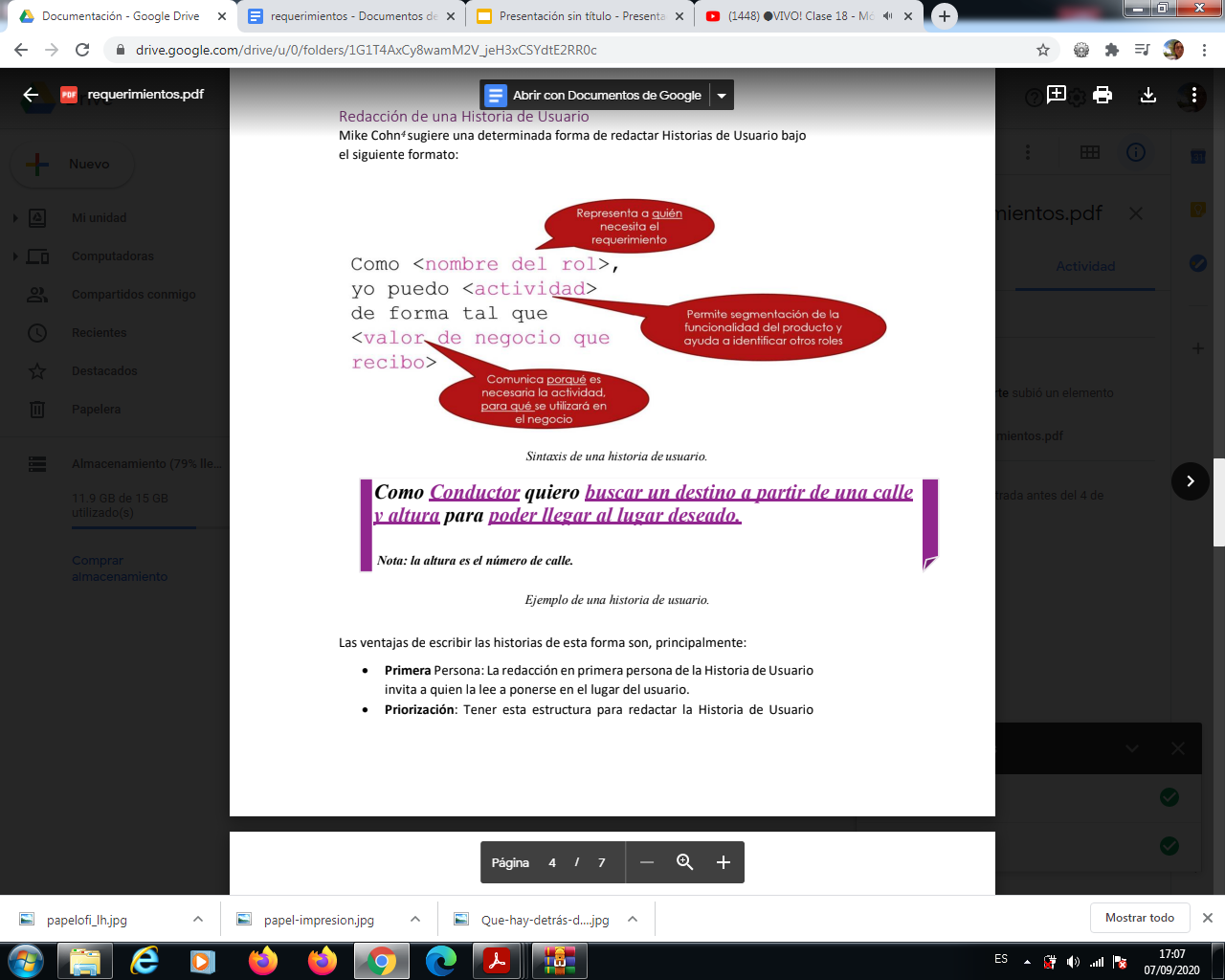
Restricciones sobre los servicios o funciones ofrecidas por el sistema, tales como limitaciones de tiempo, restricciones en el proceso de desarrollo, estándares, etc

Historias de Usuario

Una **historia de usuario** (user story) es una descripción corta de una funcionalidad, validada por un usuario o cliente de un sistema.

**Importancia:**

* La historia de usuario debe estar expresada en un lenguaje que el usuario pueda entender y que refleja una descripción sintetizada de lo que el usuario desea.
* Facilitar la comunicación.
* Eliminar en lo posible ambigüedades, pero no las dudas.
* Eliminar en lo posible las malinterpretaciones. Siempre hay que preguntar. Nunca asumir cosas de las que no estoy seguro.



Las ventajas de escribir las historias de esta forma son, principalmente:

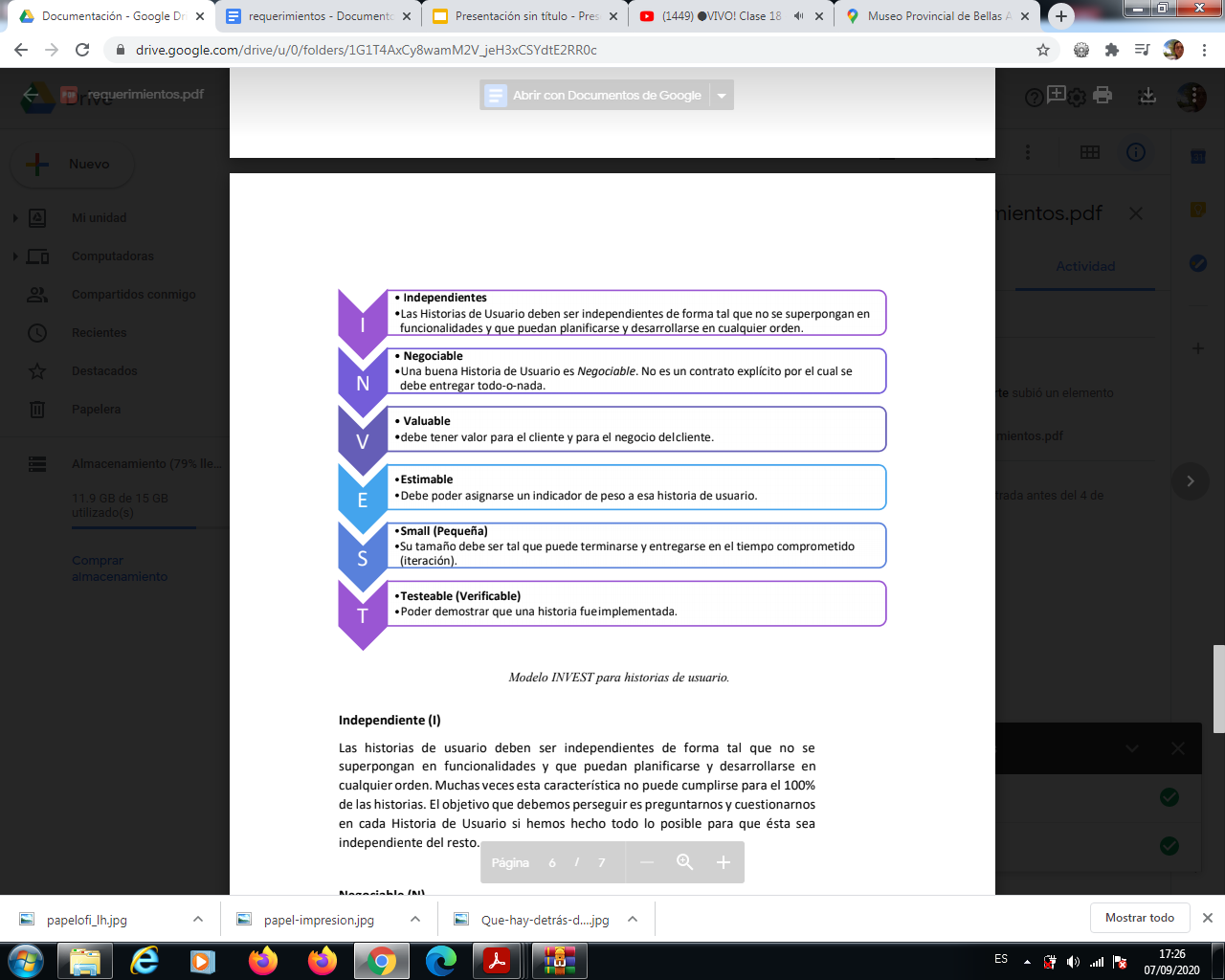
• **Primera Persona**: La redacción en primera persona de la Historia de Usuario invita a quien la lee a ponerse en el lugar del usuario.

• **Priorización**: Tener esta estructura para redactar la Historia de Usuario ayuda al responsable del producto a priorizar. Le permite visualizar mejor cuál es la funcionalidad, quién se beneficia y cuál es el valor de la misma.

• **Propósito:** Conocer el propósito de una funcionalidad permite al equipo de desarrollo plantear alternativas que cumplan con el mismo objetivo en el caso de que el costo de la funcionalidad solicitada sea alto o su construcción no sea viable.

INVEST - características de la historia de usuario

El modelo INVEST es un conjunto de características recomendadas para evaluar la calidad de una historia de usuario:



**GitHub**

Acerca del Control de Versiones

¿Qué es un control de versiones, y por qué debería importarte? Un control de versiones es un sistema que registra los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante.

Si eres diseñador gráfico o de web y quieres mantener cada versión de una imagen o diseño (es algo que sin duda vas a querer), usar un sistema de control de versiones (VCS por sus siglas en inglés) es una decisión muy acertada. Dicho sistema te permite regresar a versiones anteriores de tus archivos, regresar a una versión anterior del proyecto completo, comparar cambios a lo largo del tiempo, ver quién modificó por última vez algo que pueda estar causando problemas, ver quién introdujo un problema y cuándo, y mucho más. Usar un VCS también significa generalmente que, si arruinas o pierdes archivos, será posible recuperarlos fácilmente.

Los Tres Estados

Git tiene tres estados principales en los que se pueden encontrar tus archivos: confirmado (committed), modificado (modified), y preparado (staged). Confirmado: significa que los datos están almacenados de manera segura en tu base de datos local. Modificado: significa que has modificado el archivo, pero todavía no lo has confirmado a tu base de datos. Preparado: significa que has marcado un archivo modificado en su versión actual para que vaya en tu próxima confirmación.

Esto nos lleva a las tres secciones principales de un proyecto de Git: El directorio de Git (Git directory), el directorio de trabajo (working directory), y el área de preparación (staging area).



El directorio de Git es donde se almacenan los metadatos y la base de datos de objetos para tu proyecto. Es la parte más importante de Git, y es lo que se copia cuando se clona un repositorio desde otra computadora.

El directorio de trabajo es una copia de una versión del proyecto. Estos archivos se sacan de la base de datos comprimida en el directorio de Git, y se colocan en disco para que los puedas usar o modificar.

El área de preparación es un archivo, generalmente contenido en tu directorio de Git, que almacena información acerca de lo que va a ir en tu próxima confirmación.

El flujo de trabajo básico en Git es algo así:

● Modificas una serie de archivos en tu directorio de trabajo.

● Preparas los archivos, añadiéndolos a tu área de preparación.

● Confirmas los cambios, lo que toma los archivos tal y como están en el área de preparación y almacena esa copia instantánea de manera permanente en tu directorio de Git.

Si una versión concreta de un archivo está en el directorio de Git, se considera confirmada (committed). Si ha sufrido cambios desde que se obtuvo del repositorio, pero ha sido añadida al área de preparación, está preparada (staged). Y si ha sufrido cambios desde que se obtuvo del repositorio, pero no se ha preparado, está modificada (modified).

GIT

Git es un Sistema de control de Versiones DISTRIBUIDO

Ventajas

* **Sistema distribuido**, que permite el trabajo incluso sin conexión.
* **Super rápido y ligero**, optimizado para hacer operaciones de control muy rápidas.
* **Crear ramas y mezclarlas** poco propenso a problemas.
* **La integridad de la información está asegurada** gracias a su modelo de almacenamiento, que permite predecir este tipo de problemas.
* **Permite flujos de trabajo muy flexibles**.
* **El concepto de área de preparación** o *staging* permite versionar los cambios como nos convenga, no todo o nada.
* Operaciones locales
* Copias instantáneas

**Figma**

Es un editor de gráficos vectoriales y una herramienta de creación de prototipos que se basa principalmente en la web.

Características

* Todo está en la nube: **e**stá basado en el navegador. Lo que significa que tus archivos estarán siempre accesibles y podrás evitar perder tiempo (e invertir dinero) en servicios de almacenamiento como Google Drive y Dropbox.
* Tienes también un historial de cambios de cada archivo, por lo que siempre podrás volver a algún punto anterior. Y todo lo que borres está accesible en Deleted *Files*.
* Permite **trabajar colaborativamente** en un archivo. Esto soluciona algunos problemas que se dan en equipos de diseño, cuando dos diseñadores quieren trabajar a la vez en un solo archivo. Solo queda una opción, duplicarlo y después “juntarlo” todo (o dejarlo en dos archivos y arriesgarse a perder parte del proyecto).
* Puede utilizarse en **Windows** (y cualquier otro sistema operativo).
* **Ágil y disponible en todas partes**. Los archivos de Figma son accesibles desde cualquier ordenador y, lo que es mejor, no hace falta que descargues nada. Simplemente inicias sesión y ya lo tienes todo allí.