1)El framework Scrum es una metodología ágil para la gestión de proyectos, especialmente en el desarrollo de software. [Se basa en un pequeño equipo de personas, el Scrum Team, que se compone principalmente por tres roles](https://www.itdo.com/blog/framework-scrum-roles-y-responsabilidades/)[1](https://www.itdo.com/blog/framework-scrum-roles-y-responsabilidades/):

1. [**Product Owner**: Es el encargado de optimizar y maximizar el valor del producto, gestionando el flujo de valor del producto a través del Product Backlog](https://www.itdo.com/blog/framework-scrum-roles-y-responsabilidades/)[2](https://www2.deloitte.com/es/es/pages/technology/articles/roles-y-responsabilidades-scrum.html). [También actúa como interlocutor con los stakeholders y sponsors del proyecto, y como altavoz de las peticiones y requerimientos de los clientes](https://www.itdo.com/blog/framework-scrum-roles-y-responsabilidades/)[2](https://www2.deloitte.com/es/es/pages/technology/articles/roles-y-responsabilidades-scrum.html).
2. [**Scrum Master**: Tiene dos funciones principales dentro del marco de trabajo: gestionar el proceso Scrum y ayudar a eliminar impedimentos que puedan afectar a la entrega del producto](https://www.itdo.com/blog/framework-scrum-roles-y-responsabilidades/)[2](https://www2.deloitte.com/es/es/pages/technology/articles/roles-y-responsabilidades-scrum.html). [Además, se encarga de las labores de mentoring y formación, coaching y de facilitar reuniones y eventos si es necesario](https://www.itdo.com/blog/framework-scrum-roles-y-responsabilidades/)[2](https://www2.deloitte.com/es/es/pages/technology/articles/roles-y-responsabilidades-scrum.html).
3. [**Equipo de Desarrollo**: No hay jerarquía en el equipo de desarrollo](https://www.itdo.com/blog/framework-scrum-roles-y-responsabilidades/)[1](https://www.itdo.com/blog/framework-scrum-roles-y-responsabilidades/). [El objetivo es el resultado y tener un equipo multifuncional, que aunque en los inicios tengan que formarse en áreas que no sean de su conocimiento, en el futuro dispondrás de un equipo multidisciplinar que podrá afrontar cualquier necesidad](https://www.itdo.com/blog/framework-scrum-roles-y-responsabilidades/)[1](https://www.itdo.com/blog/framework-scrum-roles-y-responsabilidades/).

[Estos roles trabajan juntos en ciclos de trabajo llamados Sprints, que suelen durar entre una y cuatro semanas, para entregar incrementos de producto de alto valor y calidad](https://www.itdo.com/blog/framework-scrum-roles-y-responsabilidades/)[2](https://www2.deloitte.com/es/es/pages/technology/articles/roles-y-responsabilidades-scrum.html).

2) Product Backlog (Lista de Producto): Este es un documento clave en Scrum que enumera todas las características, funcionalidades y mejoras que se desean para el producto. En el contexto de la arquitectura de la computadora, esto podría incluir requisitos relacionados con la infraestructura, la seguridad, el rendimiento y otros aspectos arquitectónicos.

Definición de Hecho: Este documento define los criterios que deben cumplirse para que una tarea o una historia de usuario se considere completada. Puede incluir estándares específicos relacionados con la arquitectura, como pruebas de rendimiento, revisión de código, documentación arquitectónica, etc.

Arquitectura de Referencia: Este es un documento que describe la estructura y los componentes clave de la arquitectura de la computadora que se utilizarán en el proyecto. Puede incluir diagramas de arquitectura, patrones de diseño, estándares de codificación, etc.

Documentación Técnica: En proyectos de arquitectura de computadoras, es esencial tener una documentación técnica sólida que describa la infraestructura, la topología de red, las decisiones de diseño, los diagramas de flujo de datos, etc. Esta documentación es fundamental para que el equipo comprenda y mantengo la arquitectura.

Informe de Revisión de Arquitectura: Durante las reuniones de revisión de sprint, se pueden presentar informes sobre el estado de la arquitectura, los problemas identificados y las decisiones tomadas. Esto ayuda a mantener a todos los miembros del equipo al tanto de los aspectos arquitectónicos del proyecto.

Especificaciones Técnicas: Pueden ser necesarias especificaciones técnicas detalladas para componentes o módulos clave de la arquitectura. Estas especificaciones pueden incluir detalles sobre la interfaz de programación de aplicaciones (API), protocolos de comunicación, algoritmos utilizados, etc.

Diagramas de Arquitectura: Los diagramas, como diagramas de flujo, diagramas de componentes, diagramas de despliegue, son útiles para visualizar y comunicar la arquitectura del sistema.

Estos documentos y artefactos se utilizan en Scrum para garantizar que el desarrollo de software se realice de manera planificada y controlada, incluso en proyectos relacionados con la arquitectura de la computadora. El Product Owner, Scrum Master y el equipo de desarrollo colaboran para definir y mantener estos documentos a lo largo del proyecto y asegurar de que la arquitectura evolucione de manera adecuada.

3) En Scrum, existen varias reuniones principales con propósitos específicos. Estas reuniones están diseñadas para facilitar la comunicación, la colaboración y la toma de decisiones en un equipo ágil. Las reuniones principales de Scrum son las siguientes:

1)Planificación del Sprint (Planificación del Sprint) :

Propósito: Establecer el plan detallado para el próximo Sprint, seleccionando los elementos del Product Backlog que se incluirán en el Sprint y definiendo cómo se implementarán.

Quiénes asisten: El equipo de desarrollo, el Scrum Master y el Product Owner.

2)Scrum Diario (Scrum Diario) :

Propósito: Facilitar la sincronización diaria del equipo, revisar el progreso hacia el objetivo del Sprint y colaborar en la resolución de obstáculos.

Quiénes asisten: El equipo de desarrollo.

3)Revisión del Sprint (Revisión del Sprint) :

Propósito: Revisar y demostrar el trabajo completado durante el Sprint a las partes interesadas (stakeholders) y obtener retroalimentación para adaptar el Product Backlog.

Quiénes asisten: El equipo de desarrollo, el Scrum Master, el Product Owner y los stakeholders.

Estas reuniones son fundamentales para el funcionamiento efectivo de Scrum, y cada una cumple un papel específico en el proceso de desarrollo ágil. Además de estas reuniones principales, el equipo puede realizar reuniones adicionales según sea necesario para abordar temas específicos o desafíos que surjan durante el proyecto.

4)¿qué es [Git](https://git-scm.com/)? La definición más sencilla es git es un software que me permite controlar las versiones.

Una forma más clara de explicarlo es a través de videojuegos. Cuando estás jugando, sabes que puedes guardar tu progreso en diferentes zonas de la partida, así no perderás nada en caso de morir y podrás reanudar todo desde el punto de guardado.

Así funciona Git, pero, a diferencia de los videojuegos, en vez de guardar el progreso de una partida, git es un software que me permite **guarda un desarrollo, ya sea de proyectos de software o incluso un fragmento de código como un Script.**

Nos permite guardar cualquier archivo que estemos trabajando. Git se concentra netamente en el contenido de los archivos, evitando que se armen confusiones en el almacenamiento y el historial del árbol de archivos a la hora de nombrarlos, cosa que, en muchos casos, termina desperdiciando tiempo para los mismos desarrolladores. La principal prioridad de Git es conservar cualquier código fuente en el que se esté trabajando o esté en gestión, tales aspectos como los archivos, su contenido, los directorios, las versiones, las etiquetas y las confirmaciones. Todo esto se ubica dentro del repositorio de Git y es protegido por diferentes algoritmos que salvaguardan el código y todos los cambios que se realicen. Así quedan dentro del historial de cambios que garantiza por completo el funcionamiento sin que el código se corrompa o sufra fallos.  Git se ha pensado para posibilitar la ramificación y el etiquetado como procesos de primera importancia, las operaciones que afectan a las ramas y las etiquetas.

5) El comando "git pull" se utiliza para hacer un (pull). Los cambios desde un repositorio remoto y fusionarlos en tu rama local. Es decir, se utiliza para actualizar tu copia local del repositorio con los cambios que otros colaboradores hayan enviado al repositorio remoto desde sus copias locales.

El comando "git add" se utiliza para preparar (puesta en escena) los cambios que has realizado en tus archivos locales y que deseas incluir en tu próxima confirmación (confirmación). Antes de realizar un commit.

"git pull"= se utiliza para traer y fusionar cambios desde un repositorio remoto a tu rama local.

"git add"= se utiliza para preparar los cambios locales para una futura confirmación (commit) en tu rama local.

Repositorio:

https://github.com/Ignacio788/Planificacion-informatica-exam1