Investigación sobre el desarrollo Ágil y DevOps

Trabajo Grupo 5

ÍNDICE

PUNT0 1. DESARROLLO ÁGIL

1.1 Definición

Llamamos “desarrollo ágil” a una serie de metodologías de trabajo que buscan mejorar la eficacia de las técnicas organizativas del trabajo en el ámbito del desarrollo de software.

Este enfoque se creó por primera vez para abordar las limitaciones del método en cascada, que a su vez derivó de las metodologías desarrolladas por Henry Ford en sus cadenas de montaje en 1913. En el año 2001 se establecieron los fundamentos del desarrollo ágil a través de un Manifiesto firmado por un grupo de desarrolladores convocados por Kent Beck, los cuales están considerados los padres de esta metodología.

El enfoque ágil surgió como respuesta a las carencias del método de cascada respecto a la capacidad de reacción en tiempos de desarrollo, y la falta de flexibilidad de dicha metodología. A mediados de los 90 ya era común que la distancia temporal entre la aparición de un problema y la aplicación informática que los resolvía fuera de varios años, lo cual desde el punto de vista empresarial suponía un problema considerable. Eran tantos los cambios que podían sufrir las demandas empresariales y los mercados durante esos años, que posiblemente se cancelarían partes importantes de los proyectos de software antes de poder distribuirse. El desperdicio de tiempo y recursos llevó a que varios desarrolladores de software buscaran una alternativa.

1.2 Valores y principios

De acuerdo con lo establecido en el Manifiesto Ágil, los valores fundamentales son los siguientes:

- Las personas y las interacciones antes que los procesos y las herramientas
- El software en funcionamiento antes que la documentación exhaustiva
- La colaboración con el cliente antes que la negociación contractual
- La respuesta ante el cambio antes que el apego a un plan

Estos valores se pueden consultar en la siguiente dirección web:
Manifesto for Agile Software Development (agilemanifesto.org)

Se puede inferir en base a esto que los valores de la metodología ágil son los de conseguir resultados eficientes, flexibles y en contacto directo con las necesidades del

cliente, estableciendo además un entorno de trabajo sencillo y cómodo para el desarrollador. Esto no significa, obviamente, que lo que aparece a la derecha de cada principio no sea algo deseable (temas como la documentación o la negociación, obviamente, tienen que estar), sino que no se tomará como un valor hegemónico y definitorio en el desarrollo.

El resultado de aplicar estos valores, normalmente, se traduce en la capacidad de proporcionar pequeñas piezas de software en funcionamiento que sean capaces de ir satisfaciendo las necesidades del cliente de forma progresiva pero inmediata. Otro de los valores más comunes de las metodologías ágiles es el de proceso continuo de desarrollo y prueba; a diferencia del método en cascada, donde una fase debe finalizarse por completo antes de poder pasar a la siguiente, en las metodologías ágiles se pueden simultanear ambos procesos, y poder hacer una revisión constante de cada fase.

Además de estos valores fundamentales, en el manifiesto se establecen los denominados como 12 Principios del Desarrollo Ágil, que se derivan de los valores fundamentales, y que son los siguientes:

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 procurando 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.
9. Los patrocinadores, desarrolladores y usuarios deben mantener un ritmo constante de forma indefinida.
10. La atención continua a la excelencia técnica enaltece la agilidad. La simplicidad como arte de maximizar la cantidad de trabajo que se hace, es esencial.

11. Las mejores arquitecturas, requisitos y diseños emergen de equipos que se autoorganizan.
12. En intervalos regulares, el equipo reflexiona sobre la forma de ser más efectivo y ajusta su conducta en consecuencia.

1.3 Ciclos de desarrollo

El desarrollo ágil se divide en 6 ciclos:

- Requisitos: Para poder comenzar a trabajar en el producto y su diseño se necesita una documentación inicial que te permita comenzar a trabajar en el producto centrándose en el resultado final del proyecto como por ejemplo un editor de texto, las funciones disponibles por ejemplo diferentes tamaños de fuente y las características que no admitirá como animaciones u otros. Lo mejor es reducir estos requisitos tanto como sea posible agregando solo características esenciales, si los desarrolladores optan por ignorar esta etapa, son propensos a que las características se deslicen, situación en la que nuevas características no cruciales se agregan constantemente al proyecto, lo que quita tiempo a los desarrolladores de las tareas importantes.
- Diseño: Hay dos tipos de diseños; el virtual y la estructura arquitectónica de la aplicación
    - ➢ Diseño de software: Se introducen los requisitos creados en la primera fase, se analiza cómo abordar estos requisitos y propone herramientas con las que lograr el mejor resultado. En las iteraciones posteriores, los desarrolladores discuten la implementación de la función y la estructura interna del come.
    - ➢ Diseño UI/UX: Durante esta etapa de SDLC, los diseñadores crean una maqueta aproximada de la UI. Si el producto es apto para el consumidor, la interfaz y la experiencia del usuario son lo más importante. Por lo tanto, generalmente es una buena idea revisar a los posibles competidores para ver qué están haciendo bien, y especialmente qué están haciendo mal.

  Se dedican más iteraciones a refinar el diseño inicial y / o reelaborarlo para que se adapte a nuevas características.

- Desarrollo y codificación: La fase de desarrollo consiste en escribir código y convertir la documentación de diseño en el software real dentro del proceso de desarrollo de software. Esta etapa de SDLC es generalmente la más larga, ya

que es la columna vertebral de todo el proceso. Aquí no hay muchos cambios entre las iteraciones.

- Integración y prueba: Esta etapa se dedica a asegurarse de que el software esté libre de errores y sea compatible con todo lo demás que los desarrolladores hayan escrito antes. El equipo de control de calidad lleva a cabo una serie de pruebas para garantizar que el código esté limpio y que se cumplan los objetivos comerciales de la solución. Durante las iteraciones posteriores de esta etapa de SDLC, las pruebas se vuelven más complejas y las cuentas no solo para pruebas de funcionalidad, sino también para la integración de sistemas, interoperabilidad y pruebas de aceptación de usuarios, etc.
- Implementación: La aplicación se implementa en los servidores y se proporciona a los clientes, ya sea para la demostración o para el uso real. Más iteraciones actualizan el software ya instalado, introduciendo nuevas funciones y resolviendo errores.
- Revisión: Una vez que se completan todas las fases de desarrollo anteriores, el propietario del producto reúne al equipo de desarrollo una vez más y revisa el progreso realizado para completar los requisitos. El equipo presenta sus ideas para resolver los problemas que surgieron durante las fases anteriores y el Product Owner toma en consideración sus propuestas. Posteriormente, las fases del ciclo de vida del desarrollo de software ágil comienzan de nuevo, ya sea con una nueva iteración o avanzando hacia la siguiente etapa.

Lo que nos deja como conclusión que el desarrollo de software es un proceso iterativo estructurado. Sin embargo, no existe una única forma «correcta» de hacer Agile; solo hay algunas que se ajustan o no a un equipo en particular.Cada empresa tiene su propia idea de lo que constituye el desarrollo ágil y cada una tiene sus méritos. Lo que importa al final del día es un producto final valioso entregado a tiempo.
Y así es como en Relevant Software desarrollamos aplicaciones bien diseñadas que satisfacen las necesidades comerciales de nuestros clientes. Usamos paradigmas ágiles para todos nuestros proyectos y constantemente entregamos resultados sobresalientes.

1.4 Tipos de metodología ágil y herramientas

1.4.1 Metodología Scrum

Es un proceso de trabajo en equipo que se centra en obtener la mayor productividad posible. Se fijan objetivos a corto plazo , los cuales hay una fecha límite para entregar para lograr obtener el producto final . Todo con la finalidad de conseguir resultados en el menor tiempo posible, donde la innovación y cambios obtienen una respuesta rápida. Esto hace que lo

fundamental de la metodología SCRUM sea el factor humano, su colaboración, sincronización del proyecto y capacidad de adaptarse a los cambios. La metodología Scrum tiene 3 fases, estrategia del desarrollo incremental el resultado basado en el conocimiento de las personas y solapar las fases del desarrollo.
Para la implementación de esta metodología se emplean herramientas. Hay 5 herramientas a destacar:

➔ Jira. Herramienta producida por Atlassian. Estas son sus características más destacadas:
Tableros de Scrum personalizables.
Informes de progreso en el desarrollo del proyecto.
Rastreador de errores y problemas.
Filtros personalizados.
Gestión de usuarios.
Registro del tiempo.
Cuadro de mandos personalizable.
Reportes en tiempo real.
Numerosas integraciones de aplicaciones de terceros

➔ nTask. De propósito más general que propicia:

Portfolio de proyectos: ofrece un repositorio único que permite visualizar todos los proyectos en los que estamos trabajando, según nuestras preferencias.

Gestión de tareas

Visualización de progreso: ofrece herramientas como gráficos de Gantt y hojas de tiempo para el seguimiento del progreso del proyecto.

Seguimiento de problemas: asocia los problemas que se presentan con distintas tareas y asigna su probable resolución en función del resto.

Gestión de reuniones: facilita la organización de reuniones, incorporando de forma automatizada los puntos que deben abordarse.

Gestión del riesgo: permite asignar el índice de riesgo a cada tarea para mitigar su efecto a lo largo de todo el proyecto.

Gestión de equipos: espacios de trabajo diferentes que permite a los equipos trabajar en distintos proyectos de forma simultánea.

➔ ScrumDo. Trabaja sobre la creación de casos de usuario y la planificación de trayectos lógicos sobre el mapeo. Contiene características como:

Gestión de tareas y subtareas

Tableros de proyectos

Integración del calendario del proyecto con actualizaciones en tiempo real.

Creación de casos de usuario

Informes estadísticos basados en la metodología Scrum.

Histograma sobre los plazos de entrega.

Diagramas de flujo acumulativo.

Notificaciones y alertas.

Numerosas integraciones de aplicaciones de terceros.

1.4.2 Agile Kanban

Tiene como objetivo “visualizar el trabajo” mediante sus 4 principios:

1. Empieza con lo que haces ahora
Puedes implementar el marco Kanban a cualquier proceso o flujo de trabajo actual. A diferencia de algunos procesos de gestión ágil más definidos como Scrum, el marco Kanban es lo suficientemente flexible como para adaptarse a las prácticas centrales de tu equipo de trabajo.

2. Comprométete a buscar e implementar cambios progresivos y evolutivos
Los grandes cambios pueden ser perjudiciales para tu equipo y, si intentas cambiar todo a la vez, es posible que el nuevo sistema no funcione como

esperabas. Ese es el motivo por el cual el marco Kanban está diseñado para fomentar la mejora continua y el cambio progresivo. En lugar de cambiar todo de una vez, empieza por buscar cambios progresivos para lograr que los procesos de tu equipo realmente evolucionen con el tiempo.

3. Respeta los procesos, los roles y las responsabilidades actuales
A diferencia de otras metodologías Lean, Kanban no tiene roles integrados y puede funcionar con la estructura y los procesos actuales de tu equipo. Además, tu proceso actual podría tener elementos excelentes, que se perderían si intentaras modificar completamente tu sistema de trabajo de un momento a otro.

4. Impulsa el liderazgo en todos los niveles Con el espíritu de mejora continua, el método Kanban reconoce que el cambio puede provenir de cualquier dirección y no solo "de arriba abajo". Con Kanban, se alienta a los miembros del equipo a participar, proponer nuevas formas para lograr que los procesos evolucionen y emprender nuevas iniciativas de trabajo.

El diagrama de flujo acumulativo (CFD) es otra herramienta analítica utilizada por los equipos kanban para comprender el número de elementos de trabajo en cada estado. Estos ayudan a identificar dificultades a resolver para un mayor rendimiento.

- ➢ Trello: Es probablemente la herramienta Kanban más utilizada. Es gratuita y está disponible tanto para móviles como para ordenadores. Una de sus funciones más útiles es el sistema de notificaciones, que te avisará si el tablero sufre algún cambio.
- ➢ Kambanpad : Una de las herramientas más sencillas de utilizar, debido a los colores que utiliza para organizar el tablero. El diseño del tablero es muy visual. Además incluye la posibilidad de trabajar desde el smartphone.

1.4.3 XP o programación extrema

Se basa en valores, principios y prácticas, y su objetivo es permitir que equipos pequeños y medianos produzcan software de alta calidad y se adapten a los

requisitos cambiantes y en evolución. Lo que diferencia a XP de las demás metodologías ágiles es que hace hincapié en los aspectos técnicos del desarrollo de software. La metodología XP se rige también por las siguientes características:

- ➢ Comunicación constante entre el cliente y el equipo de desarrollo.
- ➢ Respuesta rápida a los cambios constantes.
- ➢ La planificación es abierta con un cronograma de actividades flexible.
- ➢ El software que funciona está por encima de cualquier otra documentación.

Los requisitos del cliente y el trabajo del equipo del proyecto son los principales factores de éxito del mismo.Además, se utilizan técnicas Scrum adaptadas al Kanban, lo que se conoce como Scrumban.

1.4.4 Scrumban

Es un híbrido de Scrum y Kanban .Una mezcla de las ceremonias de Scrum, con los aspectos visuales, los límites de trabajo en progreso, sistemas pull, y flujo continuo de Kanban. Perfecto para compañías que están haciendo la transición de Scrum a Kanban, o para aquellas que nunca han usado metodologías Ágiles, pero están interesadas en cambiar a un flujo de trabajo pull. El tablero Scrumban te proporciona una excelente visión del flujo de trabajo de un proceso. Informa del número de cosas en las que el equipo está trabajando actualmente, así como del número de tareas ya finalizadas. Aumenta la rendición de cuentas, la responsabilidad, la comunicación y los resultados de rendimiento.

1.4.5 SAFe (scaled agile framework)

Provee un conjunto de patrones, flujos de trabajo y roles que permiten aplicar enfoques ágiles en una organización empresarial completa.
El objetivo principal de SAFe es proporcionar a las organizaciones una guía general para aumentar la productividad en el desarrollo de productos a todos los niveles. Es decir, permite aplicar conceptos ágiles a la gestión empresarial teniendo en consideración la estrategia de la compañía y proporcionando una visión de portafolio que requiere ver mucho más allá de la siguiente iteración de cada uno de los equipos ágiles que trabajan en la empresa.

SAFe habla de nivel de equipo, donde se proporciona un modelo para equipos ágiles basados en Scrum y prácticas XP; nivel de programa, donde se coordina el trabajo de múltiples equipos ágiles; nivel de portfolio, donde los programas se alinean con las estrategias de la organización, y por último el flujo de valor, donde se orquesta la entrega de valor.

Design Thinking/Pensamiento de diseño: El Design Thinking es una metodología ágil de innovación centrada en el usuario.
El significado de Design Thinking es pensamiento de diseño de innovación centrada en el usuario, que consiste en integrar las necesidades de las personas y el uso de las nuevas tecnologías para encontrar las soluciones prácticas ante los problemas de las personas. La cual se divide en 6 fases:

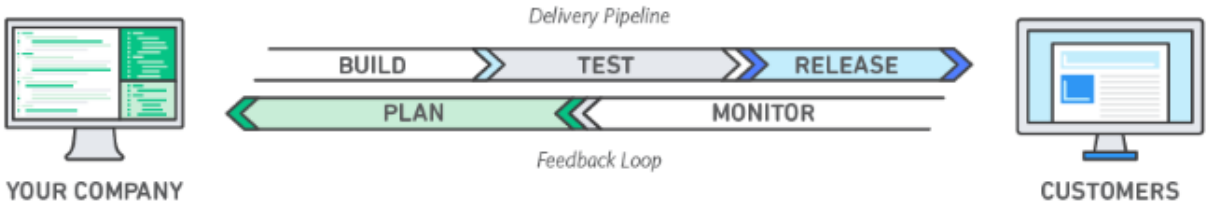
- ➔ Fase de Empatía: Esta es la primera fase del proceso de design thinking donde empleamos mapas para conectar con los consumidores y saber cuáles son sus necesidades.
- ➔ Fase de Definición: definimos el problema y qué palabras emplearemos para resolverlo. ¿Qué 3 elementos se integran en el proceso? Las personas, la tecnología y la empresa son los factores que tenemos que tratar para poder descubrir cuál es el problema.
- ➔ Fase de Ideación: Todas las ideas planteadas nos llevarán a desarrollar un plan de negocio para el producto que lancemos.
- ➔ Fase de prototipado. Podemos estar haciendo el prototipo de un producto en base a este proceso. Aquí vamos a utilizar el lienzo de modelo de negocio para implantar la innovación.
- ➔ Fase de medición y testeo: definir las métricas relevantes para cuantificar los resultados de innovación que queremos implantar en nuestros nuevos productos o servicios.

➔ Fase de aprendizaje: aquí analizaremos los resultados de todas las pruebas para ver si son satisfactorios y así ofrecer la solución correcta a los problemas de nuestros consumidores.

Algunas de las herramientas utilizadas en el Design Thinking:

➔ **Mapa de actores:** En esta técnica tienes que generar un mind map e ir desgranando todos los posibles actores intervinientes en el servicio/producto que estás analizando. Se van haciendo “clusters” o agrupaciones. Lo bueno de esta técnica es que te permite ver de un vistazo a todos los usuarios o clientes con los que tendrás que empatizar a fondo. Solo se utiliza cuando hay implicados gran cantidad y tipologías de usuarios.

➔ **Mood Board:** En este caso se trata de tableros de inspiración, que nos ayudan a través de imágenes a acercarnos el problema. El mood board se hace con imágenes sacadas de revistas, libros, o fotografías tomadas por ti. Puede reflejar la situación actual, o ser un panel de inspiración con ideas de otros lugares.



PUNTO 2. DevOps

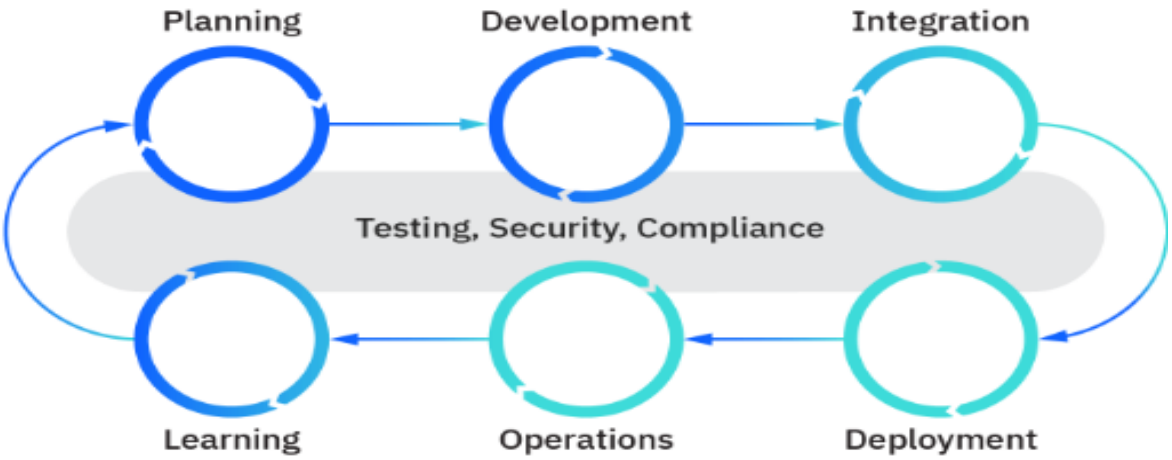
2.1. DevOps (Development and Operations)

DevOps combina el desarrollo (Dev) y las operaciones (Ops) para unir a personas, procesos y tecnología en el planeamiento de aplicaciones, el desarrollo, la entrega y las operaciones. DevOps permite a los roles previamente aislados, como el desarrollo, las operaciones de TI, la ingeniería de calidad y la coordinación y colaboración de seguridad mantenerse interconectados.

Los equipos adoptan la cultura, las prácticas y las herramientas de DevOps para aumentar la confianza en las aplicaciones que crean, responder mejor a las necesidades de los clientes y lograr objetivos empresariales más rápido. DevOps ayuda a los equipos a proporcionar valor continuamente a los clientes mediante la producción de productos mejores y más confiables.

2.1.1. DevOps y el ciclo de vida de la aplicación

DevOps influye en el ciclo de vida de la aplicación a lo largo de su planeamiento, desarrollo, entrega y fases de operaciones.
Cada fase se basa en las demás y ninguna es específica de un rol en específico.



2.1.2. Fases de del estilo de vida de la aplicación DevOps

- ➔ Planificación:
  En este flujo de trabajo, los equipos determinan las nuevas características y funcionalidades del próximo lanzamiento, a partir de comentarios y casos de

estudio de usuarios finales priorizados, así como contribuciones de todos los stakeholders internos

➔ Desarrollo:
Incluye todos los aspectos del desarrollo del código del software.

1º. Seleccionar un entorno de desarrollo.

2º. Escribir, probar, revisar e integrar el código.

3º. Compilar el código en máquinas virtuales para implementarlo en varios entornos.

4º. Usar el control de versiones (con Gilt que es un sistema de control de versiones distribuido) para colaborar en el código y trabajar en paralelo.

➔ Entrega:
La entrega es el proceso de implementación coherente y confiable de aplicaciones en entornos de producción, preferiblemente por entrega continua. (entrega continua es el proceso de automatizar la compilación, la prueba, la configuración y la implementación desde una compilación a un entorno de producción)

1º. Se define un proceso de administración de versiones con fases de aprobación manuales claras.

2º. Establecer puertas automatizadas para mover aplicaciones entre distintas versiones hasta la versión final a los clientes.

3º. Automatizar los procesos de entrega.

➔ Operations:
Implica mantener, supervisar y solucionar problemas de aplicaciones en entornos de producción.

➔ Aprendizaje (a veces llamado retroalimentación continua):
Esta es la recopilación de comentarios de usuarios finales y clientes acerca de las características, la funcionalidad, el rendimiento y el valor empresarial para volver a la etapa de planificación y añadir tanto mejoras como nuevas características al próximo lanzamiento.

➔ Pruebas continuas:
Los ciclos de vida tradicionales de DevOps incluyen una etapa de "prueba" discreta que se produce entre la integración y la implementación.

➔ Seguridad:
Mientras que las metodologías en cascada y las implementaciones ágiles "añaden" los flujos de trabajo de seguridad después de la entrega o la implementación, DevOps se esfuerza por incorporar la seguridad desde el principio (planificación), cuando los problemas de seguridad son más fáciles y menos costosos de abordar, y continuamente durante el resto del ciclo de desarrollo.

➔ Conformidad:
El cumplimiento normativo (gestión y riesgo) también se aborda mejor al principio y durante todo el ciclo de vida del desarrollo.

2.1.3. Cultura Devops

Generalmente se admite que los métodos DevOps no pueden funcionar sin un compromiso con la cultura DevOps, la que se puede resumir como un enfoque organizativo y técnico diferente al desarrollo de software.
A nivel de organización, DevOps requiere comunicación continua, colaboración y responsabilidad compartida entre todos los involucrados en la entrega de software, desarrollo de software y equipos de operaciones de TI.
En la mayoría de los casos, la mejor manera de lograrlo es descomponer estos silos y reorganizarlos en equipos de DevOps autónomos e interfuncionales que pueden trabajar en proyectos de código de principio a fin.

2.2 Herramientas DevOps

➔ Herramientas de gestión de proyectos:
Herramientas que permiten a los equipos crear una lista de casos de usuarios (requisitos) que forman proyectos de codificación, desglosarlos en tareas más pequeñas y realizar un seguimiento de estas hasta su finalización.
➔ Repositorios de código fuente de colaboración:
Entornos de codificación controlados por ediciones que permiten a varios desarrolladores trabajar en la misma base de código.
➔ Líneas de trabajo CI/CD:

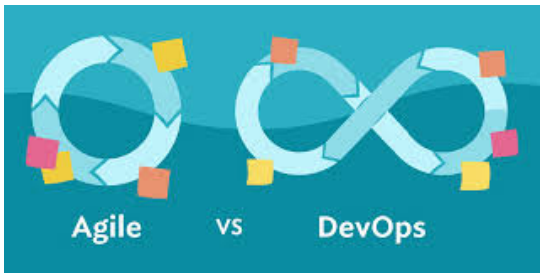
Herramientas que automatizan la extracción, la creación, la prueba y la implementación de código

➔ Marcos de automatización de pruebas: I

Incluyen herramientas de software, bibliotecas y mejores prácticas para automatizar pruebas de unidad, contrato, funcionalidad, rendimiento, usabilidad, penetración y seguridad.

PUNTO 3. Agile VS DevOps

3.1 Diferencias y similitudes



Tras definir cada uno de estos términos pasaremos a analizar sus similitudes y diferencias. Las principales diferencias que encontramos entre estas dos metodologías son :

- La diferencia fundamental es que DevOps trata de unir dos grandes equipos diferenciados para mejorar la comunicación entre ellos para lanzar software más rápidamente, mientras que Agile se centra en hacer que pequeños equipos colaboren entre sí para adaptarse más rápidamente a los posibles cambios durante el desarrollo de una aplicación.
- En cuanto al tamaño de equipo que participa en el desarrollo , la metodología Agile involucra un equipo pequeño mientras que DevOps involucra un número mayor de profesionales.
- La metodología Agile fracciones sus logros en sprints, normalmente dura poco menos de un mes, en DevOps trata de entregar código todos los días. Esto es

posible ya que la retroalimentación se produce dentro del propio equipo a diferencia de la primera.

- En la documentación, la metodología Agile proporciona mayor importancia al desarrollo dejando en segundo plano la documentación, en DevOps es necesario generar una documentación más detallada porque el programa está en constante tránsito.
- Agile se centró en acelerar el proceso de desarrollo descuidando las pruebas, DevOps aprovechó e introdujo un marcado de carácter holístico.

Sin ninguna duda, se trata de dos metodologías que no pueden considerarse como incompatibles, pero tampoco es justo decir que una es la evolución de la otra. Simplemente plantean dos enfoques con marcadas diferencias en la concepción del proceso productivo. En cuanto a sus similitudes podemos destacar:

- Ambos permiten acelerar los diferentes procesos del desarrollo y distribución de software utilizando la organización y el trabajo en equipo como herramienta principal, constituyen pasos, pautas, mecanismos, procesos, plantillas preestablecidas de cómo se debe desarrollar un proceso de desarrollo de software, así como los profesionales que intervienen en cada una de las etapas.
- Tanto Agile como DevOps dan prioridad explícitamente a las personas y las interacciones, al software en funcionamiento, a la colaboración con el cliente y a la respuesta frente al cambio.
- Están muy enfocados en ofrecer lo mejor a los clientes aunque lo hagan de maneras distintas.

3.2 Colaboración entre ambos

Aunque sean muy diferentes entre ambos, tienen la misma finalidad y no podemos excluirlos mutuamente. Es posible la colaboración entre ambos, se puede adoptar uno encima de otro y de esta manera funcionar en conjunto. Son totalmente compatibles y gracias a su combinación muchas empresas están acelerando sus proyectos consiguiendo resultados de alta calidad en un tiempo más reducido.

3.3 Mejor metodología

Para un perfil de programador con el ciclo superior de Desarrollo de Aplicaciones Multiplataforma, scrum cuadra mejor ya que se programa en equipo el cuál vas a poder usar para cualquier proyecto, es la más utilizada a día de hoy y la que se usa en las prácticas por lo que estarás más familiarizado con esta metodología.