Tema del foro: "Impulsando la innovación con Test-driven Development (TDD)"

1. **Innovación a través de la calidad del código**: Exploraremos cómo TDD puede mejorar la calidad del código al enfocarse en pruebas exhaustivas. ¿Cómo influye esto en la capacidad de innovar y desarrollar características creativas de manera más ágil?

La calidad consiste en la estandarización de los procesos , el establecimiento de controles en las operaciones la mejora continua, a su vez el conocimiento promueve la innovación en los procesos, por lo que el constante aprendizaje en las empresas les permite mantenerse competitivas en el mercado

1. **Iteración y mejora continua**: Discutiremos cómo el ciclo "Red-Green-Refactor" de TDD fomenta la iteración y la mejora continua en el desarrollo de software. ¿Cómo esto impulsa la innovación y la evolución de los productos y servicios?

El proceso iterativo es la práctica de elaborar, refinar y mejorar un proyecto, producto o iniciativa. Los equipos que usan procesos de desarrollo iterativos crean, prueban y hacen revisiones hasta que se sienten satisfechos con el resultado final. Puedes pensar en un proceso iterativo como si fuese una metodología de prueba y error que acerca el proyecto al objetivo

1. **Rápida retroalimentación y toma de decisiones**: Analizaremos cómo TDD proporciona una retroalimentación rápida sobre la calidad del código y cómo esto permite una toma de decisiones más ágil en el proceso de desarrollo. ¿Cómo se traduce esto en la capacidad de innovar y adaptarse rápidamente a los cambios?

Con esta práctica se consigue, entre otras cosas, un código más robusto, más seguro, más mantenible y una mayor rapidez en el desarrollo.

1. **Aplicación de TDD en proyectos innovadores**: Compartiremos ejemplos y casos de éxito de proyectos innovadores que han aplicado TDD en su desarrollo. ¿Cómo ha contribuido TDD a la creación de productos o soluciones disruptivas?

Ha contribuido en los siguientes:

1. Todo queda probado. Haciendo los test primero, se evita que se dejen para el final, se olviden casos de prueba, o se recorten por falta de tiempo.
2. Cliente satisfecho. Los test a cumplir se basan en los casos de uso, extraídos de los requisitos acordados con el cliente.
3. Mejor código. Tener definida una etapa de refactorización obliga a que el código escrito sea revisado y mejorado siempre.
4. Evita desarrollo innecesario. El código que se escribe se hace con el objetivo de pasar un test, y este test ha sido escrito para comprobar un caso de uso, basado en un requisito del cliente. Por lo que siempre es necesario.