Publicado en Madrid el 14/10/2021

El TDD, clave para la creación de software con menos defectos y más capacidad de mantenimiento

El testing, método para verificar el producto, se convierte en la práctica que más ayuda a la hora de crear un software de calidad. El propósito final de las pruebas de software es identificar errores, lagunas o requisitos necesarios no contemplados. Creditas identifica las principales ventajas del uso de TDD como metodología para implementar un desarrollo basado en pruebas

Cualquier empresa que se precie debe buscar siempre la forma de ofrecer calidad y de transmitir confianza a sus clientes, pero ¿de qué manera hacerlo si se depende del software de los servicios? Creditas, la plataforma líder en América Latina de soluciones de consumo y préstamos 100% online, lo tiene claro: se debe aportar valor a los clientes y trabajar para contar con un software robusto, fiable y seguro.

La práctica que más ayuda a la hora de crear un software de calidad es el testing, un método para verificar si el producto coincide con los requisitos esperados y para garantizar que el producto esté, en su mayor medida, libre de defectos. Existen diversas formas de probar el software desarrollado, entre ellas se encuentran los tests unitarios, los tests de integración, los performance testings o los tests de aceptación, pero en todos ellos se ejecutan distintos componentes de sistema que, a través de herramientas manuales o automatizadas, evalúan una o más propiedades de interés. El propósito final de las pruebas de software no es otro que identificar errores, lagunas o requisitos necesarios no contemplados.

Una de las tendencias más actuales dentro de la programación de software es el TDD o desarrollo guiado por pruebas (Test-Driven Development por sus siglas en inglés). Esta metodología se basa en convertir los requisitos de un proyecto en casos de prueba, de manera que si al ejecutarlos, fallan, se aplique el refactoring para limpiar código o mejorar la implementación. Entre las principales ventajas del uso de TDD se encuentran:

Permite descubrir errores de forma anticipada
Ofrece un código mejor diseñado, más limpio y escalable
Ayuda a comprender cómo se utilizará el código
Ofrece mayor confianza para refactorizar el código que se está implementando
Es bueno para el trabajo en equipo, aplicando técnicas como Ping Pong
Ahorra tiempo en depurar al construir software más acotado

Sin embargo, y a pesar de las ventajas del TDD, no todos los desarrolladores lo implementan y es que adoptarlo puede convertirse en una tarea difícil que requiere mucha disciplina. “En muchas ocasiones, los equipos no adoptan el TDD por falta de tiempo y/o entrenamiento previo en esta técnica de desarrollo”, explica Kevin Sotomayor, Software Engineer de Creditas. “Además, un desarrollo basado en pruebas puede parecer más lento, aunque en realidad acaba aportando más valor al proceso por

muchas razones”.

Un desarrollo basado en pruebas puede ser muy positivo, ya que al acabarlo también se finalizan los tests. Además, como la prueba estará enfocada al caso de uso, generará documentación útil para los desarrolladores, acotará lo que debe hacer ese desarrollo, basándose en lo que dicen las pruebas, y ofrecerá un código más limpio y estable.

“No hay una guía estricta de cuándo aplicar el TDD y cuándo no. Depende mucho de la capa a testear del software, de la cultura organizativa, del grado de madurez del equipo, del expertise del desarrollador, de si se está o no definiendo una arquitectura o desarrollando una funcionalidad…”, explica Miguel Alberto García, Software Engineer de Creditas. “En Creditas, el uso de TDD es una metodología de desarrollo y diseño considerada como parte fundamental de nuestro trabajo diario”.

La metodología TDD requiere de mucho tiempo de aprendizaje, que se irá reduciendo mientras se adquiere experiencia durante su práctica. El TDD a largo plazo tiene más beneficios que contras, ya que el uso de pruebas facilita la creación de software con menos defectos y es más sencillo de mantener. Además, reduce el tiempo de desarrollo y promueve los tests automáticos, creando documentación y una suite de tests que cubren los casos de uso al finalizar el desarrollo de una característica.

“Es quien construye el código, quien hace que el TDD funcione”, añade Sotomayor. “Hay que comprender que las pruebas son parte del sistema, que deben diseñarse y que el código de prueba evoluciona hacia algo más específico, mientras que el código de producción evoluciona hacia algo más genérico”.

Si no se emplean principios de diseño para hacer evolucionar el código y no se evolucionan las pruebas junto al código, las pruebas y el código irán en direcciones opuestas. Si no se tratan las pruebas como parte del sistema y no se plantea acoplarlas, separarlas y aislarlas, la arquitectura puede verse afectada.

Datos de contacto:
Trescom
91 411 58 68

Nota de prensa publicada en: Madrid

Categorías: Programación Software