


## Assignment brief A.B.

### PORTADA

<b>Nombre Alumno / DNI</b>	Eugenio Lancha Gómez / 54186490P
<b>Título del Programa</b>	OFICIAL BACHELOR'S DEGREE IN DATA SCIENCE & BUSINESS ANALYTICS
<b>Nº Unidad y Título</b>	UNIT 1-PROGRAMMING AND CODING
<b>Año académico</b>	2023-2024
<b>Profesor de la unidad</b>	GABRIELA GARCIA
<b>Título del Assignment</b>	AB FINAL
<b>Día de emisión</b>	13/10/2023
<b>Día de entrega</b>	23/01/2024
<b>Nombre IV y fecha</b>	
<b>Declaración del estudiante</b>	<p>Certifico que la presentación del assignment es completamente mi propio trabajo y entiendo completamente las consecuencias del plagio. Entiendo que hacer una declaración falsa es una forma de mala práctica.</p> <p>Fecha: 22/01/2024</p> <p>Firma del alumno:</p> 

### Plagio

El plagio es una forma particular de hacer trampa. El plagio debe evitarse a toda costa y los alumnos que infrinjan las reglas, aunque sea inocentemente, pueden ser sancionados. Es su responsabilidad asegurarse de comprender las prácticas de referencia correctas. Como alumno de nivel universitario, se espera que utilice las referencias adecuadas en todo momento y mantenga notas cuidadosamente detalladas de todas sus fuentes de materiales para el material que ha utilizado en su trabajo, incluido cualquier material descargado de Internet. Consulte al profesor de la unidad correspondiente o al tutor del curso si necesita más consejos.



## **La Relevancia del Testing y Pruebas de Código en el Desarrollo de Software**

En el dinámico mundo del desarrollo de software, la relevancia del testing y las pruebas de código es fundamental. Estos procesos son cruciales para asegurar no solo la funcionalidad y eficiencia del software, sino también su seguridad y fiabilidad. El testing y las pruebas de código se integran en el ciclo de vida del desarrollo de software para identificar y solucionar problemas desde las etapas iniciales hasta las finales, mejorando así la calidad del producto final.

El propósito de estas prácticas es doble: por un lado, verificar que el software cumpla con los requisitos específicos y las expectativas de los usuarios; por otro lado, garantizar que el software sea robusto y esté libre de errores críticos que podrían afectar su rendimiento o seguridad. En este sentido, el testing y las pruebas de código no son simplemente pasos en el proceso de desarrollo, sino elementos integrales que contribuyen significativamente al éxito y la sostenibilidad a largo plazo de cualquier proyecto de software.

A medida que las tecnologías evolucionan y se vuelven más complejas, la importancia de estas prácticas se magnifica, destacando la necesidad de métodos de prueba sofisticados y herramientas avanzadas para mantener la calidad y competitividad del software en el mercado global.

### **Definición y Diferencia entre Testing y Pruebas de Código**

**Testing:** El testing en el desarrollo de software se refiere al proceso general de evaluación del software para detectar defectos, verificar su funcionalidad y asegurar que cumple con los requisitos establecidos. Incluye una amplia gama de actividades que van desde la revisión del código hasta la ejecución de pruebas en diferentes niveles del software, como pruebas unitarias, de integración, de sistema, etc.

**Pruebas de Código:** Las pruebas de código, a menudo consideradas una subcategoría del testing, se enfocan específicamente en verificar la corrección del código fuente.

Estas pruebas incluyen la ejecución del código en condiciones variadas para asegurar que se comporta como se espera en todos los casos posibles.

### **Objetivos y Beneficios de Realizar Pruebas**

Los objetivos principales de realizar pruebas en el desarrollo de software incluyen:

**Garantizar Calidad:** Asegurar que el software cumple con los estándares de calidad establecidos y satisface las necesidades del usuario.

**Identificar y Solucionar Errores:** Detectar y corregir fallos en el software antes de su lanzamiento, reduciendo así el riesgo de problemas en producción.

**Verificar la Conformidad con Requisitos:** Confirmar que el software cumple con los requisitos y especificaciones técnicas y funcionales.

**Mejorar la Confianza del Usuario:** Generar confianza en la fiabilidad y seguridad del software entre los usuarios finales y las partes interesadas.

Los beneficios de realizar pruebas son extensos y significativos:

**Mejora de la Calidad del Producto:** Las pruebas contribuyen a la entrega de un software de mayor calidad, lo que a su vez mejora la satisfacción del usuario y la reputación de la marca.

**Reducción de Costos a Largo Plazo:** Aunque las pruebas requieren una inversión inicial, ayudan a evitar costos mucho mayores asociados con la corrección de errores después del lanzamiento.

**Prevención de Fallos Críticos:** Las pruebas ayudan a prevenir fallos que podrían ser catastróficos en términos de costos, reputación y seguridad.

### **Tipos de Pruebas**

Esta sección describe detalladamente varios tipos de pruebas utilizadas en el desarrollo de software, cada una con sus herramientas asociadas.

**Pruebas Unitarias:** Se centran en verificar la funcionalidad de componentes individuales o unidades del código. Son útiles para asegurar que cada parte del código funcione correctamente por separado. Herramientas populares incluyen JUnit para Java, NUnit para .NET, y pytest para Python.

**Pruebas de Integración:** Evalúan cómo diferentes módulos o servicios funcionan juntos. Estas pruebas son fundamentales para detectar problemas en las interfaces y en la integración entre distintas partes del sistema. Herramientas como Postman para APIs y TestNG son comúnmente usadas.

**Pruebas de Sistema:** Verifican el comportamiento y las funcionalidades del software completo para asegurarse de que cumple con los requisitos específicos. Se suelen realizar en un entorno que simula la producción.

**Pruebas de Aceptación:** Realizadas con el objetivo de validar si el software cumple con las expectativas y necesidades del cliente. Las pruebas de aceptación del usuario (UAT) son un ejemplo común, donde los usuarios finales prueban el software en condiciones reales.

**Pruebas de Carga:** Comprueban cómo el sistema se comporta bajo una carga significativa, usualmente simulando un gran número de usuarios. Herramientas como LoadRunner y JMeter son frecuentemente utilizadas.

**Pruebas de Estrés:** Similar a las pruebas de carga, pero diseñadas para llevar al sistema a sus límites y más allá, con el fin de ver cómo se comporta bajo condiciones extremas.

**Pruebas de Seguridad:** Se enfocan en identificar vulnerabilidades en el software que podrían ser explotadas por ataques maliciosos. Herramientas como OWASP ZAP y Nessus se utilizan para estas pruebas.

**Pruebas de Regresión:** Aseguran que los cambios recientes en el código no afecten negativamente a las funcionalidades existentes. Son cruciales para proyectos que están en constante evolución.

Cada uno de estos tipos de pruebas cumple con un propósito específico en el ciclo de vida del desarrollo de software y, cuando se utilizan en conjunto, contribuyen significativamente a la creación de un producto de software robusto y confiable.

### **Técnicas de Testing**

En esta sección, exploraremos diversas técnicas de testing utilizadas en el desarrollo de software, destacando sus beneficios y retos asociados.

Desarrollo Guiado por Pruebas (TDD - Test Driven Development):

**Descripción:** TDD es una técnica de desarrollo de software en la que se escribe un test antes de escribir el código funcional. El proceso se sigue en un ciclo corto: primero se escribe una prueba que falla, luego se escribe el código necesario para pasar la prueba, y finalmente se refactoriza el nuevo código.

**Beneficios:** TDD ayuda a asegurar que el código tenga una alta cobertura de pruebas y fomenta un diseño modular y claro. También facilita la detección temprana de errores.

**Retos:** Puede ser desafiante para equipos sin experiencia en esta metodología. Requiere disciplina y un cambio en la mentalidad tradicional de desarrollo.

Desarrollo Guiado por Comportamiento (BDD - Behavior Driven Development):

**Descripción:** BDD extiende los principios de TDD y se enfoca en la colaboración entre desarrolladores, QA, y no técnicos. Las pruebas se escriben en un lenguaje natural que describe el comportamiento del software.

**Beneficios:** Promueve la comprensión compartida del cómo y el porqué de las funcionalidades del software, y facilita la comunicación entre los miembros del equipo.

Retos: Implementar BDD puede requerir formación adicional y ajustes en la dinámica del equipo. Puede ser más lento al inicio debido a la necesidad de escribir pruebas detalladas.

#### Pruebas de Integración Continua:

Descripción: Esta técnica implica la integración y prueba automáticas de cambios de código en un repositorio compartido varias veces al día.

Beneficios: Permite detectar y solucionar problemas rápidamente, mejora la calidad del código y reduce el tiempo de entrega.

Retos: Requiere una configuración y mantenimiento constantes del entorno de integración continua.

#### Pruebas Exploratorias:

Descripción: En las pruebas exploratorias, los testers exploran activamente el software sin un conjunto predefinido de pruebas, utilizando su experiencia y creatividad para identificar problemas.

Beneficios: Permite descubrir errores que no se habrían encontrado con pruebas estructuradas. Es flexible y se adapta bien a los cambios.

Retos: Depende en gran medida de la habilidad y experiencia del tester. Puede ser difícil de documentar y repetir.

Estas técnicas no son excluyentes y a menudo se combinan para formar un enfoque de testing integral y efectivo. La elección de las técnicas depende del contexto del proyecto, los recursos disponibles y los objetivos específicos del equipo de desarrollo.

### **Automatización de Pruebas**

La automatización de pruebas es un componente esencial en el desarrollo de software moderno, proporcionando eficiencia y consistencia en el proceso de testing.

#### Introducción a la Automatización y sus Ventajas

La automatización de pruebas implica el uso de software para controlar la ejecución de pruebas, comparar los resultados esperados con los resultados reales, y medir el rendimiento del software. A diferencia de las pruebas manuales, que requieren intervención humana, las pruebas automatizadas se ejecutan de forma automática y repetible.

#### Ventajas:

**Eficiencia Mejorada:** La automatización permite realizar pruebas más rápidamente que las pruebas manuales, especialmente en grandes proyectos.

**Consistencia:** Elimina los errores humanos en la ejecución de pruebas, asegurando que cada prueba se ejecute de la misma manera cada vez.

**Cobertura de Pruebas:** Permite aumentar la cobertura de pruebas del software, incluyendo aspectos que pueden ser difíciles de evaluar manualmente.

**Reusabilidad de Scripts de Prueba:** Los scripts de prueba automatizados se pueden reutilizar para diferentes versiones del software, ahorrando tiempo y recursos.

**Integración con Desarrollo Continuo:** Facilita la integración y entrega continua (CI/CD) al permitir pruebas rápidas y frecuentes.

**Herramientas y Frameworks Populares para la Automatización de Pruebas.**

**Selenium:** Ampliamente usado para automatizar pruebas web. Compatible con múltiples lenguajes de programación y navegadores.

**JUnit y TestNG:** Utilizados para pruebas automatizadas en proyectos Java.

**Cypress:** Framework de pruebas end-to-end para aplicaciones web modernas.

**Appium:** Herramienta para automatización de pruebas de aplicaciones móviles.

**Robot Framework:** Framework de automatización de pruebas genérico, adecuado para varias aplicaciones y pruebas de aceptación.

La automatización de pruebas no reemplaza completamente las pruebas manuales, especialmente en escenarios donde la interpretación humana y la experiencia son clave. Sin embargo, es una parte integral del proceso de testing, contribuyendo significativamente a la calidad y eficiencia del desarrollo de software.

## **Casos de Uso y Ejemplos**

Esta sección presenta ejemplos prácticos y casos de uso donde se aplican distintos tipos de pruebas y técnicas de testing en proyectos reales.

**Uso de TDD en el Desarrollo de un Sistema de Reservas:**

**Contexto:** Una empresa desarrolla un sistema de reservas en línea.

**Aplicación de TDD:** Los desarrolladores comienzan escribiendo pruebas para cada función esperada (por ejemplo, crear una reserva, modificarla, cancelarla). Después, escriben el código para pasar estas pruebas. Este enfoque asegura que todas las funcionalidades clave estén probadas desde el principio.

**Resultado:** Se logra un código bien estructurado y con una alta cobertura de pruebas, lo que reduce significativamente los errores en producción.

**Automatización de Pruebas en un E-commerce:**

**Contexto:** Un sitio web de comercio electrónico que regularmente añade nuevas características.

**Automatización Implementada:** Se utilizan herramientas como Selenium para automatizar pruebas de la interfaz de usuario, asegurando que las funciones de compra y navegación funcionen correctamente con cada actualización.

Resultado: La automatización permite un proceso de desarrollo más ágil y reduce el tiempo necesario para lanzar nuevas características.

Pruebas de Carga en una Aplicación de Medios Sociales:

Contexto: Una popular aplicación de medios sociales experimenta fluctuaciones en el tráfico de usuarios.

Implementación de Pruebas de Carga: Se utilizan herramientas como JMeter para simular altos volúmenes de tráfico y evaluar el rendimiento de la aplicación bajo carga pesada.

Resultado: Estas pruebas ayudan a identificar y solucionar cuellos de botella, garantizando que la aplicación se mantenga estable y eficiente incluso durante picos de uso.

Estos casos ilustran cómo diferentes técnicas y tipos de pruebas se aplican en situaciones reales, destacando su impacto positivo en la calidad y fiabilidad del software. Permiten a las empresas desarrollar y mantener aplicaciones robustas que cumplen con las expectativas de los usuarios y se adaptan a las necesidades cambiantes del mercado.

## La Importancia de las Pruebas y el Testing en la Calidad del Software

En conclusión, el testing y las pruebas de código desempeñan un papel crucial en el desarrollo de software. Estas prácticas no solo garantizan que el software cumpla con los requisitos técnicos y de negocio, sino que también aseguran la calidad, seguridad y fiabilidad del producto final. A través de diferentes tipos de pruebas y técnicas de testing, desde pruebas unitarias hasta pruebas de aceptación y automatización, los desarrolladores pueden identificar y solucionar problemas de manera eficiente y efectiva.

La implementación de prácticas de testing robustas es esencial en un mercado tecnológico que evoluciona rápidamente, donde la calidad del software puede ser un diferenciador clave en la competitividad de un producto. Además, a medida que crece la complejidad de las aplicaciones de software, las pruebas se vuelven aún más importantes para asegurar que las aplicaciones sean seguras, eficientes y amigables para el usuario.

Los ejemplos presentados demuestran cómo la adopción de estas prácticas no solo mejora la calidad del software, sino que también contribuye a la satisfacción del cliente y al éxito comercial a largo plazo. Por lo tanto, las pruebas y el testing deben ser considerados no como una tarea secundaria, sino como una parte integral del proceso de desarrollo de software.

Este informe ha destacado la relevancia del testing y pruebas de código, subrayando que la calidad del software es un viaje continuo, no un destino. Con la adopción de las técnicas y herramientas adecuadas, los equipos de desarrollo pueden



enfrentar los desafíos de un panorama tecnológico en constante cambio y entregar productos de software que no solo cumplan, sino que superen las expectativas.

#### Referencias:

**OpenAI, (s.f.). Chat.** Disponible en: <https://chat.openai.com/>

**IBM, (s.f.). 'Software Testing', IBM.** Disponible en: <https://www.ibm.com/es-es/topics/software-testing>

**Academia QA, (s.f.). 'Importancia del Testing de Software', Academia QA.** Disponible en: <https://academiaqa.com/importancia-del-testing-de-software/>

**Atlassian, (s.f.). 'Tipos de pruebas de software', Atlassian.** Disponible en: <https://www.atlassian.com/es/continuous-delivery/software-testing/types-of-software-testing>

**Rootstack, (s.f.). 'Razones por las que debes aplicar pruebas a tu desarrollo de software', Rootstack.** Disponible en: <https://rootstack.com/es/blog/razones-por-las-que-debes-aplicar-pruebas-tu-desarrollo-de-software>

**Platzi, (s.f.). 'Testing: Ventajas y formas de realizar pruebas', Platzi.** Disponible en: <https://platzi.com/blog/testing-ventajas-formas-de-realizar-pruebas/>

**Platzi, (s.f.). '¿Qué es y cómo funciona TDD?', Platzi.** Disponible en: <https://platzi.com/blog/que-es-y-como-funciona-tdd/>