ESCUELA DE INGENIERÍA EN CIENCIAS Y SISTEMAS
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

ECYS | DTT
DESARROLLO DE TRANSFERENCIA TECNOLÓGICA

Día, Fecha:	Miércoles, 11/09/2024
Hora de inicio:	15:40

Análisis y Diseño de Sistemas 2 [B]

Luis Angel Barrera Velásquez

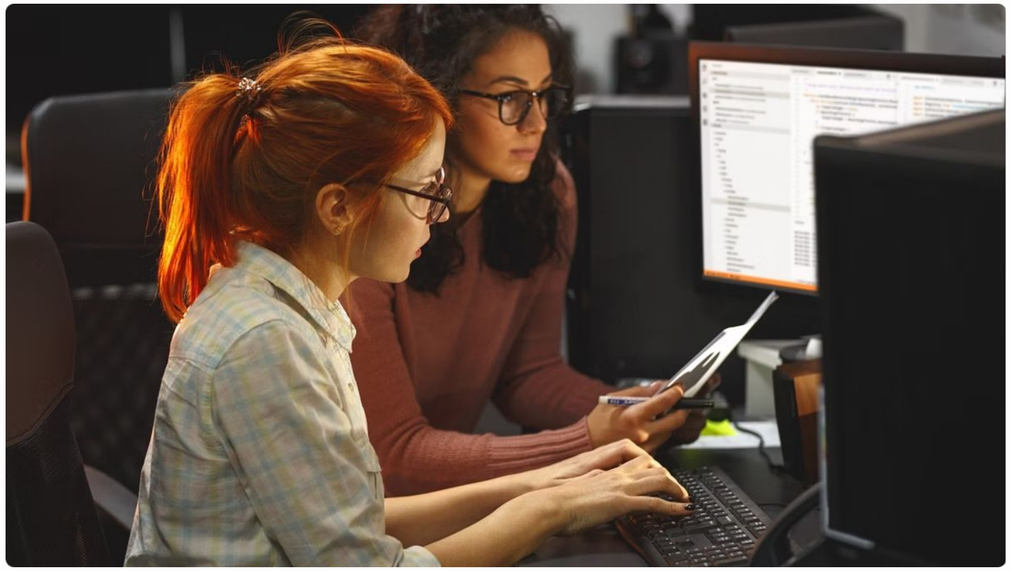
AGENDA

1	Avisos
2	Lectura enunciado
3	Contenido Teórico Pruebas Funcionales
4	Kahoot

Pruebas de software

Introducción a las pruebas de software

Las pruebas de software son un proceso fundamental para garantizar la calidad y el correcto funcionamiento de los sistemas informáticos. Permiten identificar y corregir errores antes de que el software se ponga en producción, lo que ahorra tiempo y recursos.



Importancia de las pruebas de software

1

Calidad

Las pruebas ayudan a asegurar que el software cumpla con los requisitos y funcione sin errores.

2

Satisfacción del usuario

Un software bien probado brinda una mejor experiencia de usuario y genera más confianza.

3

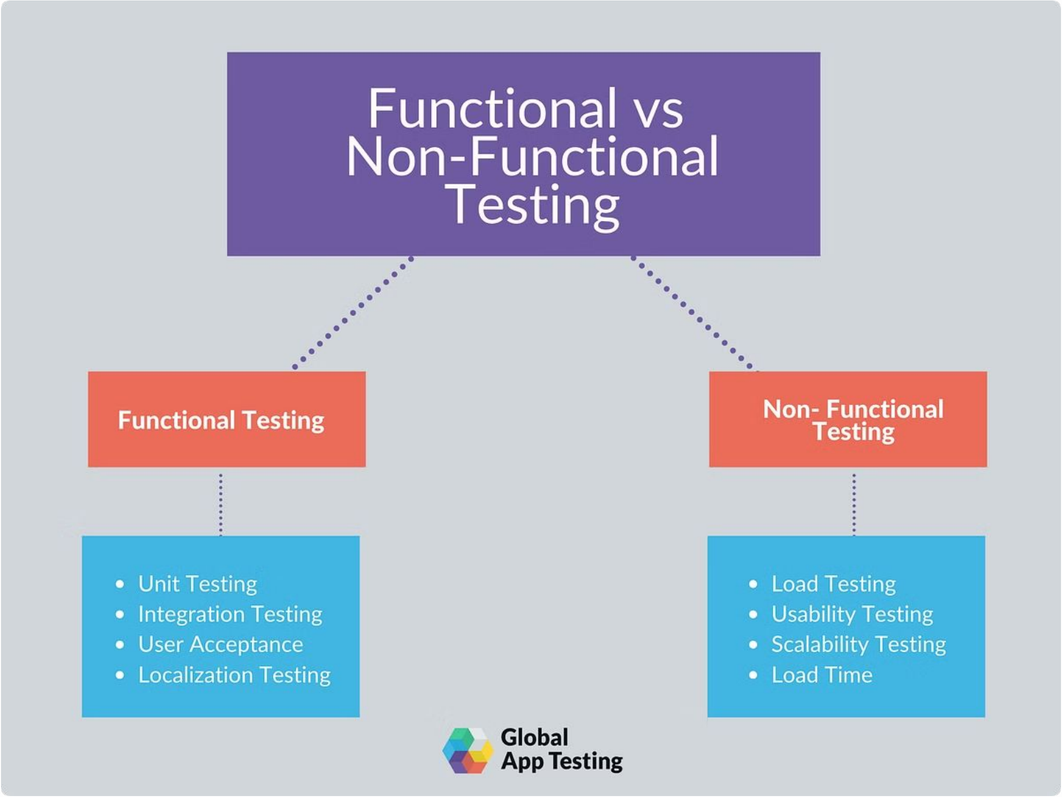
Detección temprana de errores

Identificar y corregir problemas en las primeras etapas reduce costos y esfuerzo..

Tipos de pruebas de software

Pruebas Funcionales

Validan que el software cumpla con los requisitos especificados.



Pruebas No Funcionales

Evalúan aspectos como rendimiento, seguridad, usabilidad y compatibilidad.

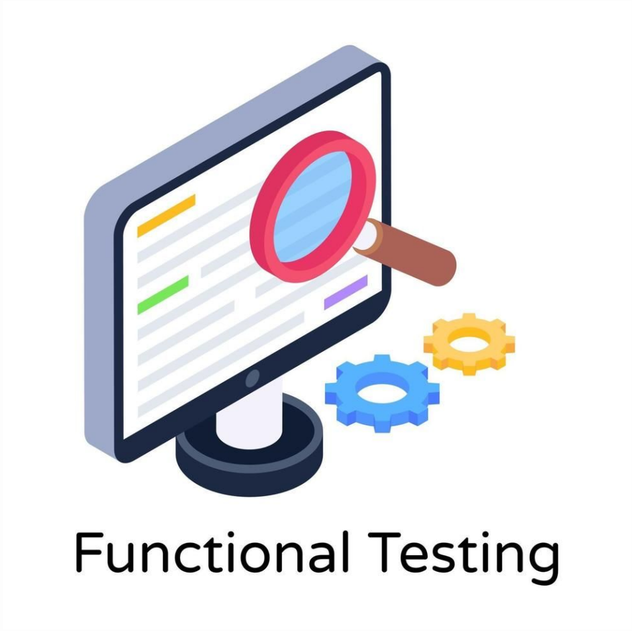
Pruebas Funcionales

Validación de requisitos

Aseguran que el software cumpla con las especificaciones del cliente.

Pruebas de aceptación

Verifican que el software sea aceptado por los usuarios finales.



Casos de prueba

Se diseñan escenarios de prueba para cubrir los diferentes casos de uso.

Pruebas de regresión

Detectan si los cambios introducidos han afectado la funcionalidad existente.

Tipos de pruebas funcionales

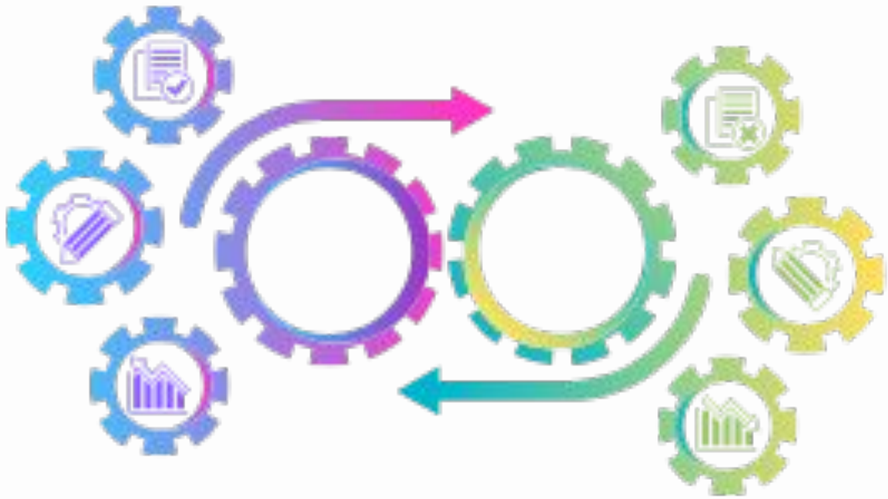
Pruebas unitarias

Las pruebas unitarias son la base de las pruebas funcionales. Se centran en verificar la menor parte testable del software, como funciones y métodos individuales, para asegurarse de que se comporten como se espera. Es fundamental para detectar errores a nivel de componente antes de que el código avance a etapas de integración más complejas.

Se recomienda implementar pruebas unitarias durante las primeras etapas del desarrollo para garantizar una base de código sólida y confiable.

Pruebas de integración

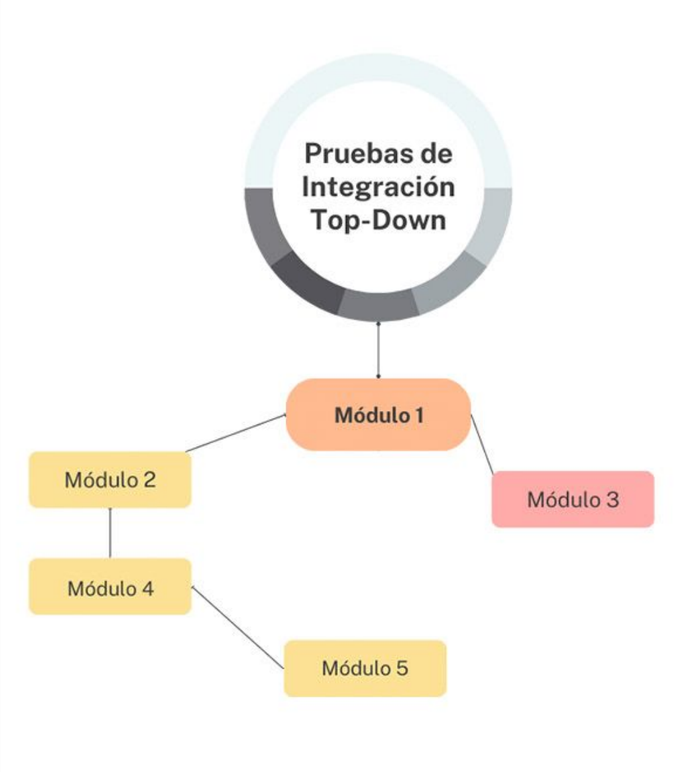
Este tipo de pruebas evalúa la forma en que interactúan y operan varios módulos de aplicaciones de software de forma cohesiva. El sistema se divide en componentes conocidos como módulos o unidades. Cada módulo es responsable de una tarea específica. El verdadero desafío llega cuando combinamos estos componentes para desarrollar todo el sistema de software.



Pruebas de integración

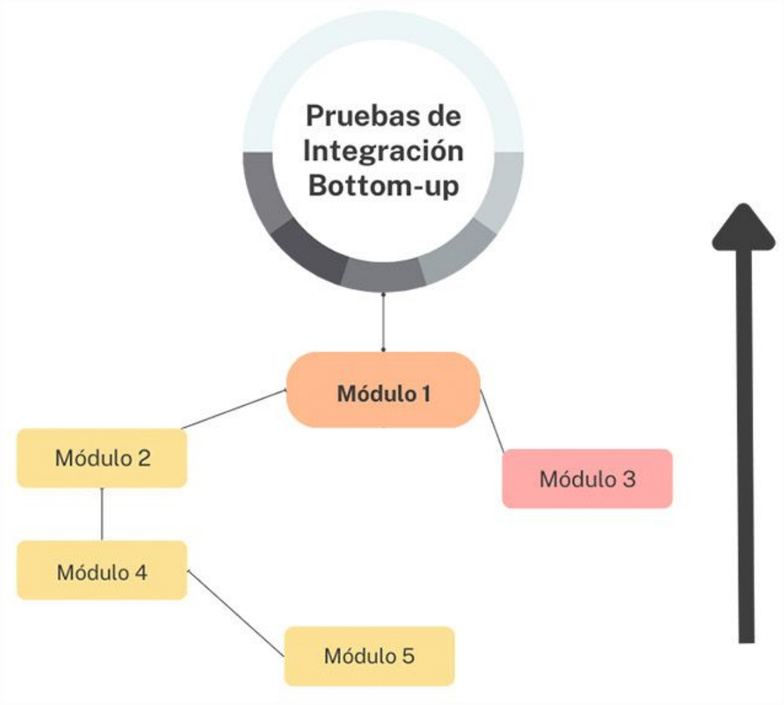
Integración Top-Down

Las pruebas top-down emplean un enfoque sistemático para probar los módulos de software desde el nivel superior hacia abajo a través de la jerarquía del sistema. Las pruebas comienzan con el módulo principal del software y continúan con los submódulos de la aplicación.



Integración Bottom-Up

Cuando se realizan pruebas bottom-up, primero se prueban los módulos de nivel inferior. Se pasa gradualmente a los módulos de nivel superior y así sucesivamente, hasta que todas las facetas del software se han probado a fondo. Esta estrategia se denomina razonamiento inductivo. Resulta beneficiosa cuando se incorporan al producto final componentes ya existentes.



Pruebas Unitarias vs. Pruebas de Integración

Aspecto	Pruebas Unitarias	Pruebas de integración
Alcance	Se concentran en unidades o módulos particulares.	Evaluan la interacción de módulos integrados.
Objetivo	Comprobar que cada elemento funciona de forma independiente.	Comprobar que todas las piezas conectadas entre sí funcionan correctamente.
Interoperabilidad	Las dependencias externas se mantienen separadas de las pruebas.	Se requiere la integración con módulos del mundo real.
Velocidad	Puesto que se concentra en unidades pequeñas, la ejecución es más rápida.	El tiempo de ejecución es un poco mayor debido a las pruebas de los diversos módulos.
Cobertura	Ofrece cobertura extensiva de código al inspeccionar módulos de forma exhaustiva.	Garantiza que los módulos funcionen correctamente como elemento del sistema genera.
Flujo de Trabajo de las Pruebas	Normalmente, los desarrolladores realizan pruebas unitarias antes que las pruebas de integración.	Estas suelen llevarse a cabo durante la fase de integración del software.

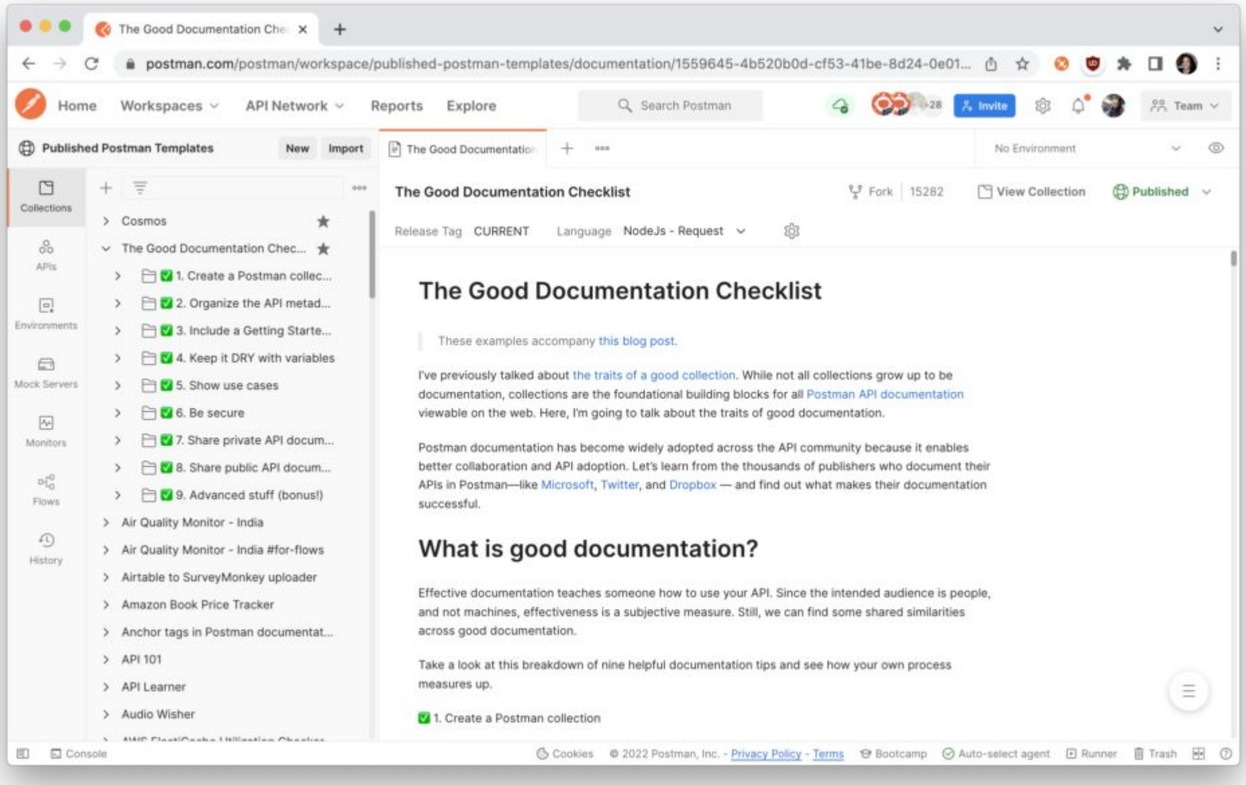
Pruebas de aceptación

Las pruebas de aceptación son pruebas formales que verifican si un sistema satisface los requisitos empresariales. Requieren que se esté ejecutando toda la aplicación durante las pruebas y se centran en replicar las conductas de los usuarios. Sin embargo, también pueden ir más allá y medir el rendimiento del sistema y rechazar cambios si no se han cumplido determinados objetivos.

https://repositorio.uci.cu/handle/ident/8092

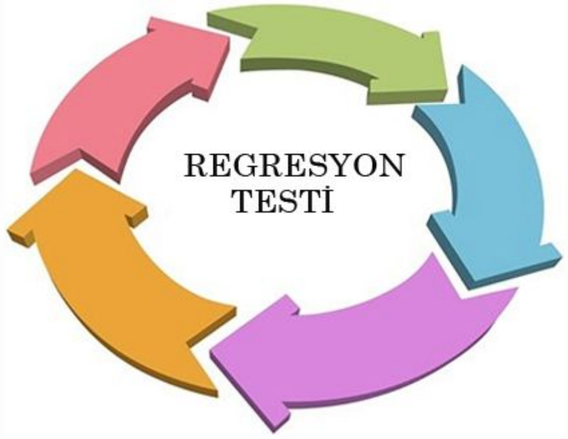
Caso de Prueba Aceptación	
Código: HU9_P10	**Historia de Usuario:** HU_9
Nombre: Listar y eliminar las notificaciones.	
Descripción: prueba para la funcionalidad de Listar las notificaciones.	
Condiciones de Ejecución: El usuario tiene que estar autenticado.	
Entrada/Pasos de Ejecución: una vez autenticado el usuario en el sistema, se debe acceder al submenú "Listar" del menú izquierdo "Notificaciones".	
Resultado Esperado: se creará una vista con el listado de todas las notificaciones que el usuario tiene registrada. Cada notificación dará la opción de ser eliminada. Si se da *click* en el botón eliminar, aparecerá un mensaje para confirmar la acción.	
Evaluación de la Prueba: prueba satisfactoria.	

La Mauske herramienta misteriosa


The Good Documentation Che
postman.com/postman/workspace/published-postman-templates/documentation/1559645-4b520b0d-cf53-41be-8d24-0e01...
Home
Workspaces
API Network
Reports
Explore
Search Postman
Invite
Team
Published Postman Templates
New
Import
The Good Documentatior
No Environment
Collections
APIs
Environments
Mock Servers
Monitors
Flows
History
Cosmos
The Good Documentation Chec...
1. Create a Postman collec...
2. Organize the API metad...
3. Include a Getting Starte...
4. Keep it DRY with variables
5. Show use cases
6. Be secure
7. Share private API docum...
8. Share public API docum...
9. Advanced stuff (bonus!)
Air Quality Monitor - India
Air Quality Monitor - India #for-flows
Airtable to SurveyMonkey uploader
Amazon Book Price Tracker
Anchor tags in Postman documentat...
API 101
API Learner
Audio Wisher
The Good Documentation Checklist
Fork
15282
View Collection
Published
Release Tag
CURRENT
Language
NodeJs - Request
The Good Documentation Checklist
These examples accompany this blog post.
I've previously talked about the traits of a good collection. While not all collections grow up to be documentation, collections are the foundational building blocks for all Postman API documentation viewable on the web. Here, I'm going to talk about the traits of good documentation.
Postman documentation has become widely adopted across the API community because it enables better collaboration and API adoption. Let's learn from the thousands of publishers who document their APIs in Postman—like Microsoft, Twitter, and Dropbox — and find out what makes their documentation successful.
What is good documentation?
Effective documentation teaches someone how to use your API. Since the intended audience is people, and not machines, effectiveness is a subjective measure. Still, we can find some shared similarities across good documentation.
Take a look at this breakdown of nine helpful documentation tips and see how your own process measures up.
1. Create a Postman collection
Console
Cookies
© 2022 Postman, Inc. - Privacy Policy - Terms
Bootcamp
Auto-select agent
Runner
Trash

Pruebas de regresión

Las pruebas de regresión se enfocan en verificar que los cambios, como las correcciones de errores, las mejoras de las funciones y las nuevas funciones añadidas no afecten negativamente las funciones existentes o la funcionalidad. Normalmente se realizan después de un cambio de código relevante o de haber desarrollado una nueva versión del software, y se emplean para asegurar que las funciones existentes sigan respondiendo según lo previsto. Las pruebas de regresión ayudan a garantizar que los nuevos cambios no generen un nuevo error ni provoquen fallos en las funciones existentes.



Ejemplo de prueba de regresión

Cómo hacer Pruebas de Regresión

Veamos cómo realizar Pruebas de Regresión con un escenario del mundo real.

Supongamos que diriges tu propia empresa de desarrollo de software y te asignan un nuevo proyecto. A continuación, tu empresa comienza a implementar el producto basándose en los requerimientos que obtiene de los clientes. Al mismo tiempo, el equipo de pruebas comenzará a crear casos de prueba de acuerdo con los requerimientos del proyecto. Tras analizar los requerimientos, tu equipo de pruebas puede proponer 1000 casos de prueba.

Una vez que la implementación del producto ha finalizado, probarás el producto utilizando esos 1000 casos de prueba y entregarás el producto al cliente si los resultados de las pruebas son satisfactorios. En este punto, es probable que el cliente solicite añadir una nueva función. Ahora, tu equipo de pruebas tiene que escribir casos de prueba para evaluar la nueva función. Digamos que tu equipo de pruebas propone 50 casos de prueba solo para evaluar las funcionalidades de la nueva característica.

Se podría pensar que probar solamente los nuevos 50 casos de prueba es suficiente antes de entregar el producto al cliente. Aquí es donde entran en juego las pruebas de regresión. Cuando tu equipo de desarrollo añada nuevas características al producto, es posible que esto también cause errores en las funciones existentes. Por lo tanto, no solo deberías ejecutar los últimos 50 casos de pruebas sino también los 1000 que creó para el producto original. Sin embargo, en caso de que no tengas el tiempo suficiente para ello, puedes seleccionar solamente los casos de prueba que evalúen las funciones que puedan verse afectadas por el desarrollo de la nueva característica. Este proceso en general se denomina “pruebas de regresión”.

https://qalified.com/es/blog/pruebas-regresion/