

Automatización de pruebas de “software”

**Breve descripción:**

En este componente, el aprendiz identificará las herramientas de pruebas de “software” que serán requeridas al momento de realizar una prueba de automatización, con el objetivo de verificar que la calidad del producto sea óptima y sin errores.

**Abril 2024**

Tabla de contenido

[Introducción 4](#_Toc164988315)

[1. Herramientas de pruebas de “software” por código 5](#_Toc164988316)

[Cypress 5](#_Toc164988317)

[Serenity BDD 7](#_Toc164988318)

[Robot “Framework” 9](#_Toc164988319)

[“TestProject.io” 11](#_Toc164988320)

[WebDriverIO 12](#_Toc164988321)

[2. Herramientas de pruebas de “software” de GUI 13](#_Toc164988322)

[Cómo seleccionar la herramienta de pruebas de “software” de GUI 14](#_Toc164988323)

[3. Desarrollo guiado por pruebas 17](#_Toc164988324)

[Ciclo del desarrollo guiado por pruebas 18](#_Toc164988325)

[4. Calidad del código fuente 22](#_Toc164988326)

[4.1. Métricas de indicadores 22](#_Toc164988327)

[4.2. Caso de prueba herramienta SonarQube 24](#_Toc164988328)

[Inspección continúa 27](#_Toc164988329)

[Síntesis 29](#_Toc164988330)

[Material complementario 30](#_Toc164988331)

[Glosario 31](#_Toc164988332)

[Referencias bibliográficas 32](#_Toc164988333)

[Créditos 35](#_Toc164988334)

Introducción

En este componente formativo, se estudiarán las diferentes herramientas para la automatización de pruebas de “software”, por lo cual, en esta sección, se abordarán las siguientes áreas de aprendizaje: descripción de las herramientas de pruebas de “software” por código y de GUI, TDD y calidad del código fuente.

En este componente aprenderán sobre la importancia de las pruebas automatizadas, las cuales tienen como objetivo principal detectar fallas en el “software” sin la necesidad de intervención manual en cada prueba. A través de este curso, se explicará se elaboran casos de prueba utilizando herramientas de automatización, lo que permite su ejecución automática y simula la interacción humana con el “software”, destacando la automatización como una estrategia clave para reducir el esfuerzo humano en tareas repetitivas o complejas y hacer el proceso más efectivo y productivo.

Además, se abordará la selección de herramientas para pruebas en proyectos de “software” según el contexto de uso, incluyendo desde opciones multifuncionales hasta herramientas de código abierto o de alto costo.

El componente se centra en el análisis de herramientas de automatización de pruebas, tanto para código como para interfaces gráficas, proporcionando una descripción detallada de su desarrollo, conceptos y características, complementado con ejemplos prácticos usando herramientas como Cypress.

# Herramientas de pruebas de “software” por código

Una vez estudiados y comprendidos los fundamentos de la calidad del “software” como elementos clave para elaborar un plan de pruebas, es posible seleccionar las herramientas de pruebas de “software” necesarias para realizar una prueba de automatización, las cuales se describirán en este apartado.

Existen varios “frameworks” que tienen integradas herramientas para la realización de pruebas unitarias, las cuales probarán las funciones de las clases y componentes que son utilizados por el producto “software”. Estos “frameworks” proporcionan ayudas para que el código de automatización de prueba sea estable, reutilizable y mantenible (Serna, M. y Serna, A., 2012).

### Cypress

Es un “framework” de automatización de pruebas para JavaScript que facilita el testeo de aplicaciones web. Destaca por la sencillez de su configuración, la velocidad en la ejecución de pruebas y su interfaz gráfica interactiva, que muestra con claridad lo que está sucediendo y lo que se está probando. Esto permite detectar rápidamente cualquier error que surja durante el proceso. Además, es importante mencionar que con Cypress se pueden escribir pruebas de todo tipo, incluyendo unitarias, de integración y end-to-end, sin necesidad de combinar distintos “frameworks” de prueba. Esto se debe a que Cypress es compatible con cualquier tipo de “framework”, lo que facilita enormemente el proceso de prueba.

Este “framework” de automatización de pruebas se enfoca en el desarrollador y se centra en permitir que el desarrollo basado en pruebas (TDD) sea una práctica simple y agradable. Cypress se ejecuta dentro del navegador de forma remota, a diferencia de otros “framework” de pruebas que se ejecutan fuera de él, considerando que tiene una arquitectura diferente y brindando resultados más consistentes. Por lo tanto, se hace una invitación a conocer el videotutorial de la instalación de la herramienta Cypress en el sistema operativo Windows 10 a 64 bits.

1. Automatización de pruebas de “software”



[**Enlace de reproducción del video**](https://www.youtube.com/watch?v=jZNkSRk4jxM)

|  |
| --- |
| **Síntesis del video: Automatización de pruebas de “software”** |
| Video tutorial del experto donde se explica la instalación del “framework” de automatización de pruebas de Cypress, desde la descarga de Node js y en adelante el paso a paso para la configuración de manera práctica, además del uso de Cypress a través de ejemplos para ver su funcionamiento y la orientación para la descarga e instalación de Visual Studio Code para Windows, el uso de librerías de código y la solución de posibles errores. |

Para afianzar sus conocimientos y realizar los pasos de instalación de la herramienta Cypress, se le invita a descargar el siguiente archivo PDF:

**Guía para instalar Cypress**

Para una instalación exitosa de Cypress, le invitamos a revisar detenidamente la guía que se encuentra en el archivo PDF “Guia para instalar Cypress”, el cual se encuentra en la carpeta Anexos.

Para utilizar este “framework”, se deben tener en cuenta las características del entorno donde será instalado y ejecutado; por lo tanto, a continuación, se mencionan los elementos requeridos:

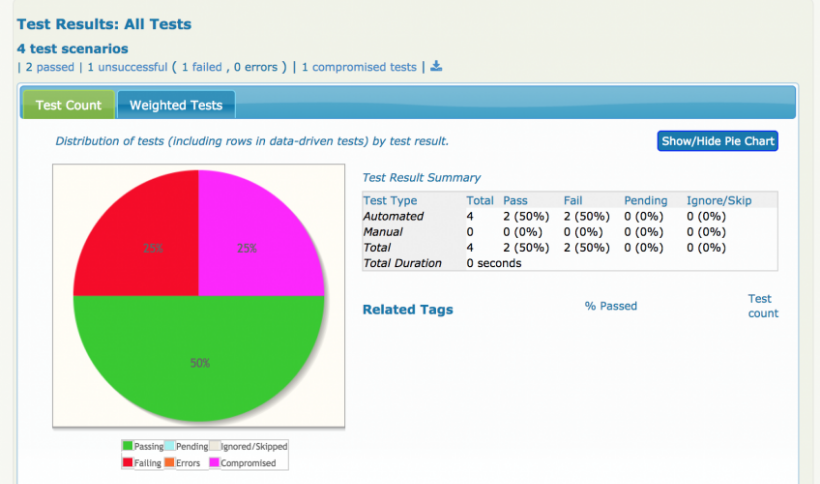
1. Requerimientos Cypress

| Requisitos | Descripción |
| --- | --- |
| “Hardware” | Portátil o computador de escritorio (Procesador 2.4GHz, 4GB RAM, 320 GB HD). |
| Navegadores | Internet Explorer 7.0.6001.1800, Mozilla Firefox 3.0.6, Opera 9.63, Safari 3.2.1, Google Chrome. |
| Editor de código | Visual Studio Code. |

### Serenity BDD

Es un “framework” de automatización de casos de prueba de código abierto basado en Java, que se integra con herramientas de desarrollo como JBehave y Cucumber, diseñado para facilitar la creación de pruebas de regresión y pruebas automáticas de aceptación. Ofrece una amplia gama de funcionalidades, incluyendo la ejecución de pruebas en paralelo, la captura de pantallas y la gestión de WebDriver, todo ello sin necesidad de escribir código adicional. Recientemente, Serenity ha ampliado sus capacidades integrándose con REST Assured, una biblioteca de Java para el testeo de API de código abierto, enriqueciendo así su oferta para los desarrolladores y QA “testers” que buscan eficiencia y efectividad en la automatización de pruebas.

1. “Framework” Serenity BDD



Para utilizar Serenity BDD, se deben tener en cuenta las características del entorno donde será instalado y ejecutado, por lo tanto, a continuación, se mencionan los elementos requeridos:

1. Requerimientos Serenity BDD

| Requisitos | Descripción |
| --- | --- |
| “Hardware” | Portátil o computador de escritorio (Core Duo P8600 2.4GHz, 4GB RAM, 320 GB HD). |
| Sistema operativo | Windows Vista Ultimate, Windows 10, Linux. 32 bits o 64 bits. |
| Navegadores | Internet Explorer 7.0.6001.1800, Mozilla Firefox 3.0.6, Opera 9.63, Safari 3.2.1, Google Chrome. |
| Compilador de código | Apache Maven 3.5.0 |
| Lenguaje de programación | Java 1.8. |
| Kit de desarrollo de Java | Java Development Kit JDK 8 o superior. |
| Editor de código | Intellij Idea CE. |

**Conociendo Serenity BDD**

Para conocer algunas funcionalidades de este “framework”, se recomienda observar el siguiente webinar. <https://www.youtube.com/watch?v=609y4K4ph6s>

### Robot “Framework”

Este “framework” se basa en Python y también podría utilizar IronPython (.NET) y Jython (Java). Además, emplea un enfoque basado en palabras clave para facilitar la creación de pruebas que sean fáciles de entender y legibles. Contiene diversas bibliotecas de pruebas y otras herramientas que se utilizan para agregar nuevas funcionalidades y desarrollar un conjunto de pruebas personalizado para la aplicación. Por otro lado, la biblioteca de prueba externa más utilizada podría ser Selenium WebDriver. No obstante, Robot “Framework” puede probar FTP, MongoDB, Android y Appium, además de sitios web.

Este “framework” se utiliza para las pruebas de aceptación, se considera entre uno de los mejores “framework” de Python. Robot “Framework” se utiliza en Python, pero, también puede ejecutarse en IronPython, basado en .net y en Jython, donde este último está basado en el lenguaje de programación Java.

Para utilizar Robot “Framework”, se deben tener en cuenta las características del entorno donde será instalado y ejecutado. A continuación, se mencionan los elementos requeridos:

1. Requerimientos Robot “Framework”

| Requisitos | Descripción |
| --- | --- |
| “Hardware” | Portátil o computador de escritorio (Procesador 2.4GHz, 4GB RAM, 320 GB HD). |
| Sistema operativo | Windows 10, 64 bits |
| Navegadores | Internet Explorer 7.0.6001.1800, Mozilla Firefox 3.0.6, Opera 9.63, Safari 3.2.1, Google Chrome. |
| Lenguaje de programación | Python 2.7.9 o superior. |
| Editor de código | SublimeText. Página para instalar <https://www.sublimetext.com/3> |
| Página para descargar | Python <https://www.python.org/downloads/windows/> wxPython. <https://wxpython.org/pages/downloads/index.html> |

**Robot “Framework”: preparando ambiente**

Para familiarizarse con las funcionalidades de este “framework”, se sugiere seguir el siguiente tutorial en video. <https://www.youtube.com/watch?v=x1klhk_ovZo>

### “TestProject.io”

Es una de las primeras plataformas de automatización de pruebas basadas en la nube, lo que facilita que los evaluadores realicen su trabajo de manera eficiente, utilizando “frameworks” de código abierto como Selenium y Appium. Al ser una herramienta de código abierto, permite que una mayor cantidad de empresas y “testers” se beneficien de las capacidades de Selenium y Appium, que son dos de los principales recursos de código abierto para la automatización. Por otro lado, ya están disponibles las implementaciones para Java y C#, y actualmente se está trabajando en la compatibilidad con JavaScript, Python y otros lenguajes de programación.

Para utilizar “TestProject.io”, se deben tener en cuenta las características del entorno donde será probado, por lo tanto, a continuación, se mencionan los elementos requeridos:

1. Requerimientos “TestProject”

|  |  |
| --- | --- |
| Requisitos | Descripción |
| “Hardware” | Portátil o computador de escritorio (Procesador 2.4GHz, 4GB RAM, 320 GB HD). |
| Sistema operativo | Windows 10, 64 bits. |
| Navegadores | Internet Explorer 7.0.6001.1800, Mozilla Firefox 3.0.6, Opera 9.63, Safari 3.2.1, Google Chrome. |

**“Test” con “TestProject”**

Para conocer algunas funcionalidades de la plataforma en automatización de pruebas, remítase al video tutorial. <https://www.youtube.com/watch?app=desktop&v=hbMbcjm492c>

### WebDriverIO

Este “framework” es muy completo para probar aplicaciones web. Es muy rápido y no necesita implementar tareas desde cero. De acuerdo con la filosofía “click and run”, contiene todo lo requerido para crear gran variedad de pruebas sostenibles y escalables, y contiene características adicionales que no se encuentran en otros “frameworks”.

Para utilizar este “framework”, se deben tener en cuenta las características del entorno donde será instalado y ejecutado; por lo tanto, a continuación, se mencionan los elementos requeridos:

1. Requerimientos WebDriverIO

|  |  |
| --- | --- |
| Requisitos | Descripción |
| “Hardware” | Portátil o computador de escritorio (Procesador 2.4GHz, 4GB RAM, 320 GB HD). |
| Sistema operativo | Windows 10, 32 o 64 bits. |
| Navegadores | Internet Explorer 7.0.6001.1800, Mozilla Firefox 3.0.6, Opera 9.63, Safari 3.2.1, Google Chrome. |
| Editor de código | Visual Studio Code. Página para descargar Visual Studio Code <https://code.visualstudio.com/download> |
| Entorno en tiempo de ejecución multiplataforma | Node.js |
| Página para descargar | node.js <https://nodejs.org/en/> |

**Webdriver**

Para conocer algunas funcionalidades de Webdriver, se recomienda explorar los siguientes videos tutoriales. <https://www.youtube.com/watch?v=Rv28n_rT1BM>

# Herramientas de pruebas de “software” de GUI

En este apartado, se identifican las herramientas de prueba de GUI (Graphical User Interface), que en español se traduce como interfaz gráfica de usuario, las cuales ayudan en las pruebas automatizadas de “software” y aplicaciones que contienen interfaces gráficas de usuario. Estas herramientas de “software” emplean casos de prueba incluidos en el plan de pruebas para garantizar que la aplicación cumpla con sus especificaciones y ejecute las funciones previstas.

Las herramientas de prueba de GUI verifican la funcionalidad y evalúan otros elementos de diseño, como el tamaño de fuente, botones, cuadros de prueba, cuadros de diálogo, formato de texto, barras de menú, barras de herramientas, contenido de enlaces, listas, imágenes, íconos, diseño, entre otros.

Por otro lado, cuando se incluyen pruebas de GUI al comienzo del proceso de desarrollo del “software”, mejora la calidad, se agiliza el desarrollo y se reducen los riesgos al finalizar el proceso de desarrollo; además, las pruebas de GUI automatizadas proveen una solución para los problemas relacionados con las pruebas manuales de GUI (Fernández, 2018).

**Automatización de pruebas de “software”**

Los invitamos a ingresar al webinar "Automatización de Pruebas de “Software”" donde exploraran las últimas tendencias y herramientas. <https://www.youtube.com/watch?v=xgJcWRQhDmY>

¿Sabías que ... las pruebas de GUI, una vez creadas, pueden repetirse tantas veces como sea necesario?

### Cómo seleccionar la herramienta de pruebas de “software” de GUI

Las pruebas de “software” de GUI constituyen un proceso funcional que puede volverse crítico. Por lo tanto, hay muchos factores a considerar al seleccionar una herramienta de pruebas de GUI automatizada. A continuación, se mencionan algunos de los factores importantes a tener en cuenta durante el proceso de selección:

* **Curva de aprendizaje**

Debe ser acorde con el alcance y los objetivos.

* **Disponibilidad de soporte**

Bien sea foros, bases de datos de errores, blogs y comunidad activa de usuarios.

* **Evaluadores**

Deben verificar el factor de costo, mantenimiento, costos de licencias por usuario.

* **Sistema operativo**

Sistema operativo compatible con aquellos paquetes necesarios para una implementación exitosa.

* **Cociente de integración**

Verificar que la herramienta incluye alguna API o tiene capacidad de personalización.

* **Correcciones regulares/ lanzamientos**

Observar si la herramienta se mantiene, las actualizaciones periódicas se lanzan y se verifica la madurez de la herramienta.

Con base en lo anterior, a continuación, se describen las herramientas para crear y ejecutar pruebas de GUI automatizadas de forma efectiva:

* **“Test Complete”**

Es una herramienta que contiene tres módulos para aplicaciones de escritorio, web y móviles, una opción de grabación manual, reproducción y modificación automatizada de prueba grabada.

* **“Selenium”**

Es una herramienta que es utilizada para la automatización de pruebas, las efectúa mediante reproducción y grabación de una aplicación web de código abierto, permite grabar acciones visuales.

* **“Squish”**

Es una herramienta de uso comercial, multiplataforma y funcional, utilizada para pruebas de regresión y de GUI, con la capacidad de probar aplicaciones basadas en diversas tecnologías, incluyendo Windows, Java, Mac, iOS, Android, etc.

Según las herramientas utilizadas para ejecutar las pruebas de GUI, se destaca el uso de “frameworks”. Es crucial considerar las características del entorno donde serán instalados y ejecutados:

**Test complete**

* Sistema operativo: Microsoft Windows XP Professional de 32/64 bits.
* Procesador: Intel Core 2 Duo 2 GHz o superior.
* RAM: 2 GB de RAM en otros sistemas operativos.
* Disco duro: 1 GB de espacio libre en disco para la instalación.
* Resolución: Resolución de pantalla de 1024 × 768 o superior.

**Selenium**

* Sistema operativo: Microsoft Windows 10 32/64 bits.
* Procesador: Intel Core 2 Duo 2 GHz o superior.
* RAM: 4 GB de RAM en otros sistemas operativos.
* Disco duro: 2 GB de espacio libre en disco para la instalación.

**Squish**

* Sistema operativo: Microsoft Windows 7 SP1 o superior 32/64 bits.
* Procesador: Intel Core 2 Duo 2 GHz o superior.
* RAM: 4 GB de RAM
* Disco duro: 2 GB de espacio libre en disco para la instalación.

# Desarrollo guiado por pruebas

Una vez seleccionadas las herramientas de pruebas, es posible optar por realizar pruebas unitarias mediante la técnica de Desarrollo Dirigido por Pruebas (TDD). Es importante señalar que este enfoque es relativamente reciente en el proceso de desarrollo, empezando a consolidarse y a ser reconocido globalmente en el ámbito del desarrollo de “software” por organizaciones, basándose en evidencia anecdótica y diversas evaluaciones empíricas. A diferencia del enfoque tradicional, donde las pruebas se emplean para validar y verificar la calidad del “software” después de desarrollar la funcionalidad, TDD utiliza las pruebas tanto para la validación y verificación como para la especificación, desarrollando primero la prueba y luego la funcionalidad. (Catalán, 2017).

En el enfoque TDD, las pruebas no solo sirven para la validación y verificación, sino también como especificación. Primero se desarrolla la prueba y luego la funcionalidad. Este proceso contribuye a mejorar la calidad del “software” y la productividad del desarrollador, ya que se optimiza el esfuerzo en depuración y se facilita un avance más ágil en el desarrollo del producto de “software”. A continuación, se presentan los beneficios:

* **Beneficios clave de adoptar esta metodología**

En el dinámico mundo del desarrollo de “software”, el Desarrollo Dirigido por Pruebas (TDD) destaca por su enfoque práctico y eficaz. A continuación, desglosamos los beneficios clave de adoptar esta metodología, que van desde la mejora de la calidad del producto hasta la optimización del rendimiento del código.

* **Verificación de funcionalidades desarrolladas**

Facilita la verificación de funcionalidades desarrolladas en ciclos más cortos, lo que promueve la detección temprana de errores y su correspondiente solución.

* **Adoptar esta metodología mejora la fiabilidad del producto de “software”**

Adoptar esta metodología mejora la fiabilidad del producto de “software”, no solo en términos de corrección de errores funcionales, sino también en cumplir con las necesidades especificadas por el cliente.

* **Optimización y mejora del rendimiento del código**

Optimización y mejora del rendimiento del código, ya que se enfatiza en escribir únicamente el código necesario para satisfacer las necesidades identificadas.

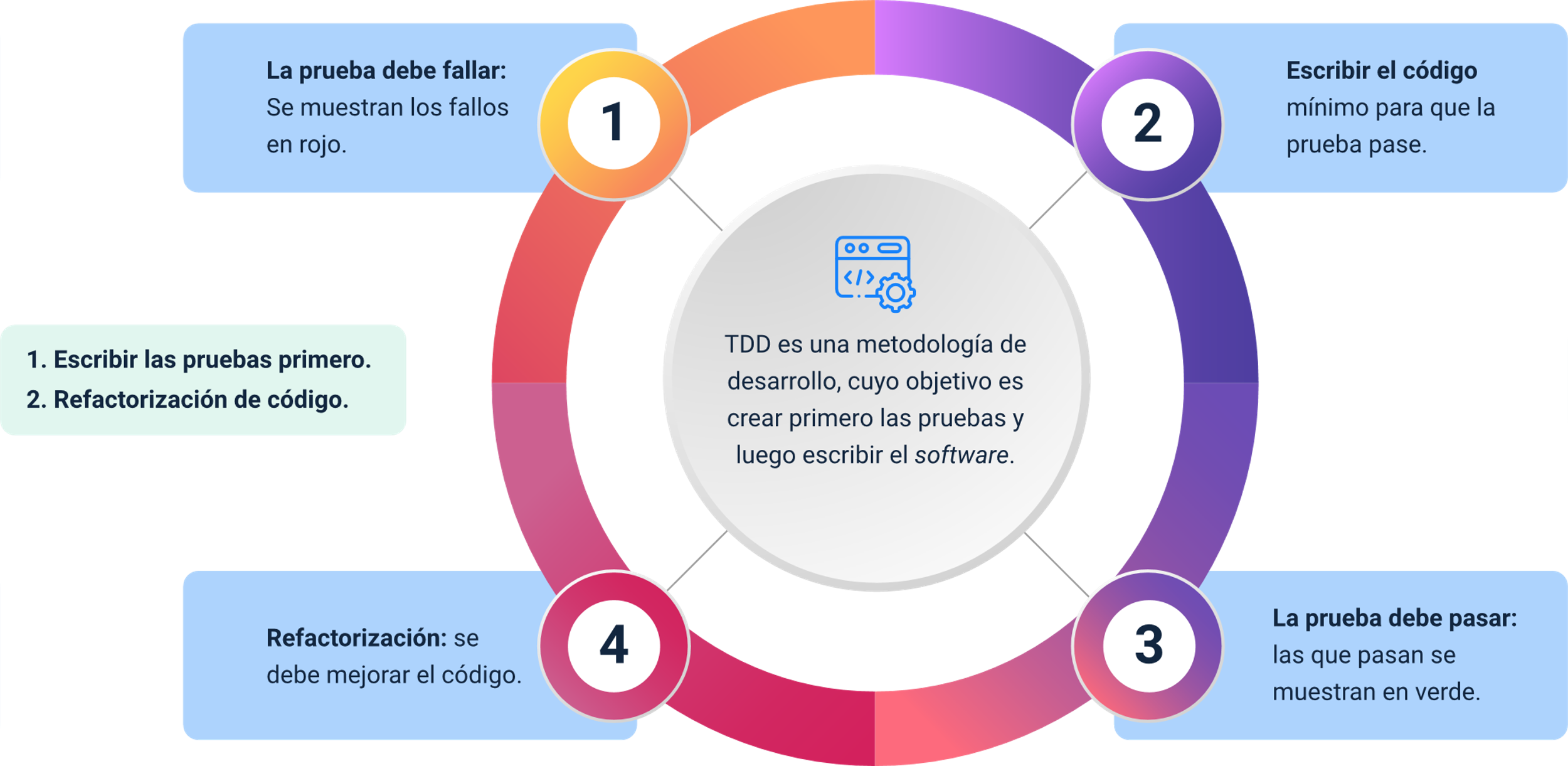
* **Simplifica el desarrollo de manera modular**

El proceso de ejecutar pruebas para cada nueva funcionalidad o requisito simplifica el desarrollo de manera modular, facilitando la escalabilidad y el mantenimiento futuro del producto de “software”.

### Ciclo del desarrollo guiado por pruebas

Ahora que ya conoce los beneficios que implica aplicar TDD, observe el ciclo completo del proceso, revisando cada uno de sus pasos. En la siguiente imagen se puede observar el flujo de manera gráfica:

1. Ciclo TDD



A continuación, se describirán los pasos involucrados en realizar el proceso de TDD.

1. **Escribir la prueba**

En primera instancia, se deben crear las pruebas y estas deben atender cada una de las nuevas funcionalidades o características definidas en los requisitos de “software”. En este primer paso, es muy importante que el desarrollador comprenda a la perfección la especificación de requisitos, dado que la discrepancia entre lo que entendió y lo que el producto “software” requiere impactará la calidad de la prueba y, asimismo, la calidad del producto.

1. **Ejecución y comprobación de “test”**

En este paso, se realiza la verificación de que los “tests” funcionan correctamente, por lo cual, detectarán errores cuando haya un problema en el código. Uno de los principales aportes de este paso es la confianza que el desarrollador adquiere al momento de desarrollar nuevas pruebas, lo cual se verá reflejado en la calidad del código.

1. **Escribir el código de la aplicación**

En este paso, se procede a escribir el código fuente de la funcionalidad. El objetivo es que la prueba escrita para esta funcionalidad anteriormente pase como exitosa. Es probable que el código no esté perfecto y no cumpla todavía con estándares, pero posteriormente hay un paso que permite realizar este tipo de ajustes, donde ya se tiene una visión más amplia del desarrollo en general y se puede optimizar.

1. **Ejecutar los “tests”**

Dado que ya se tiene implementado el código fuente, se procede a ejecutar las pruebas que fueron implementadas en el primer paso. Si las pruebas pasan exitosamente, se puede confiar en la calidad del código. En el caso de que las pruebas no sean exitosas, es el momento de regresar al código fuente y realizar los ajustes necesarios para que todas las pruebas pasen.

1. **Refactorizar el código**

En este paso, ya se dispone de un código fuente con calidad, dado que se pasaron todas las pruebas y se procederá a realizar un análisis de un mejoramiento en el mismo. En este paso, el objetivo es mejorar la escritura para que se pueda interpretar mejor, estructurar para reducir la cantidad de líneas, optimizar el nombrado de variables, constantes y clases.

1. **Repetir**

Cuando se haya completado el ciclo para una funcionalidad, requisito o historia de usuario, se deberá repetir todo el proceso.

**Desarrollo Dirigido por Pruebas (TDD)**

Lo invitamos a ingresar al video tutorial que contiene el ciclo TDD, así como un ejemplo práctico de este proceso. <https://www.youtube.com/watch?v=YuRdaR6wwWU>

# Calidad del código fuente

Este aspecto es crucial para analizar, dado que la calidad del código fuente y las métricas relevantes facilitan la identificación de errores y su posterior refactorización. Por lo tanto, el análisis de código constituye una etapa esencial del ciclo de desarrollo de “software”, que implica la revisión del código fuente para detectar errores en una etapa temprana (Dapozo et al., s.f.).

Se suele establecer un proceso de verificación de código antes de integrarlo al código base. Por ello, el código fuente debe ser redactado de manera que cumpla con sus funciones de forma eficaz y que, adicionalmente, sea sencillo de entender y modificar por otros desarrolladores.

## Métricas de indicadores

En el ciclo de vida del desarrollo de “software”, la construcción del código fuente representa un proceso fundamental, basado en un diseño arquitectónico detallado que incorpora estilos arquitectónicos, patrones y estándares de codificación. Estos elementos garantizan la mantenibilidad del código, la capacidad para desarrollar nuevas funcionalidades y la facilidad para identificar y corregir errores. Este proceso se sustenta en la aplicación de diversas métricas, las cuales se detallarán a continuación:

* **Comentarios en el código**

Está relacionado con la utilización excesiva de comentarios de código. Antiguamente, se generaban gran cantidad de comentarios, pero ahora se considera realizar buenas prácticas de nomenclatura para explicar secciones lógicas complicadas.

Cómo usarlo: los comentarios deben ser claros y concisos.

* **Líneas de código**

Hace referencia a métodos, clases, módulos y bloques muy extensos; esto indica que una parte del código contiene muchos comportamientos y responsabilidades.

Cómo usarlo: se deben encontrar métodos y clases muy extensos; si esos bloques tienden a confundir, se puede colocar en contacto con un desarrollador y, después, se crean y ejecutan sesiones de prueba.

* **Acoplamiento**

Los componentes con varios acoplamientos corren riesgo de dos formas distintas.

* Si varios componentes o clases dependen del que se está examinando, hay modificaciones.
* Si el componente o clase actual depende de varios componentes, puede causar problemas y afectar el actual

Cómo usarlo: se debe encontrar, en cualquier dirección, un acoplamiento alto y establecer cambios notables en los componentes externos y observar cómo responde el componente actual.

* **Rotación**

Es la cantidad de ediciones en un archivo de código, que generalmente se calcula por la cantidad de confirmaciones al control de código fuente.

Cómo usarlo: se debe encontrar una alta rotación en algún área; posteriormente, realizar un análisis detallado sobre estos módulos, clases, y paquetes.

* **Complejidad**

Mide la complejidad estructural del código. Se crea al calcular el número de caminos de código diferentes dentro del flujo del programa. Cuantos más caminos, más complicado es mantener y aprobar el código.

Cómo usarlo: emplear un tiempo y sobreponer las métricas de rotación con las áreas que tienen más complejidad.

## Caso de prueba herramienta SonarQube

SonarQube es una plataforma de código abierto destinada a la gestión de calidad de código, que utiliza herramientas de análisis estático de código fuente, como Checkstyle, PMD o FindBugs, para generar métricas que contribuyen a mejorar la calidad del código de una aplicación.

El objetivo de presentar esta herramienta de revisión automática de código es identificar bugs, vulnerabilidades y "code smells" en el código. Además, es compatible con varios lenguajes de programación. A continuación, se enumeran algunas de las herramientas empleadas en la gestión del código:

* **Checkstyle**:

Analiza un código fuente y busca mejorar el estándar de codificación, también verifica el código fuente para las convenciones de codificación, como encabezados, importaciones, espacios en blanco.

* **PMD**:

Identifica problemas potenciales, principalmente, códigos muertos, duplicados, complejidad ciclomática y expresiones excesivamente complejas. (De la Roca et al., 2014).

* **FindBugs**:

Analiza el código para encontrar cualquier falla de diseño y posibles errores, identifica mala práctica, código dudoso, corrección multiproceso, rendimiento malicioso, vulnerabilidad de código, seguridad experimental e internacionalización (De la Roca et al., 2014).

SonarQube ofrece una vista instantánea de la calidad del código en un momento específico, así como un historial de la evolución de esta calidad a lo largo del tiempo. Proporciona indicadores cuantitativos tales como la cantidad de código cubierto por las pruebas unitarias, la complejidad ciclomática del código y la duplicación. Esto permite medir la excelencia del proyecto y la calidad de los desarrolladores que lo mantienen.

Es importante destacar que, desde las perspectivas de:

* **Ingeniero de pruebas**

SonarQube facilita la identificación de áreas donde faltan pruebas unitarias, permitiendo su inclusión.

* **Arquitecto de “software”**

SonarQube asiste en mantener el diseño original del sistema, asegurando que las reglas de codificación se cumplan durante el desarrollo del proyecto.

Ayuda a detectar código excesivamente complejo para su posterior refactorización y a identificar dependencias incorrectamente asignadas.

Puede asegurar atributos de calidad como la mantenibilidad, la extensibilidad y el bajo acoplamiento.

* **Desarrolladores**

Les permite escribir código más legible y fácil de mantener, lo cual se traduce en una curva de aprendizaje mucho más baja para los nuevos desarrolladores que llegan a incluirse en el proyecto.

Por lo tanto, es posible mencionar algunas de las características a tener en cuenta de SonarQube:

**Características de SonarQube**

* Integración DevOps (Jenkins & Azure DevOps).
* Detecta problemas complicados.
* Se integra con IDEs de desarrollo con herramientas de integración continua.
* Analiza el código fuente de más de 20 lenguajes de programación.
* Instala un servidor local que se puede configurar con diferentes BD para analizar grandes cantidades de código o también trae su BD embebida.
* Curva de aprendizaje rápida.
* Escalable a las necesidades de una empresa.
* Inspección continúa.

Para analizar un proyecto de “software” utilizando SonarQube, se debe contar con tres componentes:

* **Servidor web de SonarQube**

Es un servidor web que tarda un par de minutos al instalarse por primera vez, ya que debe crear las tablas en la base de datos para ser ejecutado.

* **Base de datos**

Es un conjunto de datos que permite almacenar el resultado de todos los análisis que se van a ejecutar.

* **Scanner**

Existen diferentes tipos de scanner, dependiendo del lenguaje, y es quien realizará el análisis y subirá los reportes al servidor web de SonarQube.

Una vez el reporte de SonarQube se ha generado, el equipo de desarrollo de “software” podrá entrar y analizar el reporte para saber cómo mejorar la calidad a medida que va evolucionando el proyecto.

### Inspección continúa

En la construcción de un “software”, hay muchos desarrolladores involucrados, cada uno de ellos con una comprensión diferente de lo que es la calidad y los estándares del desarrollo de “software”. Adicionalmente, pueden utilizar editores de código diferentes, como visual Studio Code o IntelliJ. Posteriormente, suben el código a algún repositorio, como uno basado en Git. Después de que esto ocurre, se lanza un proceso de integración continua, puede ser en Jenkins, donde se ejecutan las pruebas del proyecto y este se puede desplegar en un ambiente de producción. Es en este punto donde se puede incluir SonarQube para mejorar la calidad del proyecto, puesto que se puede hacer que SonarQube haga parte del proceso de integración continua y, con esto, se estaría creando el proceso de inspección continua.

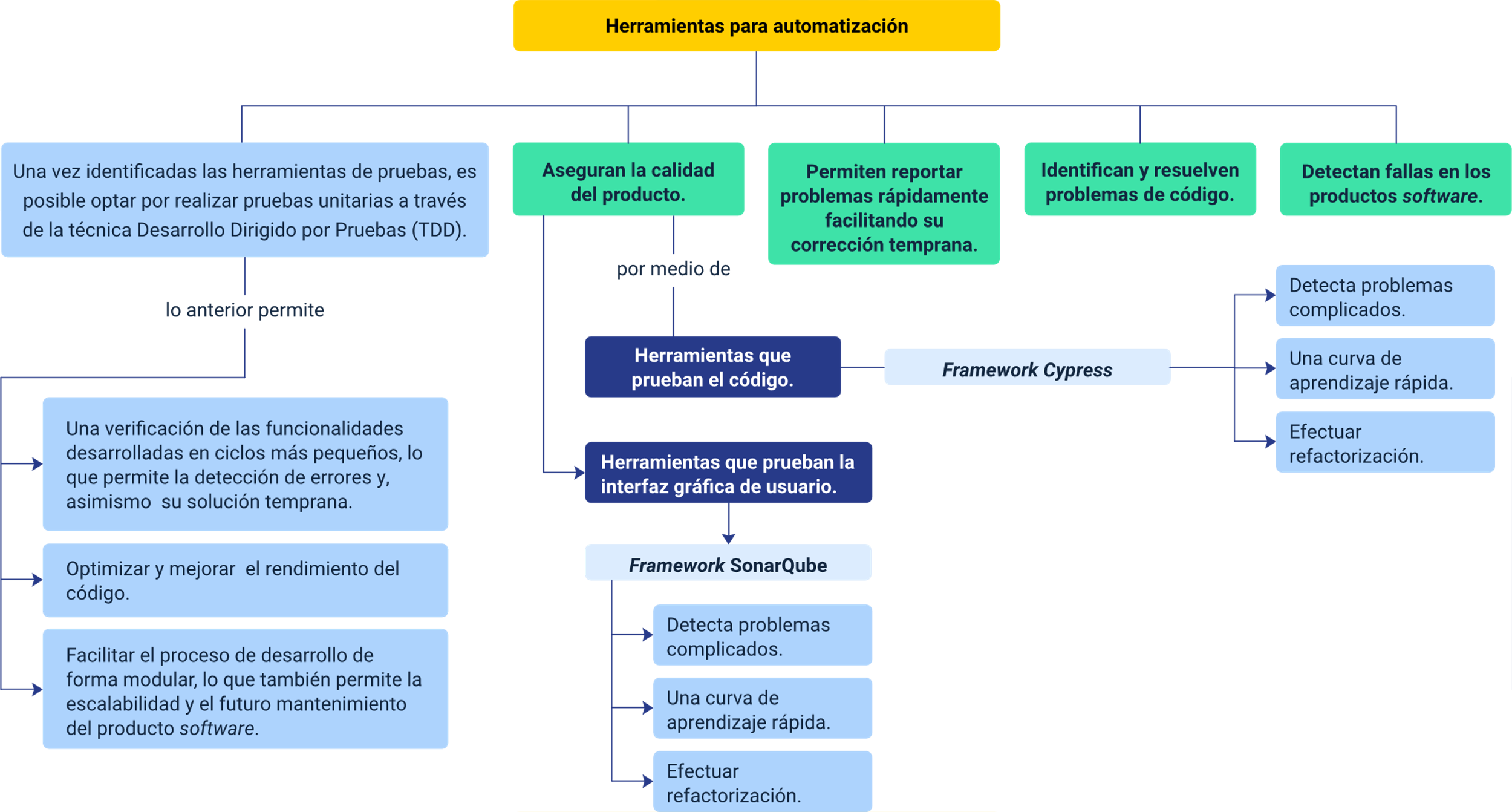
Una vez que el análisis del código está listo, los desarrolladores podrán entrar y ver el resultado del reporte para seguir trabajando en el mejoramiento continuo de la calidad del proyecto. Finalmente, es posible instalar SonarLint, que es un plugin para los editores de código, que permite realizar un análisis antes de subir el código al servidor de SonarQube para ser analizado, de tal forma que los desarrolladores mejoren la calidad con mucho tiempo de antelación.

**Probando SonarQube**

Para obtener una mejor comprensión de SonarQube, se recomienda acceder al siguiente video tutorial, el cual ofrece una guía detallada sobre cómo instalar la herramienta SonarQube. <https://www.youtube.com/watch?v=5UoygWLrBqo>

Síntesis

Las herramientas de automatización detectan fallas en los productos de “software”, identifican y resuelven problemas de código, y facilitan el reporte rápido de problemas, permitiendo su corrección temprana. Las pruebas automatizadas aseguran la calidad del producto mediante herramientas que examinan el código y las interfaces gráficas de usuario. Entre estas herramientas, Cypress y SonarQube se destacan por la facilidad de configuración, su interfaz gráfica interactiva, la detección de problemas complejos y la capacidad para realizar refactorizaciones. Además, al identificar herramientas de pruebas, es posible optar por pruebas unitarias mediante la técnica de desarrollo dirigido por pruebas (TDD), la cual facilita la verificación de funcionalidades desarrolladas, optimiza y mejora el rendimiento del código. Para una mejor interpretación de estas herramientas de automatización, se recomienda consultar el siguiente mapa conceptual.



Material complementario

| Tema | Referencia | Tipo de material | Enlace del recurso |
| --- | --- | --- | --- |
| Calidad del código fuente | Dapozo, G., Greiner, C., Irrazabal, E. y Chiapello, J. (s. f.). Análisis de código fuente orientado a la calidad del producto. Universidad Nacional del Nordeste. | Artículo | <http://sedici.unlp.edu.ar/bitstream/handle/10915/45740/Documento_completo.pdf?sequence=1&isAllowed=y> |
| Automatización de pruebas de “software” | Mascheroni, M., Cogliolo, M. e Irrazabal, E. (2016). Automatización de pruebas de compatibilidad web en un entorno de desarrollo continuo de “software”. 17º Simposio Argentino de Ingeniería de “Software”. | Tesis | <http://sedici.unlp.edu.ar/bitstream/handle/10915/57081/Documento_completo.pdf-PDFA.pdf?sequence=1&isAllowed=y> |
| Herramientas de pruebas de “software” por código | Serna, E. y Serna, A. (2012). Una evaluación a las herramientas libres para pruebas de “software”. Revista Virtual Universidad Católica Del Norte, (37), p. 44–61. | Documento | <https://revistavirtual.ucn.edu.co/index.php/RevistaUCN/article/view/387> |
| Herramientas de pruebas de “software” de GUI | Fernández, S. (2018). Herramientas de automatización para pruebas de “software”. Universidad Técnica Federico Santa María. | Tesis | <https://repositorio.usm.cl/bitstream/handle/11673/47825/3560900259665UTFSM.pdf?sequence=1&isAllowed=y> |

Glosario

**Acoplamiento**: es la medida en que un componente o clase dependen de otro, generando cambios externos o alterando la funcionalidad del mismo (cuando se modifican los atributos de una clase, se tienen que modificar los atributos de otra).

**Cohesión**: es la medida en la que un componente o clase realiza únicamente la tarea para la cual fue diseñado (una clase debe de hacer lo que respecta a su entidad y no hacer acciones que involucren a otra clase o entidad).

**FindBugs**: analiza el código para encontrar cualquier falla de diseño y posibles errores.

**GitHub**: es un servicio basado en la nube que aloja un sistema de control de versiones (VCS) llamado Git. Este permite a los desarrolladores colaborar y realizar cambios en proyectos compartidos, a la vez que mantienen un seguimiento detallado de su progreso.

**SOLID**: se compone de una serie de principios aplicables a la Programación Orientada a Objetos que, si se utilizan correctamente, ayudarán a escribir “software” de calidad en cualquier lenguaje de programación orientada a objetos. Gracias a ellos, se podrá crear código que será más fácil de leer, testear y mantener.

**TDD**: permite una verificación de las funcionalidades desarrolladas en ciclos más pequeños, lo que permite la detección de errores y, asimismo, su solución temprana.

Referencias bibliográficas

Abel Quintana Lopez. (2020). Automatizando pruebas desde cero - WebdriverIO - Desafío#1 - Solución. [Archivo de video] Youtube. <https://www.youtube.com/watch?v=fnTvy_ZEcgM&t=337s>

Abel Quintana Lopez. (2020). WebdriverIO - Desafío # 1. [Archivo de video] Youtube. <https://www.youtube.com/watch?v=Rv28n_rT1BM>

Alvarez, N. (2019). Cap #2 | Curso RobotFramework| Preparando ambiente. [Archivo de video] Youtube. <https://www.youtube.com/watch?app=desktop&v=x1klhk_ovZo>

Dapozo, G., Greiner, C., Irrazabal, E. y Chiapello, J. (s. f.). Análisis de código fuente orientado a la calidad del producto. Universidad Nacional del Nordeste. <http://sedici.unlp.edu.ar/bitstream/handle/10915/45740/Documento_completo.pdf?sequence=1&isAllowed=y>

De la Roca, A., Santa, L., Estrada, A., Aranda, B. y Villavicencio, L. (2014). Implementación de Herramientas de “Software” para mejorar la Aplicación de Pruebas Unitarias en la Etapa de Construcción del Proceso de Desarrollo y Mantenimiento de “Software” de la Norma NMX-I-059-NYCE-2011. ECORFAN. <https://www.ecorfan.org/handbooks/Ciencias%20de%20la%20Ingenieria%20y%20Tecnologia%20T-VI/ARTICULO%2010.pdf>

Fatto Consultoría y Sistemas. (2020). Webinar: Automatización de pruebas de “software”. [Archivo de video] Youtube. <https://www.youtube.com/watch?v=xgJcWRQhDmY>

Fernández, S. (2018). Herramientas de automatización para pruebas de “software”. Universidad Técnica Federico Santa María. <https://repositorio.usm.cl/bitstream/handle/11673/47825/3560900259665UTFSM.pdf?sequence=1&isAllowed=y>

JoanMedia. (2020). Cómo INSTALAR CYPRESS | Cómo USAR Cypress | TUTORIAL de Cypress | Cypress.io [Video] Youtube. <https://www.youtube.com/watch?v=VimuR2V5Gf0>

Manuel Zapata. (2020). Pruebas unitarias y Test-Driven Development | Ejemplo desde cero. [Archivo de video] Youtube. <https://www.youtube.com/watch?v=YuRdaR6wwWU>

Mascheroni, M., Cogliolo, M. e Irrazabal, E. (2016). Automatización de pruebas de compatibilidad web en un entorno de desarrollo continuo de “software”. 17º Simposio Argentino de Ingeniería de “Software”. <http://sedici.unlp.edu.ar/bitstream/handle/10915/57081/Documento_completo.pdf-PDFA.pdf?sequence=1&isAllowed=y>

Pull Request. (2023). Cómo DESCARGAR e INSTALAR Sonarqube y ANALIZAR proyecto. [Archivo de video] Youtube. <https://www.youtube.com/watch?v=5UoygWLrBqo>

Serna, E. y Serna, A. (2012). Una evaluación a las herramientas libres para pruebas de “software”. Revista Virtual Universidad Católica Del Norte, (37), p. 44–61. <https://revistavirtual.ucn.edu.co/index.php/RevistaUCN/article/view/387>

Testing Bolivia (2019). Webinar - Conociendo Serenity BDD – “Framework” de “Automation”. [Archivo de video] Youtube. <https://www.youtube.com/watch?v=609y4K4ph6s>

TestProject. (2018). How to Create Web Tests with TestProject [Archivo de video] Youtube. <https://www.youtube.com/watch?app=desktop&v=hbMbcjm492c>

Créditos

| Nombre | Cargo | Centro de Formación y Regional |
| --- | --- | --- |
| Milady Tatiana Villamil Castellanos | Responsable del Ecosistema | Dirección General |
| Olga Constanza Bermúdez Jaimes | Responsable de Línea de Producción | Centro de Servicios de Salud - Regional Antioquia |
| Ervin Andrade | Experto Temático | Centro de Teleinformática y Producción Industrial - Regional Cauca |
| Peter Pinchao | Experto Temático | Centro de Teleinformática y Producción Industrial - Regional Cauca |
| Paola Alexandra Moya | Evaluadora Instruccional | Centro de Servicios de Salud - Regional Antioquia |
| Juan Daniel Polanco Muñoz | Diseñador de Contenidos Digitales | Centro de Servicios de Salud - Regional Antioquia |
| Edwin Sneider Velandia Suárez | Desarrollador Fullstack | Centro de Servicios de Salud - Regional Antioquia |
| Edgar Mauricio Cortés García | Actividad Didáctica | Centro de Servicios de Salud - Regional Antioquia |
| Daniela Muñoz Bedoya | Animador y Productor Multimedia | Centro de Servicios de Salud - Regional Antioquia |
| Andrés Felipe Guevara Ariza | Locución | Centro de Servicios de Salud - Regional Antioquia |
| Luis Gabriel Urueta Álvarez | Validador de Recursos Educativos Digitales | Centro de Servicios de Salud - Regional Antioquia |
| Jaime Hernán Tejada Llano | Validador de Recursos Educativos Digitales | Centro de Servicios de Salud - Regional Antioquia |
| Margarita Marcela Medrano Gómez | Evaluador para Contenidos Inclusivos y Accesibles | Centro de Servicios de Salud - Regional Antioquia |
| Daniel Ricardo Mutis Gómez | Evaluador para contenidos inclusivos y accesibles | Centro de Servicios de Salud - Regional Antioquia |