**FORMATO PARA EL DESARROLLO DE COMPONENTE FORMATIVO**

|  |  |
| --- | --- |
| PROGRAMA DE FORMACIÓN | Tecnólogo en aseguramiento de la calidad de *software* |

|  |  |  |  |
| --- | --- | --- | --- |
| COMPETENCIA | 220501099 - Probar la solución de acuerdo con parámetros técnicos y modelos  de referencia | RESULTADOS DE APRENDIZAJE | 220501099-02 Diseñar pruebas de *software* de acuerdo a parámetros técnicos y modelos de referencia. |

|  |  |
| --- | --- |
| NÚMERO DEL COMPONENTE FORMATIVO | CF012 |
| NOMBRE DEL COMPONENTE FORMATIVO | 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. |
| PALABRAS CLAVE | Automatización, errores, *bug*, métricas |

|  |  |
| --- | --- |
| ÁREA OCUPACIONAL | 2 - Ciencias naturales, aplicadas y relacionadas |
| IDIOMA | Español |

1. **Tabla de contenidos**

**Introducción**

**1. Herramientas de pruebas de *software* por código**

**2. Herramientas de pruebas de *software* de GUI**

**3. Desarrollo guiado por pruebas**

**4. Calidad del código fuente**

4.1 Métricas de indicadores

4.2 Caso de prueba herramienta SonarQube

1. **Desarrollo de contenidos**

**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.

Cabe resaltar que, cuando se desarrollan aplicaciones web, uno de los desafíos es verificar que la calidad del producto sea óptima y sin errores, y una actividad que permite lo anterior es automatizar los test, que empiezan con una primera etapa de pruebas manuales, para posteriormente ser automatizadas. Conocer las diversas herramientas de pruebas de *software* permite probar las funcionalidades definidas en el plan de pruebas, verificando por ejemplo que un método de una clase específica funcione correctamente, así como asegurar que una serie de acciones en una interfaz de usuario se efectúen de forma correcta y que, además, retornen los resultados esperados. Lo invitamos a explorar el siguiente video y conocer más:

|  |
| --- |
| **Video**  **CF02\_Introducción** |

Para la elaboración de este componente, se abordaron varios autores conocidos en **automatización de pruebas de *software***, de quienes se han citado y referenciado conceptos y ejemplos para los fines educativos de esta materia, en el entendido de que el conocimiento es social y, por lo tanto, es para ser usado por quienes necesitan adquirirlo. Se espera que este documento sea útil para todos aquellos, aprendices y lectores en general, que estén interesados en acercarse a asuntos básicos del **aseguramiento de la calidad de *software***.

**1. 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.

**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.

|  |
| --- |
| **Video Tutorial**  **CF02\_1. Instalación de la herramienta Cypress** |

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 documento anexo. |

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:

**Tabla 1**

Requerimientos Cypress

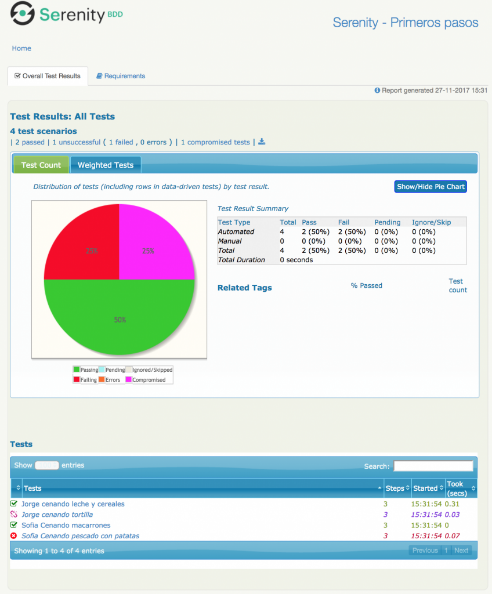
|  |  |
| --- | --- |
| **Requisitos** | **Descripción** |
| *Hardware* | Portátil o computador de escritorio (Procesador 2.4GHz, 4GB RAM, 320 GB HD). |
| Sistema operativo | Windows 10, 32 o 64 bits. |
| Navegadores | Internet Explorer 7.0.6001.1800, Mozilla Firefox 3.0.6, Opera 9.63, Safari 3.2.1, Google Chrome. |
| Editor de código | Visual Studio Code.  <https://code.visualstudio.com/download> |

***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.

**Figura 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:

**Tabla 2**

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. |

|  |  |
| --- | --- |
| Clapper board with solid fill | **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.

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:

**Tabla 3**

RequerimientosRobot 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> |

|  |  |
| --- | --- |
| Clapper board with solid fill | **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:

**Tabla 4**

Requerimientos Test Project

|  |  |
| --- | --- |
| **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. |

Esta herramienta es una plataforma de automatización de pruebas que está basada en la nube, para acceder a sus funcionalidades, se debe registrar en la plataforma online

|  |  |
| --- | --- |
| Clapper board with solid fill | **Tests con Test Project**  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:

**Tabla 5**

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/> |

|  |  |
| --- | --- |
| Clapper board with solid fill | ***Webdriver***  Para conocer algunas funcionalidades de ***Webdriver****,* se recomienda explorar los siguientes videos tutoriales.  <https://www.youtube.com/watch?v=Rv28n_rT1BM>  <https://www.youtube.com/watch?v=fnTvy_ZEcgM&t=337s> |

**2. 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.

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).

|  |  |
| --- | --- |
| Clapper board with solid fill | ***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:

|  |
| --- |
| Infografía  CF012\_2\_Herramientas de pruebas de software de GUI (1) |

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

|  |
| --- |
| Tarjetas  CF012\_2\_Herramientas de pruebas de software de GUI (2) |

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:

|  |
| --- |
| Slides  CF012\_2\_Herramientas de pruebas de software de GUI (3) |

**3. 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:

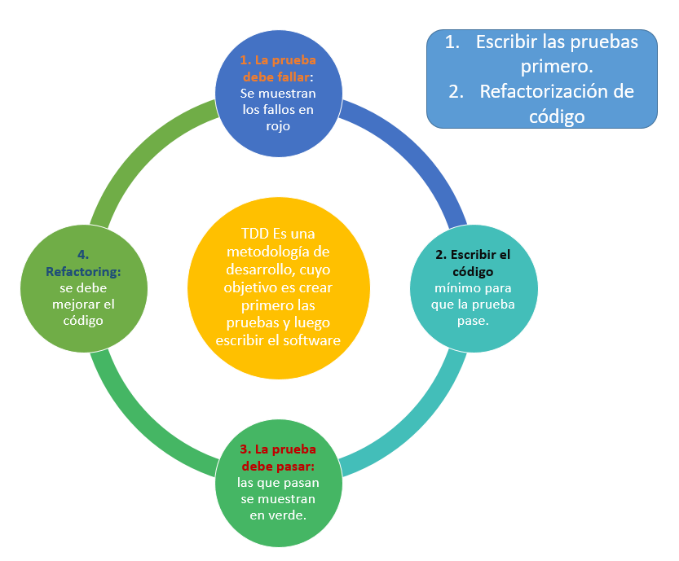
|  |
| --- |
| Video  CF012\_3\_Desarrollo guiado por pruebas (1) |

**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:

**Figura 25**

*Ciclo TDD*



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

|  |
| --- |
| Infografía interactiva  CF012\_3\_Desarrollo guiado por pruebas (2) |

|  |  |
| --- | --- |
| Clapper board with solid fill | **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> |

**4. 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.).

**4.1 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:

|  |
| --- |
| Slides  CF012\_4.1­\_Métricas de indicadores |

**4.2 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:

**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:

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

|  |
| --- |
| Acordeón  CF012\_4.2 Caso de prueba herramienta SonarQube (1) |

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

|  |
| --- |
| Acordeón  CF012\_4.2 Caso de prueba herramienta SonarQube (2) |

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**

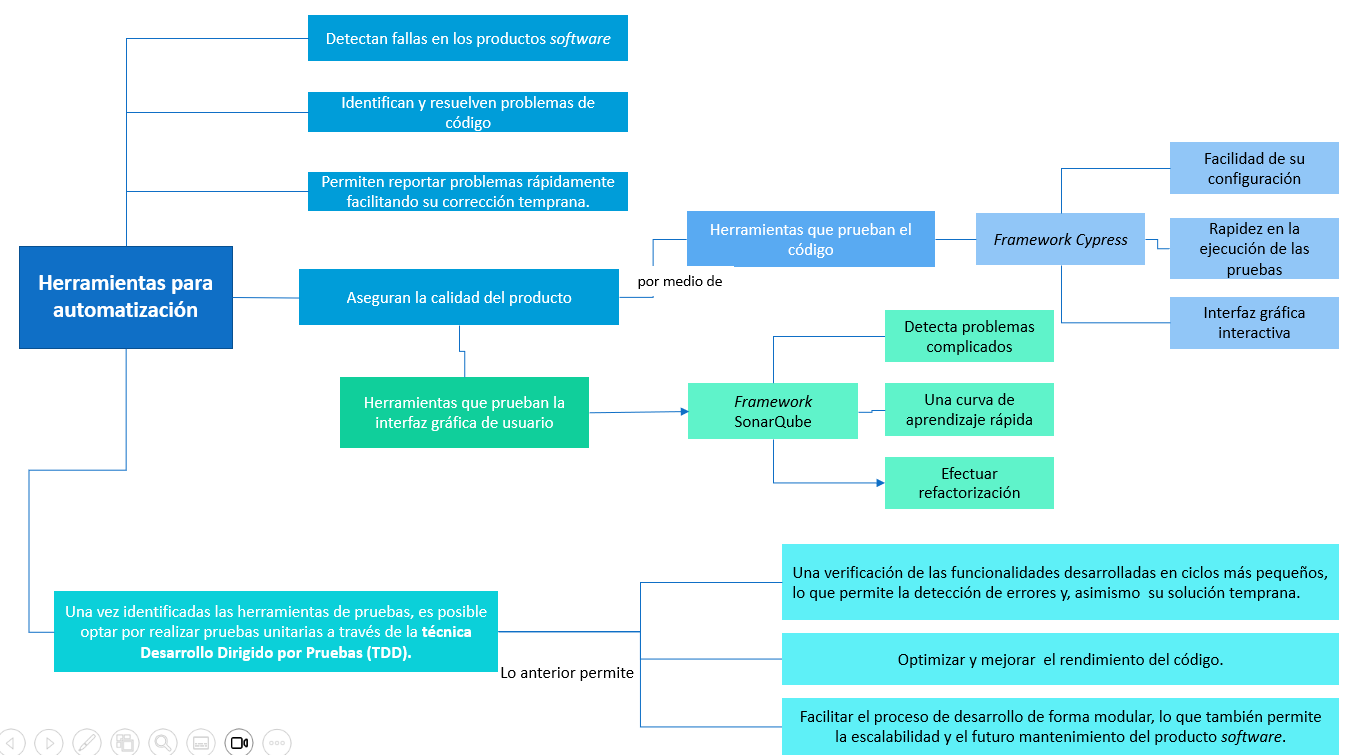
En la construcción de un *software*, hay muchos desarrolladores involucrados, cada uno de ellos con una comprensión diferente de lo que es la calidad y los estándares del desarrollo de *software*. Adicionalmente, pueden utilizar editores de código diferentes, como visual Studio Code o IntelliJ. Posteriormente, 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.

|  |  |
| --- | --- |
| Clapper board with solid fill | **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> |

1. **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.



1. **Actividades didácticas (opcionales si son sugeridas)**

|  |  |
| --- | --- |
| DESCRIPCIÓN DE ACTIVIDAD DIDÁCTICA | |
| Nombre de la Actividad | Cuestionario acerca de automatización de pruebas de *software* |
| Objetivo de la actividad | Afianzar los conceptos más importantes asociados a la automatización de pruebas de *software*. |
| Tipo de actividad sugerida | Preguntas de selección múltiple con única respuesta. |
| Archivo de la actividad  (Anexo donde se describe la actividad propuesta) | Carpeta Anexos/Anexo1\_Actividad\_1\_CF12.docx |

1. **Material complementario**

|  |  |  |  |
| --- | --- | --- | --- |
| Tema | Referencia APA del Material | Tipo de material  (Video, capítulo de libro, artículo, otro) | Enlace del Recurso o  Archivo del documento o material |
| 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. <http://sedici.unlp.edu.ar/bitstream/handle/10915/45740/Documento_completo.pdf?sequence=1&isAllowed=y> | 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. <http://sedici.unlp.edu.ar/bitstream/handle/10915/57081/Documento_completo.pdf-PDFA.pdf?sequence=1&isAllowed=y> | 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. <https://revistavirtual.ucn.edu.co/index.php/RevistaUCN/article/view/387> | 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. <https://repositorio.usm.cl/bitstream/handle/11673/47825/3560900259665UTFSM.pdf?sequence=1&isAllowed=y> | Tesis | <https://repositorio.usm.cl/bitstream/handle/11673/47825/3560900259665UTFSM.pdf?sequence=1&isAllowed=y> |

1. **Glosario**

|  |  |
| --- | --- |
| TÉRMINO | SIGNIFICADO |
| 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. |

1. **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>

JoanMedia. (2020). Cómo INSTALAR CYPRESS | Cómo USAR Cypress | TUTORIAL de Cypress | Cypress.io [Archivo de 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>

1. **Control del documento**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Nombre | Cargo | Dependencia | Fecha |
| Autor(es) | Ervin Andrade | Experto temático | SENA Regional Cauca- Centro Teleinformática y Producción Industrial | Noviembre 2021 |
| Peter Pinchao | Experto temático | SENA Regional Cauca- Centro Teleinformática y Producción Industrial | Noviembre 2021 |
| Gloria Lida Álzate Suárez | Diseñadora instruccional | Regional Distrito Capital - Centro para la Industria de la Comunicación Gráfica | Noviembre 2021 |
| Carolina Coca Salazar | Revisora Metodológica y pedagógica | Regional Distrito Capital- Centro de Diseño y Metrología | Diciembre 2021 |
| Rafael Neftalí Lizcano Reyes | Asesor pedagógico | Regional Santander - Centro Industrial del Diseño y la Manufactura | Noviembre 2021 |
| Darío González | Corrección de estilo | Regional Tolima – Centro Agropecuario La Granja | Diciembre 2021 |

1. **Control de cambios**

**(Diligenciar únicamente si realiza ajustes a la Unidad Temática)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Nombre | Cargo | Dependencia | Fecha | Razón del cambio |
| Autor(es) | Paola Alexandra Moya | Evaluadora instruccional | Centro de Servicios de Salud | Febrero 2024 | Actualización |
|  | Olga Constanza Bermúdez Jaimes | Responsable Línea de Producción Antioquia | Centro de Servicios de Salud | Febrero 2024 | Actualización |
|  |  |  |  |  |  |