ESCUELA DE POSGRADO NEWMAN

MAESTRÍA EN

GESTIÓN DE TECNOLOGÍAS DE INFORMACIÓN

Automatización de Pruebas Funcionales y Pruebas de Software en el Proceso de Control de Calidad del Ministerio de Educación del Perú, 2022

Trabajo de Investigación

para optar el Grado a Nombre de la Nación de:

Maestro en

Gestión de Tecnologías de Información

Autor:

Bach. Quispe Gutierrez, Jhon Keny

Docente Guía:

Mg. Valderrama Herrera, Roberto Marcel

TACNA – PERÚ

2023

Dedicatoria

A mi hijo Samir, a quién amo con todo mi corazón y que cuyo amor incondicional me ha fortalecido siempre y más en este proceso de aprendizaje. A los momentos felices que disfrutamos juntos y los instantes que nos extrañamos; y que gracias a sus ocurrencias me hacen ver la vida de forma distinta.

Agradecimientos

A mi madre, que siempre está a mi lado acompañándome y apoyándome. A a mi director de tesis, a mis compañeros de estudio y a los profesores por compartir sus conocimientos y experiencias y aportar al fortalecimiento académico.

Índice General

Índice de Tablas

Índice de Figuras

Índice de Anexos

Resumen

El presente estudio fijó como objetivo general precisar en qué medida la automatización de pruebas funcionales mejorará las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú. La investigación tiene dos variables, la independiente, automatización de pruebas y la dependiente, pruebas de software. En los resultados se comprobó que los de scripts automatizados de pruebas mejora significativamente la ejecución pruebas de software incrementando la eficiencia, productividad, confianza, optimizando el diseño y reduciendo los costos en el proceso de control de calidad. En la metodología se utilizó un tipo de investigación aplicada, diseño preexperimental y con un enfoque cuantitativo, debido a que se han realizado mediciones antes(pretest) con pruebas manuales y después (Postest) con pruebas automatizadas para demostrar las hipótesis que se establecieron como parte del estudio. El tamaño de la muestra que se utilizó para la investigación fueron los colaboradores que pertenecen a la Unidad de Calidad y Seguridad de la información. Los resultados obtenidos fueron los siguientes: Se logró mejorar la productividad debido a la reducción del tiempo de ejecución de casos de prueba en un 71.5% , incrementar en 33% la detección temprana de defectos, aumentar al 96% la tasa de cobertura de pruebas de regresión gracias a la repetibilidad de los scripts automatizados, mejorar en 43% el rendimiento en el diseño de casos de prueba automatizados y mejorar al 97% la eficiencia en la detección de defectos gracias a la programabilidad de los scripts automatizados.

Palabras clave: *Automatización de pruebas funcionales, Pruebas de software, Calidad de software, Control de calidad, Cobertura de pruebas, Detección temprana de defectos.*

Abstract

The general objective of this study was to determine to what extent functional test automation will improve software testing in the quality control process of the Peruvian Ministry of Education. The research has two variables, the independent one, test automation and the dependent one, software testing. The results showed that automated test scripts significantly improve the execution of software tests, increasing efficiency, productivity, confidence, optimizing the design and reducing costs in the quality control process. The methodology used was an applied research type, pre-experimental design and with a quantitative approach, due to the fact that measurements were taken before (pretest) with manual tests and after (posttest) with automated tests to demonstrate the hypotheses that were established as part of the study. The sample size used for the research was the collaborators belonging to the Information Quality and Security Unit. The results obtained were as follows: Improved productivity was achieved due to a 71.5% reduction in test case execution time, a 33% increase in early defect detection, a 96% increase in regression test coverage rate due to the repeatability of the automated scripts, a 43% improvement in automated test case design performance, and a 97% improvement in defect detection efficiency due to the programmability of the automated scripts.

***Keywords**: Functional test automation, Software testing, Software quality, Quality control, Test coverage, Early defect detection.*

Introducción

El Ministerio de Educación del Perú - MINEDU como institución rectora del sector educativo, es la responsable de diseñar, implementar y ejecutar las políticas educativas con el propósito de garantizar un sistema educativo eficaz, donde los procesos de aprendizaje sean de calidad y garanticen la inmersión al sistema de toda la comunidad educativa. Para ello, el MINEDU a través de la Oficina de Tecnologías de la Información y comunicación – OTIC, desarrolla, implementa y mantiene plataformas digitales y sistemas de información que están destinadas a favorecer el aprendizaje de la comunidad en cualquiera de sus modalidades de estudio (Presencial, Semipresencial y Virtual). La modalidad virtual ha significado todo un reto y ha conseguido gran aceptación después de la pandemia Covid19, debido a que mejora la inclusión educativa y fortalece el acceso a la educación como un derecho universal.

Por ello, es muy importante que las herramientas, plataformas y sistemas de información que el MINEDU ofrece a la comunidad siempre estén disponibles, accesibles, sean seguros y de calidad. La calidad de software puede ser medida desde perspectivas funcionales (precisión, exactitud, comportamiento y otros) y complementarias (mantenibilidad, amigabilidad, usabilidad, portabilidad y seguridad). Para garantizar que el producto educativo cumpla con cada una de estas características se hace uso de un proceso llamado “Pruebas de Software”.

Las pruebas de software garantizan el incremento de la confianza en el producto digital a través de la evaluación y verificación del cumplimiento de todas sus especificaciones. También en un entorno productivo, reducen el riesgo de la aparición de fallas en el software que finalmente terminen afectando en el proceso de aprendizaje de la comunidad. En la actualidad, gracias a la evolución de las

herramientas de licencia libre, todo el proceso de pruebas puede ser realizado de manera automatizada y es ahí donde surge la automatización de pruebas.

La automatización de pruebas es la utilización de herramientas de software para automatizar el proceso manual de revisión y validación de un producto de software.

Este trabajo se divide en cinco capítulos. Se especifica de manera detallada los temas que se abordaran en cada uno de ellos como parte del desarrollo del proceso de investigación:

Capitulo I. Antecedentes del estudio, en este capítulo se desarrolla el planteamiento y formulación del problema, las hipótesis del estudio, los objetivos, la metodología, la justificación y el cronograma de actividades.

Capitulo II. Marco Teórico, en este capítulo se desarrolla los estudios precedentes de la investigación, fundamentos teóricos de las variables o tópicos; y comparación de las bases teóricas.

Capitulo III. Marco de referencia, en este capítulo se desarrolla la reseña histórica, la filosofía, el diseño organizacional, los productos, servicios y el diagnóstico sectorial y los principales actores.

Capitulo IV. Los resultados, en este capítulo se especifican los resultados obtenidos de la investigación relacionados con los objetivos.

Y finalmente el Capitulo V Sugerencias y Recomendaciones. Se plantean sugerencias y recomendaciones en base a los objetivos, los resultados y las conclusiones obtenidas en la investigación.

CAPITULO I: ANTECEDENTES DE ESTUDIO

1.1. Título del Tema

Automatización de Pruebas Funcionales y Pruebas de Software en el Proceso de Control de Calidad del Ministerio de Educación del Perú.

1.2. Planteamiento del Problema

En todo el mundo, en el periodo del año 1990, ocurrió una transformación de las pruebas a un procedimiento muy integrador denominado garantía de calidad, y que abarcaba todo el ciclo de vida de desarrollo de software y afectaba los procesos de planificación, análisis, diseño, construcción y ejecución de casos de prueba. Además, brindaba soporte y ambientes para casos de prueba. (IBM, 2022)

PRIUS, Toyota en el 2017 retiró más de 400 mil vehículos por un problema de software en los frenos y que le costó a la empresa 3 billones de dólares. Hoy en día las empresas lideres de desarrollo de software como IBM, SAP y Oracle dedican un gran espacio para las pruebas, debido a que un producto o servicio que cumple o incluso supere las expectativas del cliente, genera definitivamente mayores ingresos e incrementa la cuota de mercado. Un software con defectos genera pérdidas económicas, daña la reputación de una marca, lo que en un futuro podría provocar la insatisfacción y pérdida de clientes. (Siroky, 2017)

En América Latina, la estrategia de testing en los proyectos es progresivamente más dinámico, estableciendo desafíos en el liderazgo de enfoque que defiendan el dinamismo y la agilidad en el progreso y rotación de equipos. Los países de Latinoamérica tienen sus propias necesidades y emprenden acciones para hacer frente a avances tecnológicos, con dirección de especialización en ciberseguridad, rendimiento, experiencia de usuario, calidad y automatización de pruebas con la adopción de herramientas low code y de licencia libre que certifiquen la calidad del

software y minimicen el costo de los defectos. En la actualidad dirigir empresas y equipos requerirá de personas auténticas, flexibles, humildes y listas para afrontar la situación actual. (Toledo, 2022).

En Perú, debido al confinamiento por la pandemia del Covid-19, la educación virtual dejo de ser una alternativa para convertirse en la forma más viable para enseñar y aprender. El MINEDU, como institución rector y facilitador del aprendizaje tuvo una ardua tarea para reducir las brechas y brindar a la comunidad educativa las herramientas tecnológicas (Hardware y Software) de calidad, confiables y que siempre estén disponibles para garantizar un proceso de aprendizaje continuo. Tecnologías basadas en nube y la adopción de entorno de trabajo agiles contribuyen a todo ello. En el ciclo de vida ágil de desarrollo de software, donde se reduce el time to Marquet que garantice un producto mínimo viable, las pruebas de software cobran gran relevancia en todo el proceso. Estas garantizan la calidad y confiabilidad del producto de software educativo.

El Ministerio de educación del Perú, mediante la Oficina de Tecnologías de Información y Comunicación – OTIC y de la Unidad de Calidad y Seguridad de la Información; desarrolla y mantiene productos de software a medida para toda la comunidad educativa soportados en un marco tradicional y ágil. Las apps y las plataformas digitales tales como: Aprendo en casa APP, Materiales educativos APP, Transmisiones, campus virtual estudiantes, campus virtual docentes, portal PerúEduca 4.0 y otros tomaron gran importancia y llegaron para quedarse.

En estos dos últimos años los requerimientos y proyectos de construcción de software se han incrementado en el sector educativo, debido a la aparición de tecnologías emergentes, iniciativas de transformación, migración a la nube, la demanda educativa, y la importancia de la educación virtual. Toda esta alta demanda

ha significado que, en el proceso de construcción de software, las actividades de pruebas de calidad, ha sufrido en rendimiento y eficiencia

Las causas que originan que el proceso de pruebas de calidad de software se vea disminuido en su rendimiento son los siguientes:

1) Incremento de requerimientos de mantenimiento y mejoras de productos de software
2) Incremento de proyectos de desarrollo de software e iniciativas de transformación.
3) Aumento de las pruebas de regresión y ejecuciones repetitivas de las pruebas de caja negra por cada sprint e iteración.
4) Tiempos altos en la ejecución de las pruebas funcionales, porque estas se ejecutan de manera manual.
5) Las nuevas políticas y la demanda de la comunidad educativa exigen la puesta en producción del software en menor tiempo.

Las consecuencias que se generan debido al problema que se presenta en el las actividades de control de calidad por disminución en la eficiencia de las pruebas de software son las siguientes:

1) Disminución en la calidad de software
2) Incremento de los costos de reparación de defectos
3) Aumento del índice de fallas en producción
4) Reducción de la tasa de cobertura de pruebas de regresión
5) Baja productividad en las pruebas de calidad de software
6) Aumento del Time to Market del Producto
7) Problemas de disponibilidad de los productos de software para los usuarios finales.

El testing manual puede ser idóneo para productos pequeños. Sin embargo, para aplicaciones más complejas, con mucha demanda y con intervalos de entrega reducidos, toman mucho tiempo. Por ello, se plantea como aporte la automatización de pruebas funcionales. La automatización permite al evaluador probar más escenarios en menos tiempo. En la actualidad existen herramientas que son aplicados con regularidad para automatizar procesos, actividades y acciones. Las pruebas automatizadas dan soporte a los grupos de trabajo para probar escenarios en diferentes plataformas, y así obtener de manera anticipada información rápida acerca de las fallas del sistema puesto bajo prueba. Actualmente, para construir entornos para automatizar las pruebas de calidad del software, existen marcos y herramientas con licencia abierta, así como soluciones de proveedores que brindan características que consigan mejorar las actividades clave de la administración de pruebas.

1.3. Formulación del Problema

Las consideraciones expuestas en el acápite anterior nos llevan a plantear las siguientes interrogantes:

1.3.1. Problema General

¿En qué medida la automatización de las pruebas funcionales mejora las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022?

1.3.2. Problemas Específicos

¿En qué medida la rapidez de los scripts automatizados mejora la productividad de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022?

¿En qué medida la fiabilidad de los scripts automatizados mejora la detección temprana de defectos de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022?

¿En qué medida la repetición de los scripts automatizados mejora la cobertura de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022?

¿En qué medida la reusabilidad de los scripts automatizados mejora el rendimiento del diseño de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022?

¿En qué medida la programación de los scripts automatizados mejora la eficiencia de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022?

1.4. Hipótesis de la Investigación

1.4.1. Hipótesis General

La automatización de las pruebas funcionales mejora significativamente las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022

1.4.2. Hipótesis Específicos

La rapidez de los scripts automatizados mejora la productividad de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022.

La fiabilidad de los scripts automatizados mejora la detección temprana de defectos de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022.

La repetición de los scripts automatizados mejora la cobertura de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022.

La reusabilidad de los scripts automatizados mejora el rendimiento del diseño de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022.

La programación de los scripts automatizados mejora la eficiencia de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022.

1.5. Objetivos de la Investigación

1.5.1. Objetivo General

Determinar como la automatización de las pruebas funcionales mejora las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022.

1.5.2. Objetivos Específicos

Determinar como la rapidez de los scripts automatizados mejora la productividad de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022.

Determinar como la fiabilidad de los scripts automatizados mejora la detección temprana de defectos de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022.

Determinar como la repetición de los scripts automatizados mejora la cobertura de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022.

Determinar como la reusabilidad de los scripts automatizados mejora el rendimiento del diseño de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022.

Determinar como la programación de los scripts automatizados mejora la eficiencia de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022.

1.6. Metodología

1.6.1. Tipo y Diseño de la Investigación

1.6.1.1. Tipo de Investigación

En la presente investigación, se utiliza una investigación aplicada a medida que se aprovechan los conocimientos adquiridos de la automatización de pruebas y se utilizan para transformar y resolver el problema de optimizar las pruebas de software en el proceso de control de calidad del MINEDU.

Según Gerena (2018), indica que “la investigación aplicada consiste en obtener conocimientos y ponerlos en práctica, así como realizar estudios científicos para encontrar soluciones o respuestas a posibles aspectos de mejora en situaciones cotidianas” (p. 6).

1.6.1.2. Diseño de la Investigación

El diseño para la investigación es preexperimental, porque se quiere probar como la automatización de pruebas funcionales(causa), optimiza las pruebas de software(efecto) en el proceso de control de calidad del MINEDU.

Según Hérnandez, Fernandéz, & Baptista (2014) mencionan que; los diseños preexperimentales permiten estudiar casos con una medida en un diseño pretest y postest aplicado a un grupo. Es decir, medir la variable dependiente antes del

tratamiento y después de agregar la variable independiente se realiza un test postestímulo (pp. 140-141).

1.6.1.3. Enfoque de la Investigación

Según Gómez (2006) explica que, “El enfoque cuantitativo es la recolección de datos equivalentes a medir. Donde medir significa asignar números a objetos y eventos de acuerdo a ciertas reglas. Muchas veces el concepto se hace observable a través de referentes empíricos asociados a él. Por tal motivo parece más apropiado definir la medición en ciencias como el proceso de articular conceptos abstractos con indicadores empíricos, mediante un plan explicito y constituido para poder clasificar los datos disponibles, en función del concepto que el investigador tiene en mente”.

Por lo tanto, se opta por un enfoque cuantitativo para la presente investigación, ya que se cuenta con datos medibles que tienen un diseño preexperimental (causa y efecto). Es decir, se medirán el rendimiento de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú antes de la aplicación de la variable independiente “automatización de pruebas funcionales” y otras pruebas posteriores a dicha aplicación.

1.6.2. Población y muestra

1.6.2.1. Población

Según Hérnandez et al. (2014) afirmaron que “la población es el conjunto de todos los casos que concuerdan con determinadas especificaciones” (p. 174).

Por lo tanto, la población que se empleará para el estudio está conformada por los casos de prueba del portal Perú educa y los 28 colaboradores de la Unidad de Calidad y Seguridad de la Información – UCSI, adscrita a la Oficina de Tecnología de la Información y Comunicaciones - OTIC. En el total de la población se considera a

los coordinadores, supervisores, especialistas y analistas de las pruebas funcionales y de seguridad.

En la tabla 1, se describe el detalle de la población:

Tabla 1

Perfiles/colaboradores del Área de Certificación de Software

Perfiles	Cantidad
Coordinador de Calidad	2
Especialistas de Calidad	5
Analistas de Control de Calidad/Seg.	21
Total	**28**

1.6.2.2. Muestra.

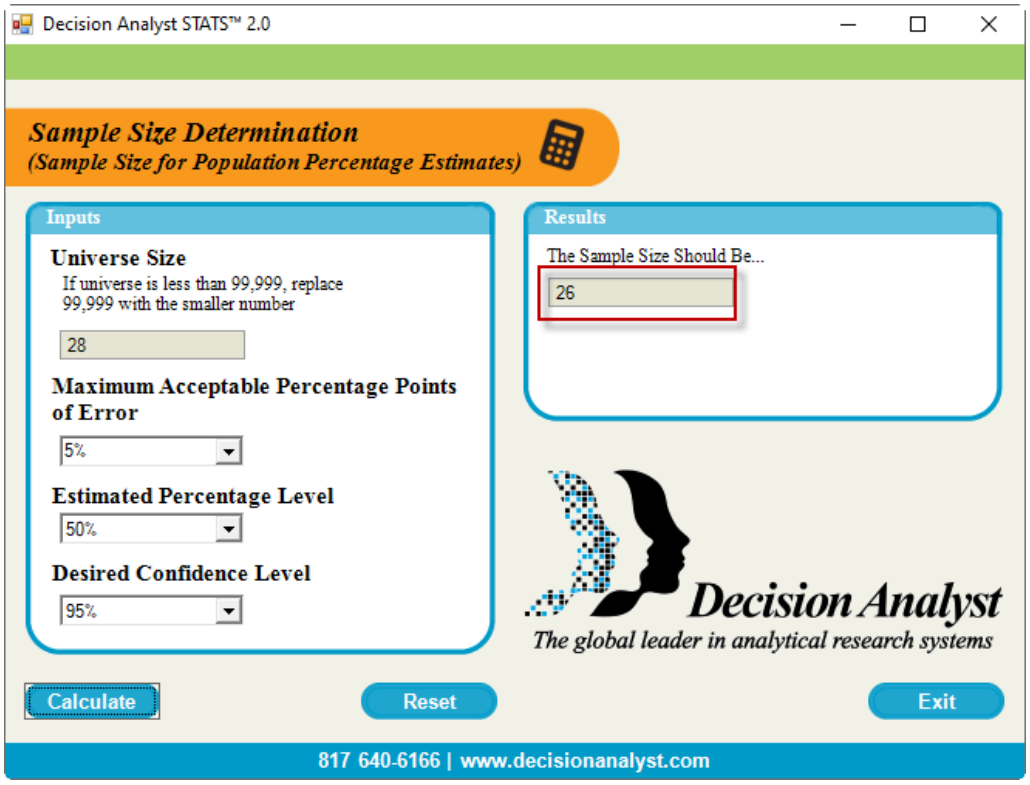
La muestra es un subgrupo particular de la población y se toman con la finalidad de estudiar particularidades y determinar características del grupo en general. (Hérnandez, Fernandéz, & Baptista, 2014)

Por consiguiente, la muestra que se empleará para el estudio está conformada los 26 analistas de pruebas funcionales de la UCSI, adscrita a la OTIC del MINEDU. Para determinar el tamaño de la muestra se utilizó el aplicativo de estadística denominado “Statistical Calculators™ 2.0 Desktop”.

En la figura 1, se resume el resultado obtenido después de ingresar los valores correspondientes para el cálculo como son: Universo, porcentaje máximo de precisión, nivel estimado de porcentaje y nivel de confianza deseado.

Figura 1

Proceso de Determinación del Tamaño de la Muestra



Nota. El segmento marcado de rojo representa el tamaño de la muestra calculada.

1.6.3. Operacionalización de Variables

Para el presente estudio se plantea como variable independiente "La "Automatización de Pruebas Funcionales" y como variable dependiente "Pruebas de Software". Seguidamente se explica al detalle el concepto y la operacionalización de las 2 variables que presenta la investigación a través de matrices donde se especifican las dimensiones, indicadores e instrumentos de cada una de ellas.

1.6.3.1. Variable Independiente: Automatización de Pruebas Funcionales.

La automatización de pruebas de software se trata de crear un grupo de guiones reutilizables, que podemos usar para mejorar contundentemente la capacidad de probar el software en términos de verificación y pruebas de regresión antes y después del despliegue de una nueva versión. Éstos agregan una mayor cobertura de

prueba, minimizan los tiempos de prueba y permiten el reúso de los mismos. La automatización complementa las pruebas manuales y no las reemplaza. (Crespo, 2018).

La automatización de pruebas ofrece la facultad de ejecutar pruebas en paralelo, sin supervisión y en diferentes plataformas web. Esto disminuye los costos e incrementa la validación de la funcionalidad del producto en distintos navegadores. (Toledo, Curcio, & Scuoteguazza, 2014).

1.6.3.2. Variable Dependiente: Prueba de Software.

Según el International Software Testing Qualifications Board (ISTQB, 2018) conceptualizan a “las pruebas de software son un proceso que contiene muchas actividades diferentes, que van desde la planificación, el análisis, el diseño, la implementación, ejecución, informe de avances, cierre y seguimiento y control de las pruebas. El objetivo de cada etapa es hallar defectos y garantizar el cumplimiento de los requisitos iniciales especificados en las bases de prueba” (p. 23).

Por tanto, se puede aseverar a las pruebas de software como un proceso donde se valida que el producto cumpla con los requisitos especificados durante su concepción y construcción, con la finalidad de detectar defectos antes de desplegarse en producción.

En la tablas 2 y 3, se especifican a detalle la operacionalización de las variables de estudio.

Tabla 2

Operacionalización de la Variable independiente V_1: Automatización de Pruebas Funcionales

VARIABLE INDEPENDIENTE	DIMENSIONES	INDICADORES	FORMULAR	INSTRUMENTO
V_1 - Automatización de Pruebas Funcionales	1. Rapidez	Medir el tiempo del proceso en las ejecuciones		Ficha Técnica de recolección de datos cuantitativos
	2. Fiabilidad	Errores en la ejecución de casos de Prueba.		Ficha Técnica de recolección de datos cuantitativos
	3. Repetibilidad	Ejecución de casos de pruebas en distintas plataformas.		Ficha Técnica de recolección de datos cuantitativos
		Ejecución de casos de pruebas con diferentes datos y valores de entrada.		
	4. Reusabilidad	Reusar scripts de prueba para diseñar otros casos de prueba		Ficha Técnica de recolección de datos cuantitativos
	5. Programable	Eficiencia en la detección de defectos		Ficha Técnica de recolección de datos cuantitativos

Tabla 3

Operacionalización de la Variable dependiente V2: Pruebas de Software

VARIABLE DEPENDIENTE	DIMENSIONES	INDICADORES	FORMULAR	INSTRUMENTO
V_2 – Pruebas de Software	1. Productividad	Tiempo de ejecución de pruebas	TEPF=TEPFM-TEPFA TEPF: Tiempo de ejecución de pruebas funcionales TEPFM: tiempo de ejecución de pruebas funcionales manuales TEPFA: Tiempo de ejecución de pruebas funcionales automatizadas	Ficha Técnica de recolección de datos cuantitativos
	2. Detección temprana de defectos	Cantidad de defectos reportados	TBR= TBPA - TBPM TBR: Total de Bugs TBPM: Total Bugs con pruebas manuales TBPA: Total de Bugs con pruebas automatizadas	Ficha Técnica de recolección de datos cuantitativos
	3. Cobertura de Pruebas	Cantidad de pruebas de regresión ejecutadas.	TPR=TPRM-TPRA TPR: Total de pruebas de regresión TPRM: Total pruebas de regresión manuales TPRA: Total pruebas de regresión automatizadas	Ficha Técnica de recolección de datos cuantitativos
	4. Diseño de Pruebas	Tasa de diseño de casos de prueba automatizados	TDCPA=(TCPAR/TCPAI) *100 TDCPA=Tasa (%) de diseño de casos de prueba automatizados TCPAI= Total de casos de prueba automatizados iniciales TCPAR=	Ficha Técnica de recolección de datos cuantitativos
	5. Eficiencia	Eficiencia en la detección del defecto	EDD=(DS/DS+DP) *100 EDD: Eficiencia en la detección del defecto DS: Defectos detectados en pruebas del sistema DP: Defectos detectados en Producción	Ficha Técnica de recolección de datos cuantitativos

Tabla 4

Matriz de Consistencia

PROBLEMA GENERAL	OBJETIVO GENERAL	HIPÓTESIS GENERAL	VARIABLE(S)	DIMENSION	INDICADORES	METODOLOGÍA
¿En qué medida la automatización de las pruebas funcionales mejora las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022?	Determinar como la automatización de las pruebas funcionales mejora las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022	La automatización de las pruebas funcionales mejora significativamente las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022	**Independiente: Automatización de Pruebas funcionales.** Según (Sarco, 2019) Las características más significativas de la automatización de pruebas son: la rapidez, porque las herramientas de pruebas automatizadas se ejecutan notablemente más rápido que los probadores manuales; la Fiabilidad, debido a que las pruebas realizan exactamente las mismas operaciones disminuyendo los errores; la repetición, porque se puede testear cómo reacciona el producto de software bajo repetidas ejecuciones de las mismas operaciones; programable, porque se pueden programar	Rapidez	Medir el tiempo del proceso en las ejecuciones	**Tipo de Investigación: Aplicada** Según Gerena (2018), indica que la investigación aplicada consiste en obtener conocimientos y ponerlos en práctica, así como realizar estudios científicos para encontrar soluciones o respuestas a posibles aspectos de mejora en situaciones cotidianas. **Diseño de la Investigación: Experimental** Según Hernández, Fernández, & Baptista (2014) mencionan que; los diseños preexperimentales permiten estudiar casos con una medida en un diseño pretest y postest aplicado a un grupo. Es decir, medir la variable dependiente antes del tratamiento y después de agregar la variable
				Fiabilidad	Errores en la ejecución de casos de Prueba.	
				Repetibilidad	Ejecución de casos de pruebas en diferentes plataformas.	
					Ejecución de casos de pruebas con diferentes datos y valores de entrada.	
				Reusabilidad	Reutilizar scripts de prueba para diseñar otros casos de prueba	
				Programable	Eficiencia en la detección de defectos	

			pruebas sofisticadas y complicadas que revelen datos desconocidos del sistema y la reusabilidad, debido a que se pueden rehusar los scripts con pruebas automatizadas para diseñar otros scripts con facilidad.			independiente se realiza un test postestímulo. **Enfoque de la Investigación: Cuantitativo** **Nivel de la Investigación: Aplicativo**
PROBLEMAS ESPECÍFICOS	**OBJETIVOS ESPECÍFICOS**	**HIPÓTESIS ESPECÍFICOS**	**VARIABLE(S)**	**DIMENSION**	**INDICADORES**	**Área de estudio**: Ministerio de educación del Perú – Personal de TI **Muestra:** Analistas de Pruebas -UCSI **Instrumentos:** Ficha técnica de recolección de datos.
¿En qué medida la rapidez de los scripts automatizados mejora la productividad de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022?	Determinar como la rapidez de los scripts automatizados mejora la productividad de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022.	La rapidez de los scripts automatizados mejora la productividad de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022	**Dependiente: Pruebas de Software.** Según (ISTQB, 2018) definen “las pruebas de software son un proceso que contiene muchas actividades diferentes, que van desde la planificación, el análisis, el diseño, la implementación, ejecución, informe de avances, cierre y seguimiento y control de las pruebas”.	Productividad	Tiempo de Ejecución de Pruebas	
¿En qué medida la fiabilidad de los scripts automatizados mejora la detección temprana de defectos de software en el	Determinar como la fiabilidad de los scripts automatizados mejora la detección temprana de defectos de	La fiabilidad de los scripts automatizados mejora la detección temprana de defectos de software en el proceso de control		Detección temprana de defectos	Cantidad de defectos reportados	

proceso de control de calidad del Ministerio de Educación del Perú en el año 2022?	software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022.	de calidad del Ministerio de Educación del Perú en el año 2022.		
¿En qué medida la repetición de los scripts automatizados mejora la cobertura de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022?	Determinar como la repetición de los scripts automatizados mejora la cobertura de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022.	La repetición de los scripts automatizados mejora la cobertura de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022.	Cobertura de Pruebas	Cantidad de Pruebas de regresión ejecutadas.
¿En qué medida la reusabilidad de los scripts automatizados mejora el rendimiento del diseño de las pruebas de software en el proceso de	Determinar como la reusabilidad de los scripts automatizados mejora el rendimiento del diseño de las pruebas de software en el	La reusabilidad de los scripts automatizados mejora el rendimiento del diseño de las pruebas de software en el proceso de control de calidad del	Diseño de Pruebas	Tasa de Casos de Prueba automatizados

control de calidad del Ministerio de Educación del Perú en el año 2022?	proceso de control de calidad del Ministerio de Educación del Perú en el año 2022.	Ministerio de Educación del Perú en el año 2022.		
¿En qué medida la programación de los scripts automatizados mejora la eficiencia de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022?		La programación de los scripts automatizados mejora la eficiencia de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022.	Eficiencia	Eficiencia en la detección del defecto

1.6.4. Técnicas e Instrumentos de Recolección de Datos

1.6.4.1. Técnica de recolección de datos

Las técnicas e instrumentos para la recolección de datos e información tienen por propósito recopilar, almacenar, evaluar, analizar y compartir datos que forman la base de una investigación particular, están compuestos por un grupo de mecanismos, recursos o medios. Por consiguiente, se alcanza a concluir que las técnicas, y las herramientas de recolección de datos e información, son los procedimientos y/o recursos básicos de recolección utilizados por el investigador para comprobar los hechos y acceder a determinados resultados de estudio a partir de ese conocimiento.

La técnica utilizada para la recolección de datos en esta investigación es la observación. Según Hérnandez et al. (2014) indican que “es un registro sistemático, válido y confiable de comportamientos en situaciones observables, a través de un conjunto de categorías y subcategorías” (p. 252). Esta técnica ha sido aplicada en la ejecución de casos de prueba y en los analistas del Área de Seguridad y Calidad de la Información del Ministerio de Educación del Perú. Para mayor detalle revisar la sección de anexos del presente documento.

1.6.4.2. Instrumento de recolección de datos

Para este se ha utilizado ficha técnica de observación como instrumento de recolección de datos e información. Es de elaboración propia y cuyo objetivo es registrar de manera confiable datos de cada evento en el proceso de las pruebas de software (pre y post prueba) y medir el efecto de la automatización de pruebas funcionales en las pruebas de calidad. Es un instrumento basado en indicadores cuantitativos.

1.7. Justificación

1.7.1. Justificación Teórica

La investigación propone determinar si la automatización de pruebas funcionales como un marco de eficiencia y buenas prácticas de pruebas de calidad y genera un efecto positivo en las actividades de prueba en la unidad de calidad y seguridad de la información. Con los resultados obtenidos se pueden proponer acciones encaminadas al desarrollo de scripts automatizados para aumentar la productividad del servicio de calidad.

1.7.2. Justificación Práctica

La automatización de pruebas permite aumentar la productividad del proceso de pruebas, reduciendo el tiempo empleado en ellas, mejorando la detección temprana de los defectos, multiplicando la cobertura de las pruebas, optimizando el diseño de las pruebas con la finalidad de minimizar los impactos en producción. Todo ello gracias a las características de rapidez, reusabilidad, fiabilidad, programabilidad y repetitividad de los scripts automatizados que se efectúan de manera automática en las computadoras y con poca intervención humana.

1.7.3. Justificación Metodológica

El diseño que se va aplicar en el estudio es de tipo pre - experimental, esto en base a que se medirá el impacto de la variable independiente automatización de pruebas funcionales(causa) sobre las pruebas de software(efecto). Para la captura de información de la prueba previa y posterior se utilizará una hoja técnica de observación para datos cuantitativos. Los aportes del estudio se relacionan con la originalidad y la novedad, debido a la realización de pruebas automatizadas usando herramientas de código abierto, las buenas prácticas de la agilidad como un entorno de trabajo y una arquitectura de fácil mantenimiento. En ese contexto, los resultados de la investigación

se ponen disponibles a los interesados en la materia y pueden ser utilizados en otros proyectos de investigación.

1.8. Alcances y Limitaciones

1.8.1. Alcances

La investigación se desarrollará de manera virtual en las instalaciones Unidad de Control de Calidad y Seguridad de la Información – UCSI MINEDU de la Provincia de Lima. La realización de la investigación tiene como propósito demostrar como las características de los scripts automatizados pueden mejorar significativamente las pruebas de software en el proceso de control de calidad a través del estudio de dos variables, una es la automatización de pruebas funcionales y la otra es la prueba de software. El estudio no contempla la implementación de un entorno/marco de automatización.

1.8.2. Limitaciones

El desarrollo del estudio se extiende en un período de 8 meses, de agosto de 2022 a marzo de 2023. Además, otra limitante es la forma de trabajo, el tiempo, la disponibilidad de datos, recursos e información del proceso de control de calidad, que puede manifestarse por cambios en la sede por lo cual la investigación puede tomar mas tiempo de lo planificado.

Cabe señalar que, debido a las políticas de confidencialidad y privacidad de la institución, la divulgación de los nombres de los participantes, algunos números y cierta información considerada importante no será divulgada en esta investigación.

1.9. Principales definiciones

1.9.1. Automatización de Pruebas Funcionales

La automatización de pruebas consiste en usar herramientas de programación para crear scripts automatizados que usan computadoras para probar el correcto funcionamiento del software con mayor rapidez; contribuye al aumento de la eficiencia, productividad, alcance y detección de errores en el proceso de control de calidad. La aplicación de un marco de prueba automatizado debe regirse por estructuras de patrones de diseño y buenas prácticas de agilidad.

1.9.2. Prueba de Software

Es el proceso de evaluar y verificar que un producto, aplicación o componente de software cumpla con todos los requisitos especificados, las necesidades y expectativas del usuario final. El propósito de las pruebas de software es hallar el mayor número de defectos con la finalidad de garantizar su calidad y generar confianza, bienestar y satisfacción de su uso por comunidad educativa peruana.

CAPITULO II: MARCO TEÓRICO

2.1. Antecedentes de la Investigación

En este apartado se exponen los antecedentes nacionales e internacionales que han servido de apoyo y se relacionan con la presente investigación

2.1.1. Antecedentes Nacionales

Fernandez Avalos (2018). "*Automatización de Procesos para mejorar las Pruebas de Software en el área de calidad del Banco de Crédito"*. Resumen: El objetivo principal de esta tesis fue establecer en qué grado la automatización de procesos optimizaría las pruebas de software en el área de calidad del Banco de Crédito. El estudio hace el análisis de dos variables: la independiente, automatización de procesos y la dependiente, pruebas de software. El estudio confirmó que el uso de la automatización de procesos beneficia las pruebas de software al potenciar la eficiencia y la productividad en la generación de datos de prueba. La metodología utilizó un tipo de investigación aplicada, diseño preexperimental y enfoque cuantitativo, ya que se empleó la recopilación de datos a través de una ficha técnica para demostrar las hipótesis que se desarrollaron como parte de la investigación. El volumen de la muestra utilizado para la investigación fueron de 30 colaboradores que pertenecen al departamento de calidad del software. Los resultados arrojaron que el tiempo de creación de datos de prueba se redujo en un 56,78 %, y el volumen de datos generados se incrementó en un 7,71 %. Por lo tanto, gracias a los resultados obtenidos, se concluyó que, al automatizar los procesos se alcanzó mejorar de manera representativa las pruebas de software.

Cabrera Serna & Pareja Verastegui (2021). "*Automatización de pruebas funcionales web para mejorar el área de calidad de software en una empresa del rubro de retails en el año 2021"*. Resumen: La investigación nace como respuesta a la

necesidad de realizar pruebas automatizadas durante el ciclo de construcción de software que ayuden a acortar los tiempos de ejecución de las pruebas y abordar una amplia cobertura de escenarios. El objetivo de esta tesis fue automatizar las pruebas basadas en el comportamiento de un sistema web para perfeccionar el área de calidad de software de la empresa de retail a través de diversas herramientas de código abierto, como Selenium WebDriver y otros. La metodología utilizada fue de tipo investigación aplicada, diseño experimental y enfoque cuantitativo. Se empleó como técnica e instrumento de recopilación de datos la encuesta y el cuestionario respectivamente. El tamaño de muestra utilizado para la investigación fue de 20 personas, quienes conforman el departamento de Calidad y Desarrollo de Software. El estudio demostró que la automatización de las pruebas redujo los tiempos de ejecución de testing funcional en un 60% y del testing de Regresión en un 61% para el módulo de Reclamos e Incidencias respectivamente.

Medina Yacupoma (2020)**.** "*Automatización de pruebas para proyectos ágiles aplicando el desarrollo dirigido por comportamiento para una compañía de líneas de belleza".* Resumen: El trabajo de investigación aplicó la automatización de pruebas utilizando el marco de ágil de desarrollo guiado por el comportamiento-BDD para una empresa de cosméticos, con la finalidad de mejorar y optimizar las pruebas de regresión, en respuesta a los problemas que presentaba la empresa porque, el proceso de testing se realizaba de manera manual y con ciertas automatizaciones parciales, pero sin seguir un estándar. Como objetivo del estudio propone reducir en un 60% y 50% la cantidad y el tiempo en la realización de las pruebas manuales de regresión respectivamente. El método de investigación utilizado fue cuantitativo con un diseño transversal de correlación causal. Esta metodología ayudó a identificar cómo la automatización de pruebas de fin a fin basadas en BDD, influyeron en las

pruebas de regresión de los requerimientos ágiles. Se utilizó SCRUM como marco ágil. En sus resultados resume el éxito de la implementación, ya que logró reducir en un 60% el número de casos de prueba ejecutados manualmente y en un 50% el tiempo de ejecución respecto al tiempo invertido en dichas pruebas.

Capcha Coronado (2018). "*Implementación del Framework de automatización de proceso de QA en un proyecto de diseño de software en una consultora"*. Resumen: Plantea la implementación de un marco de trabajo de automatización de las actividades de calidad en un proyecto de diseño de software, donde el problema principal es el tiempo necesario para ejecutar casos de prueba de regresión manual después de cada compilación. Para ello expuso como meta disminuir el tiempo que se tarda en la ejecución de los casos de prueba sin la participación humana. Añadió una nueva actividad denominada "Automatización de casos de prueba dentro del proceso de aseguramiento de la calidad" y finalmente crear un framework de automatización. La metodología que utiliza es el marco ágil scrum, apoyado en las actividades de creación, estimación y construcción de requisitos de usuario para todo el proceso de aseguramiento de la calidad; consiguiendo como resultado: Reducción considerable en el tiempo de ejecución de casos de prueba de días a horas, gracias a la implementación eficiente del marco de automatización que sirvió como apoyo de control de calidad.

2.1.2. Antecedentes Internacionales

Cortés Pabón (2020) "*Automatización de pruebas de regresión para reducción de tiempo de entrega de nuevas versiones de software"*. Resumen: El trabajo se realizó en la compañía Surecomp, la cual se dedica a la fábrica de sistemas de software para la transferencia de datos financieros. El problema delimitado fue la falta de un proceso formal de calidad de software que ayudara a establecer los métodos

necesarios para proporcionar productos con los patrones solicitados por el mercado, tales como: la eficiencia en la gestión de los plazos y la documentación al momento de suministrar una nueva versión del software. El objetivo de esta tesis fue automatizar las pruebas de regresión para reducir el esfuerzo que implica realizar estas pruebas manualmente. La metodología utilizada es el análisis cuantitativo-cualitativo sustentado en el desarrollo de las actividades de referenciación y estado del arte, definición del proceso de calidad, selección de los casos de prueba, automatización, ejecución de casos, toma de métricas y evaluación de la solución. En sus resultados concluye el éxito de su implementación, debido a se logró reducir el tiempo total dedicado a las pruebas de regresión del 60% al 37,42%. Además, en una encuesta aplicada a los diversos roles participantes en la ejecución del proyecto se obtuvieron respuestas satisfactorias con respecto a proveer productos de excelente calidad y valor para los usuarios finales. Todo ello gracias a las pruebas automatizadas. Además, la implementación de este proyecto facilitó abrir un nuevo servicio de automatización de pruebas, que la compañía Surecomp puso en marcha en las sucursales de Israel y Alemania

También Rivera Martínez (2018). "*Automatización de pruebas de regresión".* Resumen: En su trabajo menciona que existen elementos cruciales en el proceso de construcción de software para adquirir sistemas de calidad: procesos, personas y tecnología. Donde los procesos tienen un impacto considerable en la calidad del producto. Los sistemas en la actualidad son complicados de probar, es por ello que la automatización de pruebas de software es una estrategia adoptada en muchas compañías actualmente. La problemática principal que presentaba la empresa de televisión donde se realizó el trabajo de tesis, es debido a que las actividades de pruebas se realizaban de manera manual y las consecuencias derivadas de estas

fueron: el reconocimiento retardado de errores y la ineficiencia en el uso de recursos. El propósito primordial de la tesis fue delimitar, establecer e implementar un proceso de prueba centrado en la automatización de pruebas de regresión que sea ventajoso en términos de tiempo y esfuerzo en comparación con la realización de pruebas manualmente. La metodología utilizada para automatizar fueron el método de separación funcional, de palabra clave y el método de pruebas basado en acciones. Para la ejecución de los scripts automatizados se construyó un instrumento de automatización de pruebas de regresión, que fue desarrollada a partir de dos modelos iniciales. Todo ello redujo las incidencias en el sistema de ventas y atención al cliente en casi un 50%, redujo el time-to-Market a 65 días y redujo las horas-hombre en un 30%.

Borio y Paterno (2021). "*Automatización de pruebas de regresión"*. Resumen: El objetivo de la investigación fue indagar e implementar un proyecto de pruebas que aborde la automatización de las pruebas de regresión. En ella se analiza detalles relacionados con la configuración del proyecto en cuestión, el contexto de su creación, las referencias de las pruebas manuales y diferentes aspectos que motivaron en participar y desarrollar el trabajo. La metodología utilizada es de tipo descriptivo, debido a que se analizan varias herramientas de testing automatizado con la finalidad de realizar una comparativa de los beneficios y desventajas de cada uno. Con el trabajo se comprobó que las herramientas de prueba de aplicaciones web Selenium WebDriver y Cypress ayudan a aumentar la velocidad de realización de las pruebas, optimizar el control de la calidad y reducir costos y tiempos mediante la sustitución de proceso manuales.

2.2. Bases Teóricas de las variables

En este apartado se explican las bases teóricas de cada una de las variables de la investigación.

2.2.1. Automatización de Pruebas Funcionales

2.2.1.1. Definición de Automatización de Pruebas

Existen diferentes conceptos sobre la automatización de pruebas funcionales.

Toledo, Curcio, y Scuoteguazza (2014) afirman que "La automatización de pruebas funcionales se llevan a cabo mediante una herramienta que ejecuta los casos de prueba automáticamente, y lee de algún modo la definición de los mismos, que pueden ser scripts en un programa en específico. Logrando al final reducir los costos, aumentado la cobertura y el alcance de las pruebas".

También, Fewster y Graham (2000) definen de manera similar declarando que la automatización de las pruebas de software puede reducir significativamente el esfuerzo requerido para realizar las pruebas adecuadas o aumentar significativamente las pruebas que se pueden realizar en un tiempo limitado. Las pruebas se pueden ejecutar en minutos, lo que llevaría horas ejecutar manualmente. Sin embargo, para que se logren los beneficios en la reducción del tiempo y costo de la automatización, indica que la pruebas a automatizar deben ser cuidadosamente seleccionadas e implementadas con un enfoque correcto.

Por otro lado, Hayes (1995) afirma que las pruebas de software de entrega automatizada brindan tres beneficios clave: cobertura acumulativa para detectar errores y reducir el costo de fallas, repetibilidad para ahorrar tiempo y reducir el costo de comercialización y aprovechamiento para optimizar el rendimiento de los recursos.

Hoy en día, la necesidad de velocidad(agilidad) es prácticamente el mantra de la era de la información. Debido a que la tecnología actualmente se utiliza como un

arma competitiva en primera línea de la interacción con el cliente. Los cronogramas de entrega están sujetos a las presiones del mercado. Los productos tardíos pueden generar pérdida de ingresos, de clientes y cuotas de mercado. Pero las presiones económicas también exigen reducciones de recursos y también de costos; lo que lleva a muchas empresas a adoptar la automatización de pruebas funcionales para disminuir el time to Market y recortar los presupuestos de prueba.

Llegar tarde al mercado puede ser costoso, más puede ser catastrófico entregar un producto defectuoso. Las fallas de software pueden costar millones o incluso miles de millones y, en algunos casos, han hecho que los proyecto fracasen. Primero hay que enfocarse en la mejora del proceso de pruebas para después implementar la automatización para aumentar la cobertura, crear un producto confiable y reducir la probabilidad de fallas en producción.

2.2.1.2. Beneficios de la Automatización de Pruebas

Son muchos los beneficios que se pueden obtener al implantar un entorno de trabajo de automatización de pruebas funcionales con el enfoque correcto y en un proceso definido de pruebas. Van desde la reducción del costo hasta la mejora en la confianza de la calidad del producto. Veamos fundamentos de autores sobre el tema.

Toledo et al. (2014) mencionan que “los beneficios que se obtienen de las pruebas automatizadas son: la identificación de defectos de manera anticipada, direccionar de manera eficiente las pruebas manuales, aumentar el alcance y la cobertura de las pruebas, disminución de costos; y de ese modo mejorar la calidad y reducir los riesgos en los productos de software y el negocio”.

Las pruebas automatizadas agregan valor al negocio mejorando la calidad, reduciendo el tiempo de puesta en marcha del producto y de los costos asociados en la corrección de defectos de software en diferentes etapas. También agrega valor al

proceso de control de calidad, porque permite simplificar tareas rutinarias y enfocar los esfuerzos en otras tareas relacionadas.

También Fewster y Graham (2000) manifiestan que los beneficios de la automatización de pruebas son el esfuerzo mínimo que se emplea en las pruebas de regresión, mayor prueba en menos tiempo, pruebas difíciles y a gran escala, mejor uso de recursos, consistencia y repetibilidad de las pruebas, reutilización de las pruebas debido a que el esfuerzo empleado en decidir qué probar, diseñar las pruebas y construir las pruebas se puede distribuir entre muchas ejecuciones de esas pruebas, tiempo de comercialización más temprano y mayor confianza al conocer que un amplio conjunto de pruebas automatizadas se ha ejecutado con éxito.

Por otro lado, Patton (2001) afirma que las pruebas automatizadas favorecen en la velocidad de la ejecución de las pruebas, optimizan las pruebas porque reducen el tiempo de la ejecución y agrega más tiempo para planificar y pensar en nuevas pruebas, en la exactitud y precisión al comprobar los resultados y la repetibilidad de las pruebas, porque se pueden ejecutar las pruebas varias veces.

2.2.1.3. Que casos de prueba se debería de automatizar

En el proceso de control de calidad del software se desarrollan estrategias para identificar y elegir los casos de prueba a automatizar. Estas están sujetas al nivel riesgo, complejidad, nivel de incertidumbre, rutinas, cambios y costo en la que se encuentra la funcionalidad a probar. Veamos diversos enfoques y consideraciones de autores sobre los requisitos que deben cumplir para seleccionar los casos de prueba a automatizar.

De acuerdo a Toledo et al. (2014) establecen como criterios de automatización, el testing basado en riesgos. Es decir, funcionalidades que tienen mayor riesgo, porque si llegarán a fallar serían los más costosos y generarían muchas

consecuencias negativas. Para ello considera criterios como la severidad e importancia para las operaciones del negocio, costo de fallas en producción y acuerdos de nivel de servicio. También considera otros criterios como la repetibilidad de las pruebas, la cantidad de veces que será desplegado la aplicación que se va a validar, las diferentes plataformas donde se va a aplicar y los diferentes tipos de datos. Todo ello se debe tener en cuenta para decidir si automatizar o no.

También Fewster y Graham (2000) de manera similar sostienen que del conjunto de pruebas a automatizar se debe de realizar una subselección en base la cantidad de veces y el tiempo que tomaría en ejecutarlos, por la dificultad de escribir una prueba y el factor molestia en ejecuciones posteriores. También consideran como criterios a las pruebas de amplitud, las pruebas para las funciones más importantes, pruebas que son fáciles de automatizar, pruebas que darán la recuperación más rápida y las pruebas que se ejecutan con más frecuencia.

Por otro lado (Hayes, 1995) considera de manera más general que para iniciar con la automatización de pruebas hay que conocer con anticipación el comportamiento esperado de la aplicación, controlar el entorno y los datos de prueba, considerar un diseño estable, caso contrario se terminará con una complejidad excesiva, una alta probabilidad de falla en la prueba y mayores costos de mantenimiento. También menciona como criterios fundamentales para la automatización contar con probadores con experiencia, tiempo y recursos.

Existen muchos criterios a tomar en cuenta antes de iniciar con la automatización de pruebas de software, y estos dependerán del grado de madurez de los procesos de la organización, de los objetivos de la automatización, del conocimiento, de los costos relacionados y de los recursos disponibles.

Algunos criterios a considerar para automatizar son los riesgos asociados a una funcionalidad, las tareas repetitivas, aplicaciones con pocos cambios en su diseño, funcionalidades complejas, funcionalidades acumulativas(regresión), aplicaciones multiplataforma (diferentes sistemas operativos, navegadores y configuraciones) y maximizar el alcance de las pruebas reduciendo el tiempo utilizado en las pruebas.

2.2.1.4. Enfoque de Automatización de Pruebas

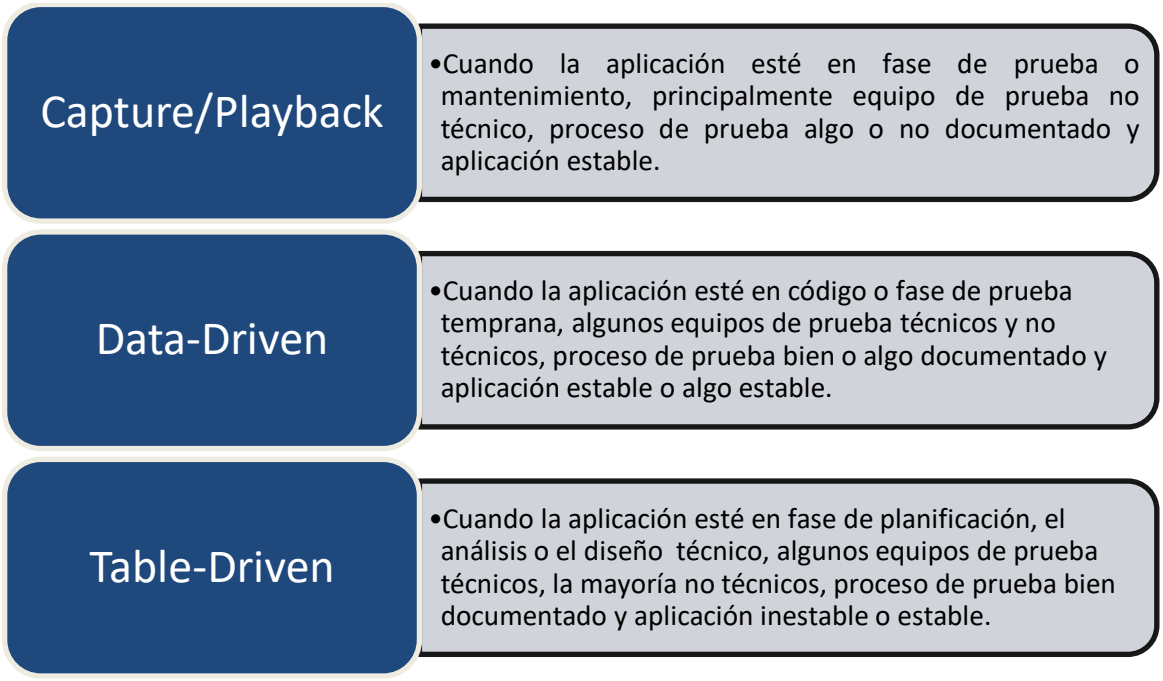
Existen varias formas de abordar la automatización de pruebas, como el empirismo de los probadores, entornos de automatización, arquitecturas y herramientas de automatización. Ahora revisaremos a detalle algunos de los enfoques en los siguientes párrafos.

Hayes (1995) sostiene que antes de optar por un enfoque de automatización se debe de realizar una evaluación que este diseñada para ayudar a identificar la posición en términos de la aplicación, equipo de prueba y proceso de prueba. En función de los resultados obtenidos, se podrá seleccionar el enfoque de automatización adecuado para las necesidades. Para ello propone preguntas como: ¿En qué fase de desarrollo esta la aplicación?, ¿Cuál es el grupo de habilidades del equipo de pruebas?, ¿Qué tan bien está documentada el proceso de pruebas? Y ¿Qué tan estable es la aplicación? De acuerdo a las respuestas propone tres enfoques de automatización: Capture/Playback, Data-Driven y Table-Driven.

En la figura 2, de detallan los enfoques de automatización y las consideraciones a evaluar para elegir su aplicabilidad.

Figura 2

Enfoques de Automatización de Pruebas – ¿Cuándo Aplicar?



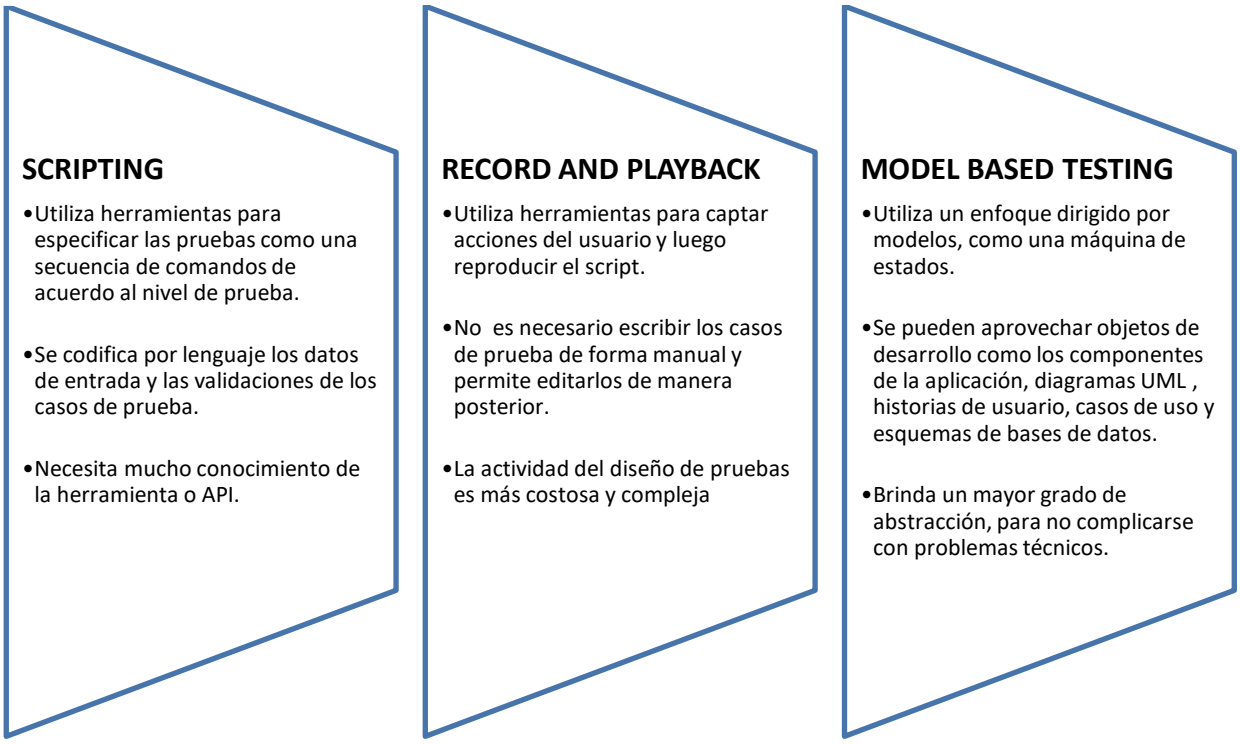
Nota. Enfoques de Automatización de Pruebas esquematizados a partir de una evaluación interna del proceso de control de calidad de la organización. Elaboración propia.

De manera similar Toledo et al. (2014) proponen tres enfoques de automatización y afirman que estas deben de aplicarse de acuerdo al contexto en que se utilizaran. El scripting es uno de los enfoques más comunes que utiliza herramientas(Selenium, Junit y APIS) que proporcionan lenguajes de programación que permiten declarar los casos de pruebas como una serie de acciones; Récord and Playback, ayuda a especificar el procedimiento de la prueba a través de una grabación y que luego se ejecutará sobre el mismo sistema; y finalmente el testing basado en modelos que implica automatizar la ejecución y diseño de las pruebas a través de modelos, como por ejemplo una máquina de estados.

En la figura 3, se detallan los enfoques de automatización de acuerdo al contexto en que se emplearan.

Figura 3

Enfoques de Automatización de Pruebas – Según el Contexto



Nota. Enfoques de Automatización de Pruebas esquematizados según el contexto. Elaboración propia.

Por otro lado, Fewster y Graham (2000) sostienen que la automatización de pruebas es más que capturar y reproducir (Récord and Play) el caso de prueba. Es tentador pensar que la forma más rápida de automatizar las pruebas es simplemente activar la función de grabación de una herramienta de reproducción de captura cuando el probador se sienta a probar. Luego, la herramienta puede reproducir la grabación, repitiendo así la prueba exactamente como se realizó manualmente. Concluye que es una equivocación pensar así, debido a que la automatización incluye también el diseño y la verificación automatizada de la prueba.

2.2.1.5. Herramientas de Automatización de Pruebas

En la actualidad existen muchas herramientas y frameworks para la automatización de pruebas. Hay herramientas de código abierto y de pago. Existen

para los diferentes tipos y niveles de pruebas. Por ejemplo, tenemos a Selenium Web Driver para pruebas funcionales web en diferentes plataformas; Appium para pruebas móviles; SoapUI, Postman y RestAssured para pruebas de APIs; Jmeter para pruebas de performance, carga y stress. También existen frameworks que facilitan el proceso de automatización. Dentro de este grupo tenemos a Junit y TestNG basados en lenguaje java y también Cucumber basado en DBB – Desarrollo basado en el comportamiento - que hace posible automatizar los casos de prueba BDD.

En los siguientes párrafos se revisará cuáles son las consideraciones según los autores expertos en el tema, para elegir uno u otra herramienta de automatización.

Patton (2001) afirma que para seleccionar una herramienta de automatización es necesario definir con anterioridad el tipo de software que se va a probar y los tipos de pruebas (caja blanca o de caja negra) que se van a realizar. Considera dos tipos de herramientas: No invasivo, utilizado solo como monitor y para examinar el software sin modificarlo; e Invasivo, que modifica el software puesto a prueba. Para las pruebas automatizadas considera herramientas de “grabado y ejecución”, “macros programados” y “herramientas de pruebas automatizadas programables”.

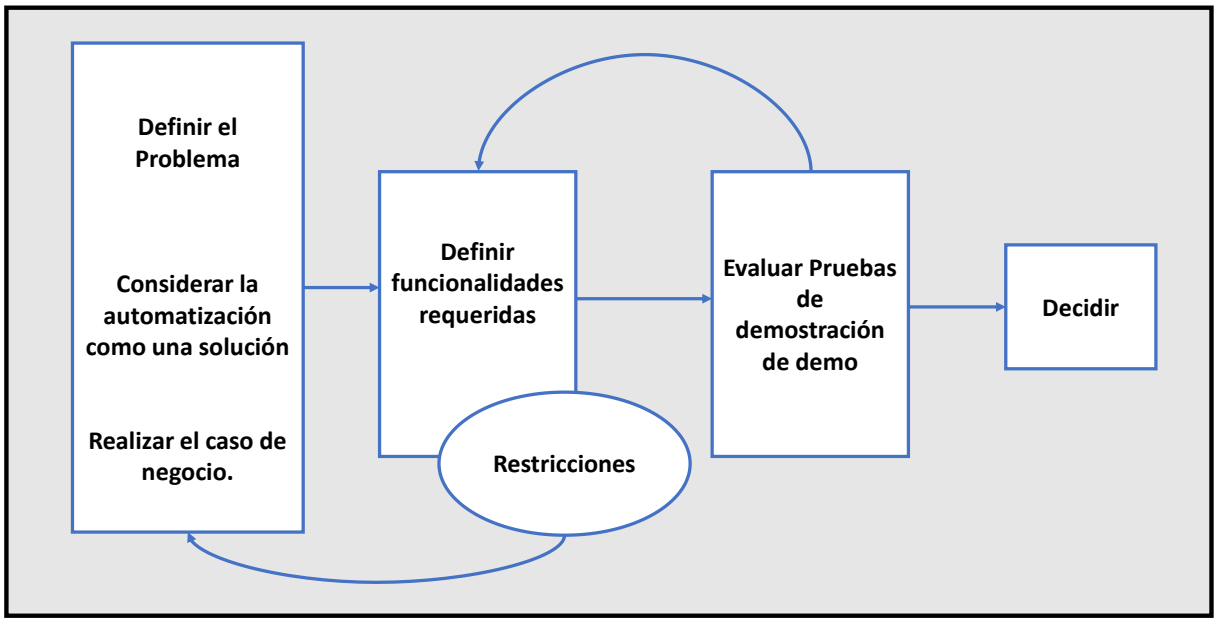
En su análisis sobre las herramientas de automatización, Ron Patton no se enfoca en estudiar a detalle las herramientas que se mencionaron en la introducción de este acápite, sino que los menciona de manera más generalizada.

Por otro lado, Fewster y Graham (2000) establecen que, para seleccionar una herramienta de pruebas automatizadas, se tiene que realizar primero una evaluación de las ofertas del mercado y elegir solamente una y la que se ajuste más a la organización. Después de ello la herramienta se implementará en la organización a través de un proceso. También acentúa que para la elección de la herramienta se

tiene que tomar en cuenta la magnitud de las personas que las van a utilizar y en base a ello proponer un proyecto para la obtención de la herramienta.

Figura 4

Proceso de selección de una herramienta de automatización de pruebas



Nota. Definir proyecto para la adquisición de una herramienta de automatización.

2.2.1.6. Dimensiones de Automatización de Pruebas

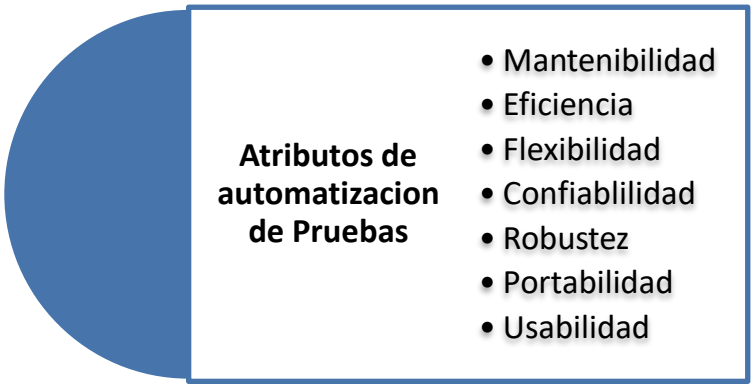
La automatización de pruebas basada en scripts automatizados contiene muchos atributos como repetición, exactitud, reutilización, programación entre otros. En los siguientes párrafos se detallará las concepciones que tienen algunos autores sobre el tema.

Fewster & Graham (2000) Sostienen que, los atributos de la automatización de pruebas que deben medirse tienen que estar relacionados con los objetivos de automatización que se planteó en un inicio. Afirman también que, no es necesario elegir todos los atributos, sino los necesarios. Estos son los atributos que ellos platean: La mantenibilidad, referido a la facilidad y rapidez de la actualización de las pruebas automatizadas; La eficiencia, relacionado al costo, que permitan realizar pruebas con menos esfuerzo y en menos tiempo; La confiabilidad, referido a brindar resultados

precisos y repetibles; La flexibilidad, que permite trabajar con diferentes subconjuntos de pruebas; La Usabilidad referido a la facilidad de uso por parte de los usuarios; La Robustez, relacionado a la solidez y estabilidad para hacer frente a una gran variedad de cambios en el software; y finalmente la portabilidad, relacionado a la habilidad de ejecutar pruebas en diferentes plataformas.

Figura 5

Dimensiones de un régimen de automatización de pruebas



Por otro lado, Hayes (1995) afirma que, es un error suponer que la automatización de pruebas es simplemente la captura y reproducción de un proceso de prueba manual. De hecho, la automatización es fundamentalmente diferente de las pruebas manuales: existen problemas y oportunidades completamente diferentes. Incluso la mejor automatización nunca reemplazará por completo las pruebas manuales, porque la automatización se trata de previsibilidad y los usuarios son inherentemente impredecibles. Por lo tanto, use la automatización para verificar lo que espera y use pruebas manuales para lo que no espera. Por consiguiente, para el éxito en la automatización es necesario comprender los fundamentos y atributos del proceso que son: Mantenibilidad, optimización, independencia, modularidad, contexto, sincronización y documentación.

Tabla 5

Fundamentos y Atributos de la Automatización de Pruebas

Fundamentos/atributos	Explicación
Mantenibilidad	Los scripts de pruebas deben ser fácil de actualizarse.
Optimización	La optimización es importante para asegurarse de tener suficientes pruebas para hacer el trabajo sin tener que administrar demasiados.
Independencia	La independencia se refiere al grado en que cada caso de prueba es independiente. Es decir, ¿el éxito o el fracaso de un caso de prueba depende de otro y, de ser así, ¿cuál es el impacto de la secuencia de ejecución?
Modularidad	secuencias de comandos que pueden ser eficientes ensamblado para producir un sistema unificado sin redundancia u omisión.
Contexto	Se refiere al estado de la aplicación durante la reproducción de prueba. Debido a que una prueba automatizada se ejecuta al mismo tiempo que la aplicación, es fundamental que permanezcan sincronizados.
Sincronización	La sincronización entre la prueba y la aplicación requiere que se ejecuten a la misma velocidad
Documentación	En un momento crítico, la biblioteca de prueba podría ejecutarse manualmente.

2.2.2. Pruebas de Software

2.2.2.1. Definición de Pruebas de Software

Existen diversos conceptos con respecto a las pruebas de software. Este término está muy asociado con el concepto de pruebas de calidad, y otros. Veamos algunas definiciones.

Por otro lado, la ISTQB (2018) define a “las pruebas de software como un proceso que contiene varias actividades distintas, que van desde la planificación, el análisis, el diseño, la implementación, ejecución, informe de avances, cierre y seguimiento y control de las pruebas”.

GLENFORD, BADGETT, y SANDLER (2012) definen a las pruebas de software como un proceso o conjunto de procesos, elaborados para garantizar de que el código

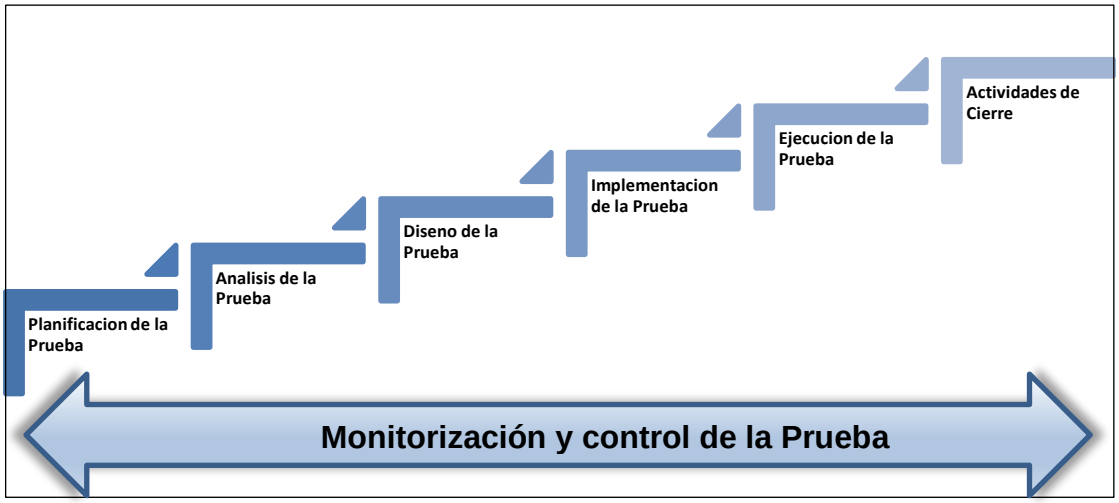
de la computadora realice para lo que fue construido hacer y viceversa, que no haga nada indeseable. El software debe ser previsible, coherente y no sorprender a los usuarios.

De la misma manera Pressman (2010) define las pruebas como un grupo de tareas que pueden diseñarse de antemano y llevarse a cabo de manera estructurada. Por esta razón, en el transcurso del proceso de software, se debe establecer una plantilla para las pruebas: una serie de acciones secuenciales que comprenden métodos de prueba y técnicas específicas de diseño de casos de prueba.

Por tanto, se concluye en base a las revisiones teóricas que las pruebas de software no están enmarcadas en la sola actividad de ejecución, sino más bien es un proceso sistemático de actividades estrechamente relacionadas que van desde la planificación hasta el cierre y cuya finalidad principal es prevenir y detectar defectos en el sistema bajo prueba.

Figura 6

Actividades y tareas del proceso de pruebas de Software



Nota. La figura ha sido elaborada con la teoría abstraída del manual del probador certificado – ISTQB.

2.2.2.2. Niveles y Tipos de Prueba

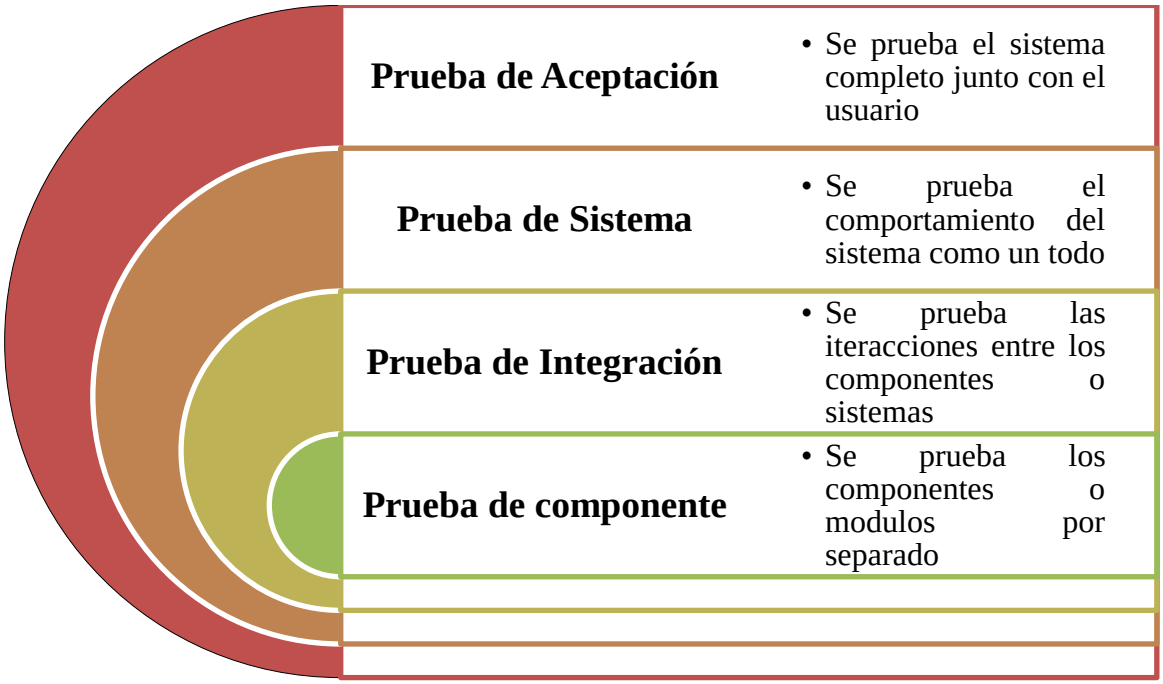
Los tipos y niveles de pruebas están sujetas al modelo de ciclo de vida de desarrollo de software que cada organización tiene implementado. Independiente al modelo a cada actividad de desarrollo le corresponde una actividad de prueba, cada nivel tiene sus propios propósitos. Los probadores participan desde el inicio del proyecto y las actividades de prueba deben empezar lo más antes posible. (ISTQB, 2018).

2.2.2.2.1. Niveles de Prueba

Según (ISTQB, 2018) los niveles de prueba son un conjunto de actividades que se gestionan y organizan de manera conjunta. Cada nivel es una representación del proceso de prueba y donde se tienen que desarrollar las actividades de prueba de manera imperativa.

Figura 7

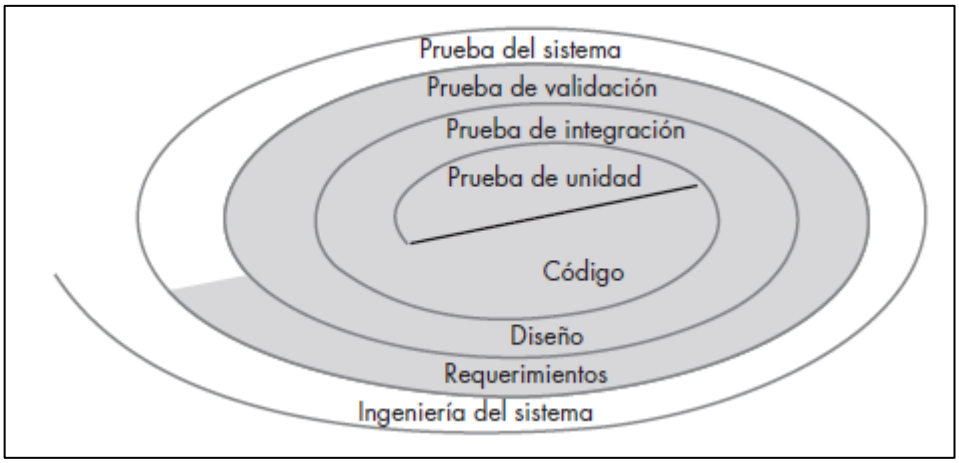
Niveles de Pruebas - ISTQB



De manera similar Pressman (2010) sostiene que el proceso de desarrollo de software debe realizarse en forma de una curva, que comienza con la captura de los requerimientos del sistema y este va avanzando hacia su punto más interno, pasando por el análisis, diseño y finalmente la codificación. Y las pruebas deben de avanzar de manera contraria. Desde su punto más interno, las pruebas unitarias, avanzando a las pruebas de integración, de validación y finalmente las pruebas del sistema.

Figura 8

Estrategia de Pruebas de Software – Visión General



Nota. Pressman, R. (2010), *Ingeniería del Software. Un enfoque práctico. 7ta edición.* p. 386

2.2.2.2.2. Tipos de Prueba

Un tipo de prueba es una serie de tareas que evalúan características de comportamiento, no funcionales como la seguridad y la portabilidad, la estructura interna de un componente, también verifican la resolución de cambios y efectos no deseados en el sistema durante su mantenimiento. (ISTQB, 2018).

En base a los estipulados anteriores los tipos de pruebas se categorizan en: Prueba funcional, prueba no funcional, prueba de caja blanca y prueba asociadas al cambio.

Tabla 6

Tipo de pruebas de software

Tipo de Prueba	Descripción	Ejecutado en Nivel	Tipo de Técnica
Prueba Funcional	Evalúan el comportamiento y las funciones que deben de cumplir los sistemas a partir de las especificaciones del sistema o componente	En todos los niveles de prueba.	De caja negra
Prueba No Funcional	Evalúan características no funcionales del producto de software como la usabilidad, amigabilidad, portabilidad, seguridad, entre otros.	En todos los niveles de prueba	De caja negra
Prueba de caja blanca	Evalúan la composición interna del producto de software y estos pueden incluir: código, arquitectura y flujo de datos	En el nivel Prueba unitaria	De caja blanca
Prueba Asociada al cambio	Evalúan los cambios surgidos en el sistema a raíz de corrección defectos después de puesta en producción. Existen dos tipos de prueba **Pruebas de confirmación**: Valida que el defecto ya no exista. **Prueba de Regresión:** Valida que la corrección de una falla no afecte otras funcionalidades.	En todos los niveles de prueba	Principalmente de caja negra

2.2.2.3. Pruebas en un entorno ágil

El desarrollo ágil promueve el desarrollo iterativo e incremental, con pruebas significativas, que está centrado en el cliente y acepta cambios durante el proceso. Todos los atributos de los enfoques tradicionales de desarrollo de software descuidan o minimizan la importancia del cliente. Aunque las metodologías ágiles incorporan flexibilidad en sus procesos, el énfasis principal está en la satisfacción del cliente. El cliente es un componente clave del proceso; En pocas palabras, sin la participación del cliente, el método Agile falla. (GLENFORD, BADGETT, & SANDLER, 2012)

El aumento de la competencia y la interconexión en todos los mercados han obligado a las empresas a acortar su tiempo de comercialización sin dejar de ofrecer productos de alta calidad a sus clientes. Esto es particularmente cierto en el sector de construcción de software, en donde las tecnologías de la información hacen posible el

suministro rápido de aplicaciones y servicios de software. Hoy en día las metodologías agiles han desplazado muchísimo a los entornos de trabajo tradicionales, porque su principal característica es entregar software funcional en menos tiempo y generar valor en el negocio. Sin embargo, la rapidez no es sinónimo de calidad, por ello las pruebas de software juegan un papel importante al garantizar la calidad y se potencian más en entornos agiles de desarrollo.

En la siguiente tabla veremos a manera de resumen los principales entornos agiles de desarrollo y pruebas.

Tabla 7

Entornos de trabajo de desarrollo y pruebas de software ágil

Entorno	Descripción
Scrum	Es un conjunto de buenas prácticas muy utilizada y extendida en el desarrollo de productos y servicios. Se basa en el prototipado rápido, flexibilidad y la adaptabilidad durante el sprint. Utiliza eventos y artefactos cuya función principal es transparentar e inspeccionar el trabajo del equipo de desarrollo. Artefactos (Sprint Backlog, Product Backlog y BurnDownChart) y Eventos (Sprint, Planeamiento, Reunión diaria, Revisión y Retrospectiva del sprint)
Extreme programming (XP)	Metodología que se enfoca en el progreso de la construcción en lugar de los componentes de dirección de proyectos. Fue creado de tal manera que las compañías tuvieran la oportunidad de comprar toda o parte de la metodología. Las iniciativas de XP inician con la fase del plan de lanzamiento, continuado de varios ciclos, cada una de las cuales finaliza con la prueba de aceptación del usuario. Una vez que el producto tenga todos los requisitos para satisfacer a los usuarios, el equipo de dejará de iterar y lanzará el software.
Lean Development	Propone reducir las tareas sin valor agregado perfeccionando el flujo y suprimiendo la congestión. Definir Lean en relación con el software es difícil ya que no existe una metodología o proceso específico asociado con él.
Kanban	Se centra en optimizar la transparencia de la forma de trabajo, delimita el alcance del trabajo a realizar conforme a la disponibilidad de recursos y mide la duración de una actividad. Para lograr ello fracciona el esfuerzo en grupos, donde cada tarea se cataloga en alguno de los grupos y se coloca en un panel principal, conocido como tablero Kanban

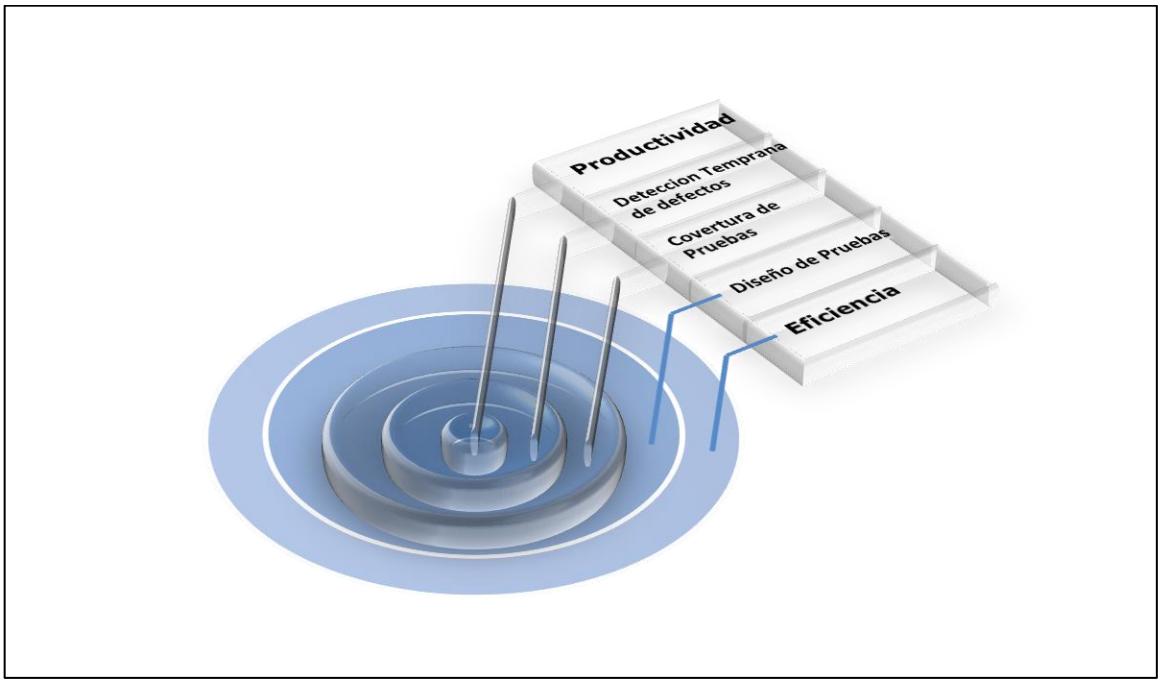
Nota. Adaptado (Cabrera Serna & Pareja Verastegui, 2021)

2.2.2.4. Dimensiones de Pruebas de Software

La validación de la funcionalidad de un producto de software se alcanza mediante un conjunto de pruebas que determinan la aceptación respecto a los requisitos de usuario. Se desarrolla un proceso de pruebas que garantice: la productividad de las pruebas, la detección temprana de defectos, coberturar lo más posible las funcionalidades del sistema, disminuir el impacto de fallas en producción y diseñar scripts de pruebas automatizados que ayuden a ampliar el alcance de las pruebas con menor esfuerzo y costo. (Pressman, 2010)

Figura 9

Dimensiones de la prueba de software, desde el contexto de la calidad – (Pressman, 2010)

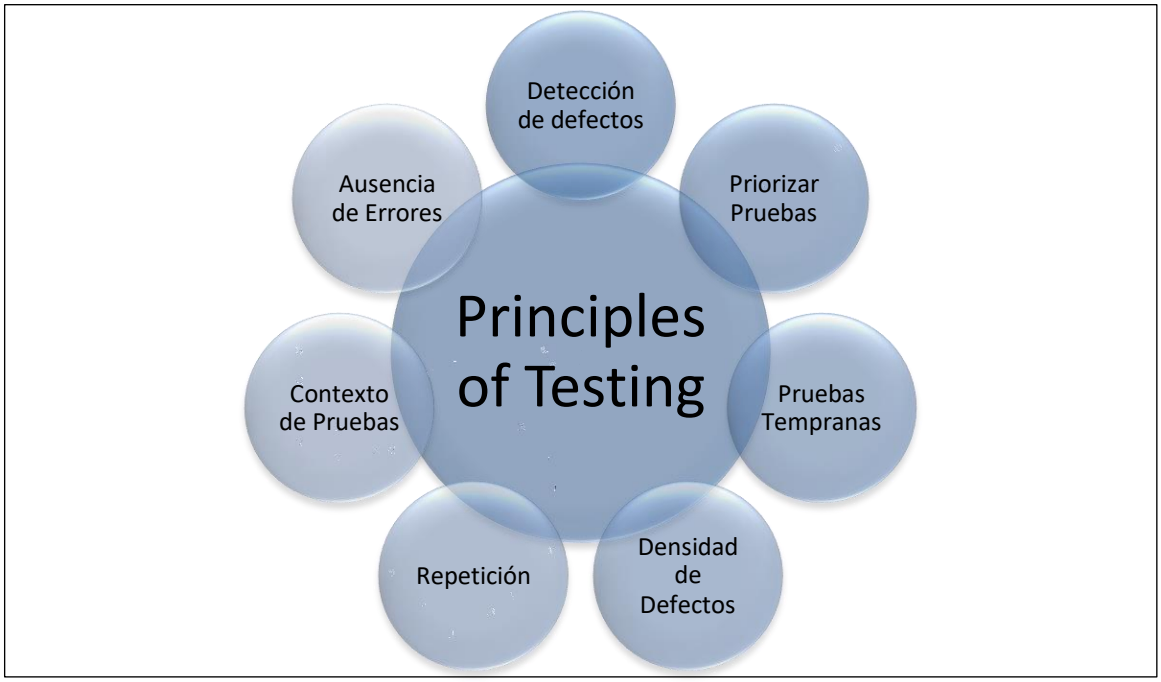


Por otro lado, ISTQB (2018) menciona que en el proceso de pruebas de software se tienen que seguir 7 principios básicos que ayudaran a la efectividad de las pruebas, y estas son: La prueba no garantiza la ausencia de defectos, es decir ayuda a reducir al máximo la probabilidad de que queden defectos no descubiertos;

Es imposible probar todas las combinaciones y entradas, se tiene que utilizar análisis basada en riesgos y técnicas de pruebas para orientar los esfuerzos; La detección temprana de errores ayuda a eliminar y recudir costo de reparación de defectos; La agrupación de los defectos ayuda a centrar los esfuerzos para solucionarlos; La repetición de las mismas pruebas no es lo más optimo, es necesario cambiar las pruebas para encontrar nuevos defectos; La prueba depende del contexto, es decir que va a depender del sistema bajo prueba y Que encontrar todos los defectos posibles no asegura el éxito de un producto de software.

Figura 10

Atributos de la prueba de software, desde los 7 principios del Testing – (ISTQB, 2018)



2.2.2.5. Métricas de Pruebas de Software

En el proceso de pruebas de software el propósito primordial es la detección de defectos y el cumplimiento de la cobertura funcional. Para garantizar la eficiencia de

este proceso se tiene que controlar y para controlar hay que medir y de acuerdo a los resultados conseguidos se tomaran acciones de mejora.

Fewster y Graham (2000) afirman que, los objetivos principales de las pruebas son establecer confianza y encontrar defectos. Ante ello proponen algunas medidas de eficacia, cuya elección está sujeta al criterio de la propia organización. Entre las medidas está el porcentaje de detección de defectos (DDP), porcentaje de corrección de defectos (DFP), Medida relacionada con la confianza de la eficacia de la prueba

Tabla 8

Métricas del proceso de pruebas de software según Fewster & Graham

Métrica	Formula	Explicación
Porcentaje de detección de defectos	DDP=Defectos encontrados por pruebas/Total defectos conocidos	El número total de defectos conocidos es la cantidad de defectos encontrados por testing más el número de defectos encontradas después.
Porcentaje de corrección de defectos	DFP=Defectos corregidos antes del despliegue/Total de defectos encontrados	Consiste en la efectividad de la depuración para la eliminación de un defecto.
Eficacia de la Prueba	Tabla de ratio de confianza	Puntuación de los analistas sobre la confianza de las pruebas.
Cobertura de Pruebas	% de Casos pruebas que cubren cierta funcionalidad.	Se expresa en porcentaje. Muy utilizado en pruebas de regresión.

De manera similar (Pressman, 2010) contempla diversas medidas, tanto para el producto como para el proceso. En las pruebas de software considera las siguientes métricas: Eficiencia en la remoción del defecto, productividad de las pruebas en tiempo y costo, detección temprana de defectos, diseño de pruebas y cobertura de pruebas de regresión.

Por otra parte, (Hayes, 1995) menciona que, las métricas de prueba son simplemente medidas de su proceso de prueba que lo ayudarán a determinar dónde se encuentra la aplicación y cuándo estará lista para su lanzamiento. En un mundo

ideal, mediría sus pruebas en cada fase del ciclo de desarrollo, obteniendo así una visión objetiva y precisa de qué tan exhaustivas son sus pruebas y qué tan cerca cumple la aplicación con sus requisitos. Describe de manera explicativa los ítems que se pueden medir en el proceso de pruebas.

El resumen de las métricas propuestas en la siguiente tabla:

Tabla 9

Métricas del proceso de pruebas de software según Linda Hayes

Métrica	Explicación
Medida de Progreso	La prueba a veces es una tarea interminable, entonces se tiene que medir para para establecer el progreso y determinar el punto donde se tiene que liberar el sistema.
Cobertura de Código y de Requerimientos	La cobertura de código es una medida de qué porcentaje del código fuente de la aplicación subyacente se ejecutó durante la prueba. Y la cobertura de requerimientos el porcentaje de requerimientos probados por un conjunto de casos de prueba.
Criterios de salida.	Permite determinar un punto hasta donde se debe de probar para lanzar el producto a producción. Ejemplo cobertura exitosa de requisitos.
Cobertura de casos de prueba	Cantidad de Casos pruebas que se ejecutaron y determinar el % de pruebas exitosas y fallidas.
Ratio de defectos	La tasa de defectos mide cuántos errores se encuentran como porcentaje de pruebas ejecutadas. Y confirmar si el error detectado es un defecto.
Tasa de corrección de defectos	Mide la cantidad de defectos corregidos, después de haberse informado. Considerar a tasa de corrección desde el momento en que se informa el defecto hasta que se introduce la corrección correspondiente en la biblioteca de origen.
Tasa de recurrencia	La tasa de recurrencia es el porcentaje de arreglos que fallan en corregir el defecto. Es extremadamente útil para medir la calidad de su unidad. y prácticas de prueba de integración
Defectos Post Release	Es una medida que permite determinar la cantidad de defectos encontrados después del lanzamiento de producto a producción.

De manera general, (ISTQB, 2018) establece algunas métricas que se pueden emplear para medir el proceso de pruebas. Entre ellas mencionan: Avance de las pruebas con respecto al cronograma y presupuesto, eficacia de las actividades de prueba, Porcentaje de avance de preparación de casos de prueba, porcentaje avance

en la preparación del entorno de prueba, cantidad de casos ejecutados correctos y fallidos, densidad de defectos, defectos encontrados y corregidos, cobertura de prueba, y uso recursos y esfuerzo.

Por lo tanto, es prescindible medir las pruebas en el proceso de ciclo de vida de desarrollo de software, porque gracias a los resultados obtenidos en las métricas se plantearán acciones de mejora continua cuya finalidad será madurar el proceso de prueba y generar confianza en la calidad del producto puesto bajo prueba.

2.3. Análisis comparativo de las Bases Teóricas

En esta sección se realiza una comparación de las bases teóricas para cada una de las variables de la presente investigación.

2.3.1. Automatización de Pruebas Funcionales

En la tabla 10, se realiza un análisis comparativo de las definiciones sobre la automatización de pruebas y sus limitaciones que tienen en la actualidad.

Tabla 10

Análisis comparativo de las definiciones sobre la Automatización de Pruebas

Autor	Definición	Aporte
(Toledo, Curcio, & Scuoteguazza, 2014)	La automatización de pruebas funcionales se realiza a través de una herramienta logre ejecutar los casos de prueba en forma automática, leyendo la especificación del mismo de alguna forma, que pueden ser scripts en un lenguaje en específico. Logrando al final reducir los costos, ampliar la cobertura y alcance de las pruebas	Esta definición es una de las más consistentes y cercanas porque menciona el uso de scripts automáticos y herramientas para la ejecución de casos de prueba. También hace énfasis en los beneficios.
(Fewster & Graham, 2000)	la automatización de las pruebas de software puede reducir significativamente el esfuerzo requerido para realizar las pruebas adecuadas o aumentar significativamente las pruebas que se pueden realizar en un tiempo limitado. Las pruebas se pueden ejecutar en minutos, lo que llevaría horas ejecutar manualmente. Sin embargo, para que se	Esta definición plasma de manera más directa los beneficios que se logran con la automatización de pruebas. No define el uso de herramientas de manera explícita. Sin embargo, su análisis nos ayuda a comprender que la

	logren los beneficios en la reducción del tiempo y costo de la automatización, indica que la pruebas a automatizar deben ser cuidadosamente seleccionadas e implementadas con un enfoque correcto.	automatización de pruebas es más que ejecutar scripts automáticos.
(Hayes, 1995)	afirma que las pruebas de software de entrega automatizada brindan tres beneficios clave: cobertura acumulativa para detectar errores y reducir el costo de fallas, repetibilidad para ahorrar tiempo y reducir el costo de comercialización y aprovechamiento para mejorar la productividad de los recursos.	Es una definición muy concisa, enfocado también en los beneficios de la automatización de pruebas. Pero, hace mucha referencia en el uso de scripts, herramientas y enfoques.

En la tabla 11, se realiza un análisis comparativo sobre los enfoques de la automatización de pruebas y sus diferentes aplicaciones.

Tabla 11

Análisis Comparativo de los Enfoques Sobre la Automatización de Pruebas

Autor	Enfoques	Aporte
(Toledo, Curcio, & Scuoteguazza, 2014)	El autor propone tres enfoques de automatización: • Scripting • Capture/Playback • Modelo Based Testing	De acuerdo al análisis de los tres enfoques, definitivamente el que tiene mayor potencial es el scripting y su auge al día de hoy se debe a la evolución de las herramientas y frameworks de automatización de pruebas gratuitos que han permito lograr scripts más escalables y portables. Además, este enfoque permite combinarse con el Data Driven y el Table Driven.
(Fewster & Graham, 2000)	El autor explica que: La automatización de pruebas es más que capturar y reproducir (Record and Play)	El autor no detalla de manera explícita los enfoques de automatización de pruebas. Sin embargo, hace mucho hincapié que la automatización de pruebas es más que capturar y reproducir. Este es un enfoque muy desfasado y su utilidad mayor está en las

		pruebas exploratorias y adhoc, donde ya no se necita mucha documentación.
(Hayes, 1995)	El autor propone tres enfoques de automatización: • Capture/Playback • Data-Driven • Table-Driven.	De los enfoques propuestos los que más se adecuan para organizaciones que tienen el proceso de prueba maduro son el Data y Table Driven respectivamente, porque permiten dinamizar la ejecución de prueba automatizadas. La elección de cualquiera de ellos dependerá de los resultados de la evaluación interna de la madurez del proceso de pruebas.

En la tabla 12, se plasma un análisis comparativo sobre las dimensiones(atributos) de la automatización de pruebas.

Tabla 12

Análisis Comparativo de las Dimensiones(atributos) Sobre la Automatización de Pruebas

Autor	Dimensiones (Atributos)	Aporte
(Sarco, 2019)	El autor propone los siguientes atributos. 1. Rapidez 2. fiabilidad (Rapidez) 3. Repetición 4. Programable 5. Reusable	Después de analizar y revisar los atributos que plantean los autores sobre la automatización de pruebas se debe de elegir el modelo que este más orientado a la diseño, construcción y mantenimiento de los scripts de automatización. En la evaluación de los tres modelos se observa que hay coincidencias significativas entre ellos. Sin embargo, se resalta el modelo de (Fewster & Graham,
(Fewster & Graham, 2000)	El autor propone los siguientes atributos. 6. Mantenibilidad 7. Eficiencia (Rapidez) 8. Flexibilidad (Repetibilidad) 9. Fiabilidad/Confiabilidad	

	10. Robustez (Programable) 11. Portabilidad 12. Usabilidad/Reusabilidad	2000), porque se ajusta más a lo que el presente estudio busca y además en su análisis se observa una explicación diferenciada entre las pruebas y la automatizacion de pruebas.
(Hayes, 1995)	El autor propone los siguientes atributos: 1. Mantenibilidad 2. Optimización 3. Independencia 4. Modularidad 5. Contexto 6. Sincronización 7. Documentación	

2.3.2. Pruebas de Software

En la tabla 13, se realiza una evaluación comparativa sobre las definiciones de las pruebas de software y su importancia en el control de la calidad.

Tabla 13

Análisis comparativo de las definiciones sobre las Pruebas de Software

Autor	Definición	Aporte
(ISTQB, 2018)	Las pruebas de software son un proceso que contiene muchas actividades diferentes, que van desde la planificación, el análisis, el diseño, la implementación, ejecución, informe de avances, cierre y seguimiento y control de las pruebas.	Después del análisis de concluye que las tres definiciones tienen coincidencias muy importantes. Sin Embargo, por temas de actualidad se va a elegir la definición de (ISTQB, 2018). Además, porque en su análisis incluye referencias de la ISO/IEC/IEEE 21119-2 con respecto al proceso de prueba.
(GLENFORD, BADGETT, & SANDLER, 2012)	Un proceso, o un conjunto de procesos, diseñados para asegurarse de que el código de la computadora haga lo que fue diseñado para hacer y, a la inversa, que no haga nada no deseado. El software debe ser predecible y consistente, sin presentar sorpresas a los usuarios	
(Pressman, 2010)	Un conjunto de actividades que pueden diseñarse de antemano y llevarse a cabo de manera estructurada. Por esta razón, en el transcurso del proceso de software,	

	se debe establecer una plantilla para las pruebas: una serie de acciones secuenciales que comprenden métodos de prueba y técnicas específicas de diseño de casos de prueba	

En la tabla 14, se realiza una evaluación comparativa sobre las dimensiones de las pruebas de software.

Tabla 14

Análisis Comparativo de las Dimensiones de las Pruebas de Software

Autor	Dimensiones	Aporte
(Pressman, 2010)	Dimensiones. 1. Productividad 2. Detección Temprana de defectos 3. Cobertura de Pruebas 4. Diseño de Pruebas 5. Eficiencia	Después de analizar los dos modelos presentados, definitivamente se va a optar por el de (Pressman, 2010), debido al enfoque procedimental que este tiene. No solamente se enfoca en las pruebas y detección de defectos, sino también al proceso en sí. Además, muestra las pruebas de software dentro de los procesos macro de aseguramiento de la calidad y calidad de software respectivamente
(ISTQB, 2018)	Dimensiones 1. Detección de defectos 2. Priorizar Pruebas 3. Pruebas Tempranas 4. Densidad de Defectos 5. Repetición 6. Contexto de Pruebas 7. Ausencia de Errores	

En la tabla 15, se realiza un análisis comparativo sobre las Métricas de las pruebas de software.

Tabla 15

Análisis Comparativo de las Métricas de las Pruebas de Software

Autor	Métricas	Aporte
(ISTQB, 2018)	Métricas: 1. Avance de las Pruebas 2. Eficacia de las pruebas 3. Porcentaje de Avance Preparación de Casos de Prueba 4. Porcentaje de Avance Preparación de Entorno de Prueba 5. Cantidad de casos de prueba ejecutas, con fallas 6. Densidad de defectos 7. Cantidad de defectos encontrados 8. Cobertura de Pruebas	Después un análisis exhaustivo de las métricas presentadas por cada autor, se ven muchas coincidencias, inclusive métricas que van a medir lo mismo, pero que sin embargo tienen nombre distinto. La utilización de las métricas está estrechamente relacionada a lo que se quiere medir del proceso y del producto. No existe una fórmula mágica. Se pueden usar algunas o todas las métricas. Para el estudio se puede realizar una combinación de varias de ellas y seleccionar las más adecuadas, porque el proceso de pruebas es universal, trasparente y está sujeto a estándares internacionales como la norma ISO / IEC 29119. Sin embargo, se va a optar por las métricas de (Pressman, 2010) a pesar que no hace una descripción directa de ellos, pero del análisis sobre el desarrollo del tema en su libro se rescatan dichas métricas
(Pressman, 2010)	Métricas 1. Tiempo de ejecución de las Pruebas 2. Cantidad de defectos Reportados 3. Tasa de defectos Rechazados 4. Tasa de Pruebas de Regresión 5. Tasa de Pruebas Automatizadas 6. Eficiencia en la detección de defectos.	
(Fewster & Graham, 2000)	Métricas 1. Porcentaje de detección de defectos 2. Porcentaje de corrección de defectos 3. Eficacia de la Prueba 4. Cobertura de Pruebas	
(Hayes, 1995)	Métricas 1. Medida de Progreso 2. Cobertura de Código y de Requerimientos 3. Criterios de salida. 4. Cobertura de casos de prueba 5. Ratio de defectos 6. Tasa de corrección de defectos 7. Tasa de recurrencia	

2.4. Análisis Crítico de las Bases Teóricas

En base a toda la literatura considerada en las bases teóricas y las cuales han servido para seleccionar el modelo y las dimensiones para desarrollar el estudio, se puede concluir que los términos automatización de pruebas y las pruebas de software son completamente diferentes y complementarios a la vez. Coinciden, porque ambos buscan objetivos comunes en todo el proceso de control de la calidad, que son detectar defectos/fallas, prevenir defectos/fallas, e incrementar la confiabilidad en la calidad del software. Además, ambos buscan la efectividad de las pruebas y poder cumplir con la cobertura de requisitos que se acumula en cada iteración cuando se entrega una nueva versión del producto de software dentro del proceso de desarrollo. Y se diferencian, porque cada uno tiene sus propias consideraciones en cada actividad del proceso de prueba, ambos tienen su propia planificación, diseño, implementación, y ejecución de los casos de prueba. La automatización de pruebas lo hace a través de herramientas y frameworks que permiten establecer una arquitectura para crear, diseñar y mantener scripts y las pruebas de software a través de herramientas de gestión de pruebas (defectos y evidencias) y testing manual. También, se diferencian porque ambos poseen sus propia dimensiones y métricas que fueron especificadas de manera detallada en la sección anterior.

Todos los autores coinciden en mencionar que la automatización de pruebas impacta positivamente en el proceso de pruebas de software, ayudando a este a mejorar la productividad, la eficiencia, reducir tiempo/costos y también mejorando la calidad del producto bajo pruebas. Esta hipótesis es lo que sé demostrará como parte del desarrollo del estudio.

Antes de sustentar el modelo que se va a utilizar por cada variable de estudio es importante aclarar que toda esta investigación se fundamenta en un modelo macro

de ciclo de vida de desarrollo de software. No sé ha profundizado en este tema, porque no es el objetivo del estudio, pero si se ha explicado en las bases teóricas. Además, mencionar que el Ministerio de Educación del Perú cuenta con modelos de desarrollo implementados en proceso de madurez. Utiliza el modelo V para desarrollo tradicional y SCRUM para desarrollo ágil. Este estudio se desarrollará bajo el marco de referencia de buenas prácticas y agilidad SCRUM, debido a que, bajo este modelo, la automatización de pruebas potencia la calidad del producto.

Para la variable automatización de pruebas se va utilizar el modelo que propone (Fewster & Graham, 2000), debido a que sustenta de manera más técnica la diferencia entre pruebas y automatización de pruebas, indicando para cada una de ellas sus propias dimensiones y métricas. Además, en el enfoqué diferencia de manera muy clara que la automatización de pruebas va mucho más allá del solo hecho de grabar y reproducir la prueba. Indica que este enfoque está desfasado por los costos altísimos que genera en el diseño y actualización de los casos de prueba automatizados. Las dimensiones que propone están mucho más afianzadas con el proceso de la automatización que consiste planear, diseñar, ejecutar y mantener scripts.

Por otro lado, para la variable pruebas de software se va a utilizar el modelo que propone (Pressman, 2010) tanto para las dimensiones y métricas, porque enmarca a las pruebas dentro de dos procesos macros que son el aseguramiento de la calidad y la calidad del producto de software. Además, para las métricas se va a utilizar también el modelo de (ISTQB, 2018) como complemento, porque las pruebas de software son trasparentes y está sujeto a estándares internacionales como la norma ISO / IEC 2911, que define el vocabulario, procesos, documentación, técnicas y un modelo de evaluación del proceso de pruebas de software.

CAPITULO III: MARCO REFERNENCIAL

3.1. Reseña Histórica

El Perú es un país multicultural, la historia y la realidad actual lo han demostrado. Por ello, la educación desde inicios de la época republicana hasta la actualidad ha ido adoptando diferentes enfoques y transformaciones. Después de la declaración de la independencia en 1821, por parte del General don José De San Martín, el país vivía un gran proceso de adaptación a una nueva era, y la educación no era ajena a ello, por ello:

> El 4 de febrero de 1837, durante la feroz y republicana agitación de un país que comenzaba a vivir independizado, Don Andrés de Santa Cruz, Capitán y Presidente de Bolivia, Gran Mariscal de la Conciliación del Perú, Supremo Protector de los Estados del Sur y Nora. Peruano decidió crear un ministerio llamado Ministerio de Educación Pública, Beneficencia y Asuntos Eclesiásticos.
>
> Inicialmente, el ministerio, que se llamó Educación Pública, Caridad y Asuntos Eclesiásticos, tiene tres empleados. La nómina aumentó al día siguiente de la designación del primer ministro del departamento, el Dr. Manuel Gaspar de Villarán y Loli. A mediados del siglo XIX, el General de Castilla dictó las primeras normas educativas que establecían la separación de la enseñanza pública y privada, al mismo tiempo que se declaraba al Colegio de Guadalupe universidad nacional y se establecía la facultad como institución pública. (Gobierno del Perú, 2020, pág. 2)

Décadas más tarde, en el año 1866, en el gobierno de don Mariano Ignacio Prado se regula la Educación Superior y se crean escuelas gratuitas para los domingos. En los gobiernos posteriores se inicia con la creación de escuelas gratuitas y unidades escolares en las capitales de los distritos, se instaura la educación

secundaria en 5 años, se establece el día del maestro, se fomenta la educación inicial y el desarrollo de la educación superior.

Hasta que, “En 1993, se promulga un nuevo reglamento de Organización y Funciones del Ministerio, y se hace efectiva la racionalización con el cese de trabajadores del sector, por reorganización. A partir de entonces la educación peruana ha experimentado grandes cambios y enormes progresos en cuanto a infraestructura y calidad, aunque todavía es mucho lo que queda por avanzar” (Gobierno del Perú, 2020).

A inicios del 2020, la pandemia del COVID 19, ha desnudado la fragilidad y precariedad del sistema educativo de nuestro país. El ente rector del sector educación, el Ministerio de educación del Perú (MINEDU) en los inicios se ha visto avasallado por las consecuencias que generaba la pandemia en el proceso educativo y el impacto directo en la comunidad educativa. Las clases virtuales fueron un arma muy importante al inicio. Sin embargo, no todos los estudiantes tenían acceso a una Tablet/computador o a los servicios de internet que eran necesarios para acceder a esta modalidad de estudio, sino también la débil infraestructura tecnológica del sector. La comunidad educativa de extrema pobreza y de zonas rurales fueron los más golpeados. Todo ello, ha hecho que las brechas educativas se incrementen durante la pandemia.

En la actualidad, post pandemia, el MINEDU está enfocando sus esfuerzos en incrementar la reinserción y continuidad educativa a través de actividades colaborativas con el sector privado, dirigidas a estudiantes, directivos y docentes que motiven la participación de las familias y la comunidad, así como la cooperación dinámica de las direcciones regionales de educación (DRE). Todo ello, soportado en el uso de tecnologías de la información (TIC). Las tecnologías emergentes están

haciendo que las políticas educativas puedan implementarse, garantizando un proceso educativo accesible, continuo, flexible y de calidad para el estudiante. Gracias a estas tecnologías, se puede pasar de modalidad presencial a semipresencial o virtual sin impactar el proceso de aprendizaje. El Minedu ha innovado y migrado a la nube su plataforma digital “PerúEduca” y Campus virtual docentes con actualizaciones constantes. Además, ha creado sistemas como Aula virtual docentes, Aprendo en casa, planificador de clases, y aplicativos como Aprendo en casa APP, y materiales educativos APP.

Por tanto, el Minedu como ente rector del sector educativo ha dado un giro hacia la modernidad en infraestructura tecnológica para reforzar el proceso de formación de sus estudiantes, y finalmente garantizar una verdadera inclusión y una educación colaborativa y de calidad.

3.2. Filosofía Organizacional

Todo peruano tiene derecho a una educación digna y de calidad, y éste es el principal propósito del MINEDU, y lo hace a través de las siguientes funciones:

Somos el ente conductor de la política educativa nacional y brindamos liderazgo a través de la colaboración intergubernamental y la comunicación con los gobiernos locales y regionales, promoviendo acciones de diálogo y cooperación.

Cooperamos con los municipios regionales para desarrollar, orientar, regular y evaluar la política educativa nacional. Además, hemos desarrollado una política de igualdad.

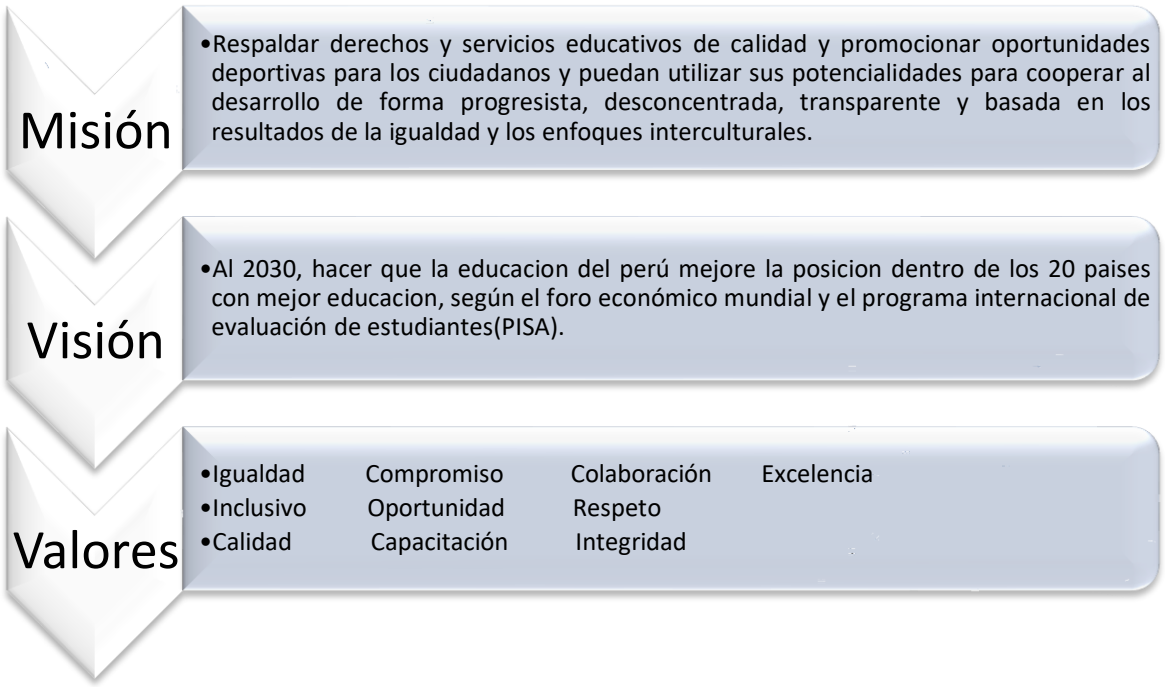
Desarrollamos marcos de niveles y modelos del sistema educativo, programas nacionales para directores, docentes y estudiantes, políticas

vinculadas con la distribución de becas y créditos educativos, y procesos de medición y evaluación de resultados de aprendizaje. (Gobierno del Perú, 2020)

El ministerio de Educación del Perú como organismo autónomo también posee su plan estratégico institucional y por lo tanto tiene su misión, visión, valores y políticas como podemos ver en la figura 11.

Figura 11

Misión, Visión y valores del Ministerio de Educación del Perú – MINEDU



El Minedu, como ente rector del sector educación tiene muchas políticas nacionales y que están en sincronía con lo establecido en el acuerdo nacional. Sin embargo, para el desarrollo del estudio no centraremos en las políticas de gobierno digital y del sistema nacional informático porque, el estudio se realizará en la unidad de calidad y seguridad de la información (UCSI) adscrito a la oficina de tecnologías de la información y comunicación (OTIC) del MINEDU. En la tabla 16, se detallan las

políticas y objetivos del plan estratégico institucional (PEI) 2019-2022 relacionados al gobierno digital.

Tabla 16

Políticas y objetivos del MINEDU con respecto al Gobierno Digital y Sistema Nacional Informático.

Acuerdo nacional		**Ley de Gobierno Digital**	**PEI 2019-2022 - MINEDU**	
Eje	**Política**	**Objetivo**	**Objetivo Estratégico**	**Acción Estratégica**
ii) Equidad y Justicia Social	12. Concesión total a una educación estatal gratis y de calidad y el fomento y protección de la cultura y el deporte	Crear un marco de gobernanza del gobierno digital para administrar adecuadamente las identidades y los servicios digitales, la arquitectura, la interoperabilidad, y la seguridad digital y de datos.	Reformar la administración y el financiamiento de las instituciones y los sistemas de educación	Implementar la gestión digital de acuerdo con la política nacional de sistemas de información del Ministerio de Educación
iv) Gobierno Eficaz, Desconcentrado y Transparente	35. Sociedad de la Información y Sociedad del conocimiento			

3.3. Diseño Organizacional

En esta sección se especifica la estructura organizacional del Ministerio de Educación y se explica de manera general los órganos que lo componen.

El MINEDU está conformado por la alta dirección, los órganos Consultivo, de Control, de Defensa Jurídica, de Asesoramiento, de Apoyo, de Línea, Descentralizados y los Programas Nacionales Adjuntos. La Alta Dirección está conformada por la oficina Ministerial, el cuerpo de asesores, la secretaria principal y los despachos de los viceministerios. El Consejo Nacional de Educación como

organismo de asesoría es independiente, y se encarga de los planes y políticas de educación a largo y mediano plazo. Los órganos de control inspeccionan y cautelan la veracidad y transparencia del uso recursos y bienes del Ente Rector. Y por otro lado se tiene a la Procuraduría como institución de defensa del ministerio. La secretaría de planificación estratégica junto a las diversas oficinas de gestión son los encargados de evaluar y ejecutar la políticas y estrategias del sector. Los órganos descentralizados son los responsables de desarrollar y ejecutar los proyectos de inversión pública (Gobierno del Perú, s.f.) .

Seguidamente, en las figuras 12, 13 y 14, se indican a detalle la estructura organizacional del Ministerio de Educación del Perú hasta el tercer nivel de expansión.

Figura 12

Estructura Organizacional del Ministerio de Educación del Perú – MINEDU – Parte 1/3

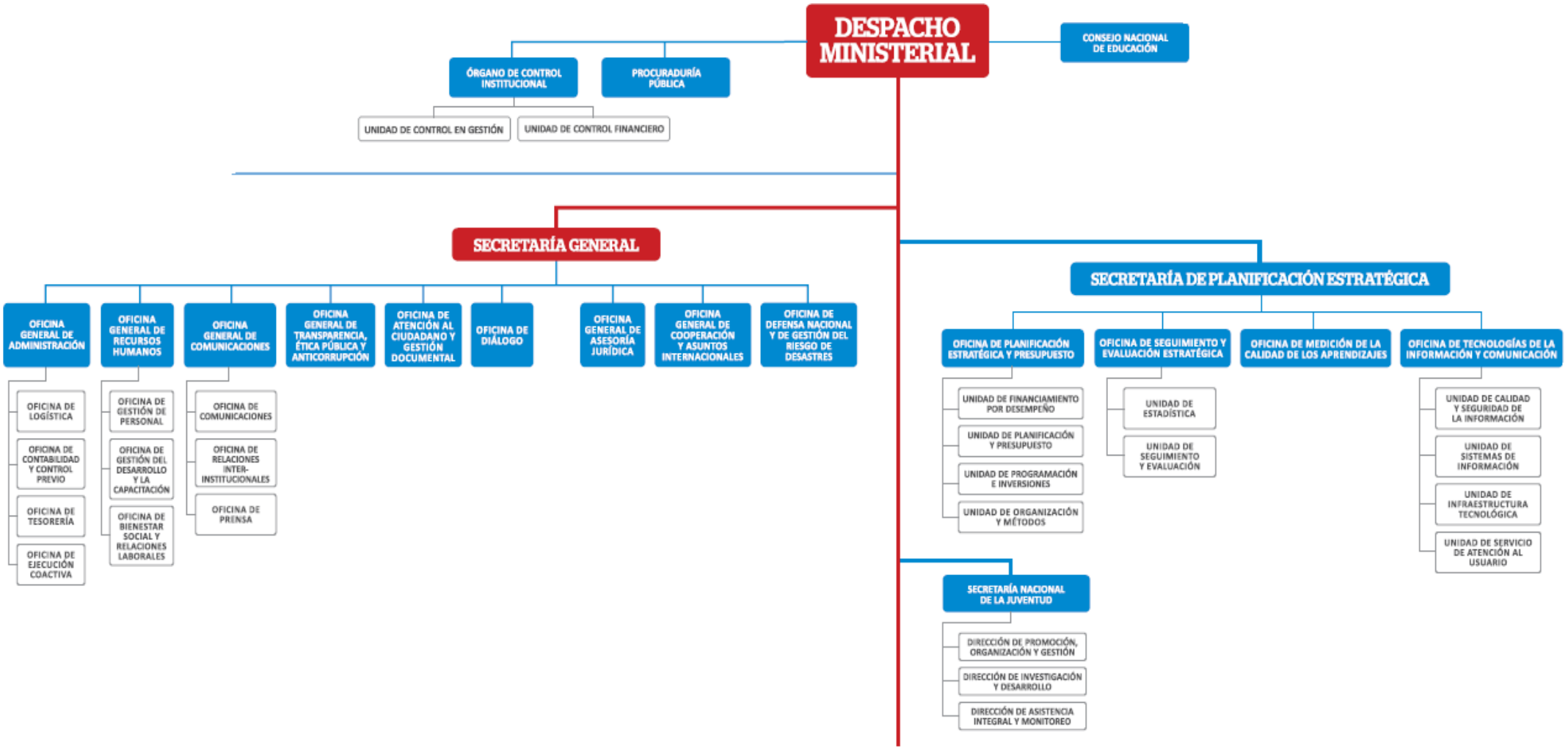


Figura 13

Estructura Organizacional del Ministerio de Educación del Perú – MINEDU – Parte 2/3

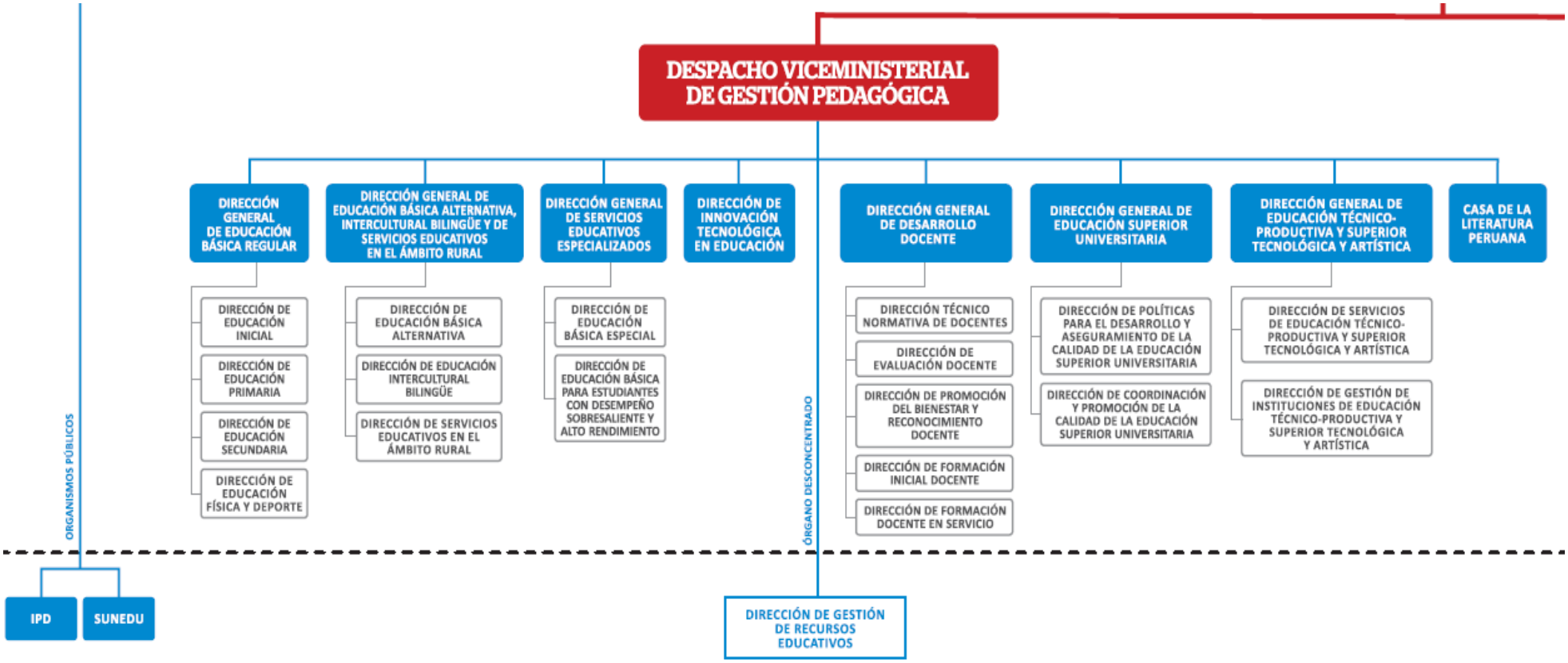
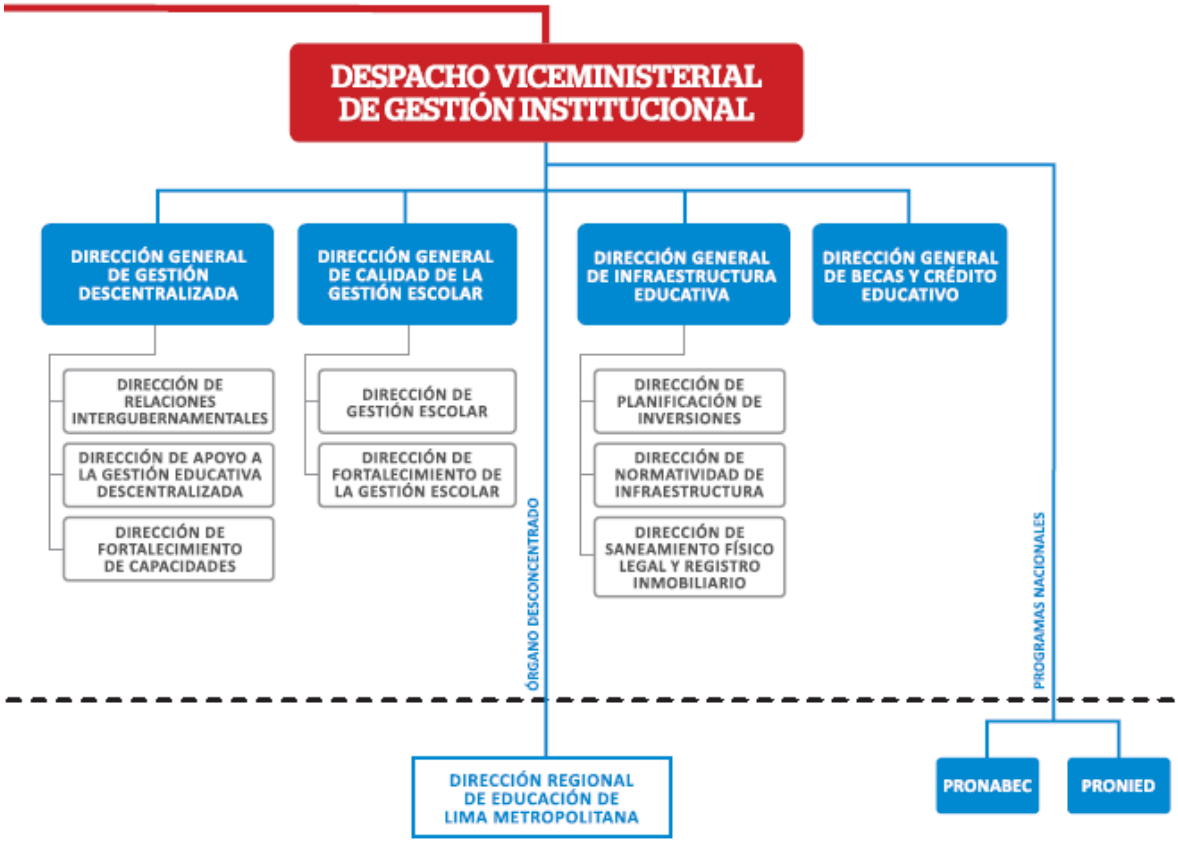


Figura 14

Estructura Organizacional del Ministerio de Educación del Perú – MINEDU – Parte



Nota. Organigrama del Ministerio de Educación del Perú, aprobado el año 2015 por decreto supremo Nro. 001-2015 – MINEDU. Tomada de *Organigrama del Ministerio de Educación del Perú* [Figura], MINEDU, 2015, Portal de Transparencia Web Site(http://www.minedu.gob.pe/p/xtras/organigrama_minedu.pdf).

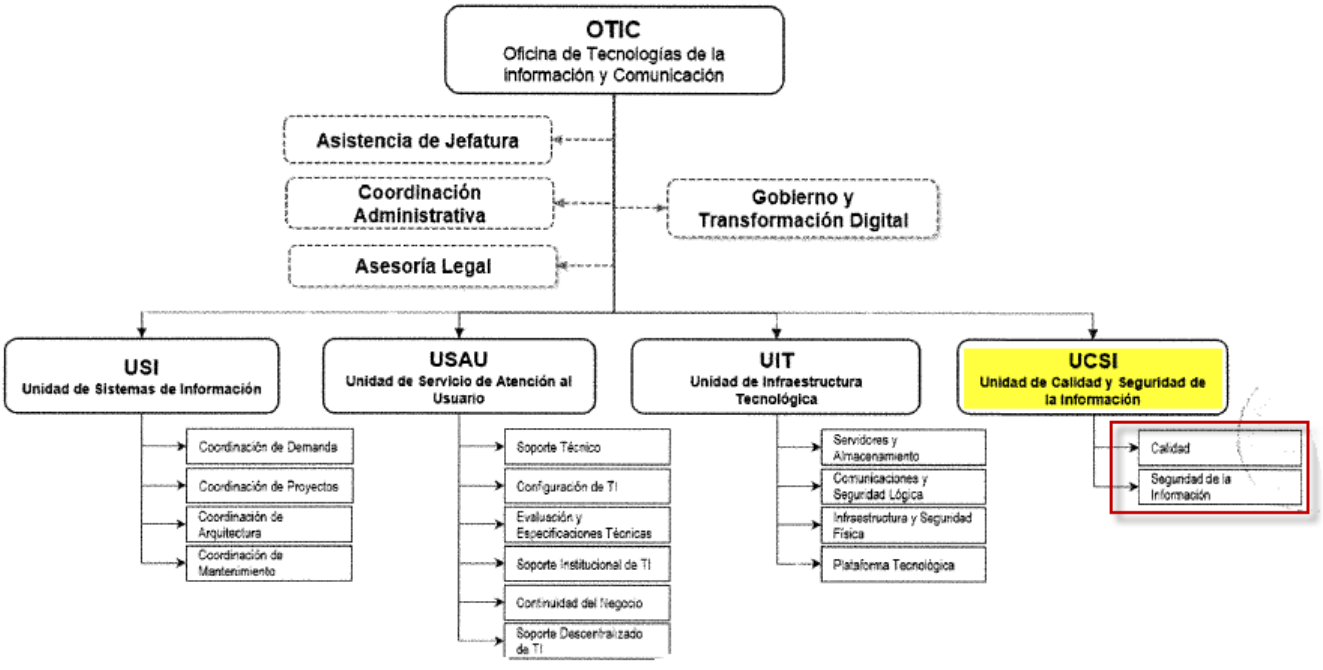
3.4. Descripción de Actores

El departamento responsable de la gestión de los recursos de TI, así como plantear planes, estándares y normas en el sector educación es la Oficina de Tecnologías de la Información y Comunicaciones – OTIC; esta oficina cuenta organizacionalmente con varias unidades orgánicas para el desempeño de sus funciones. Además, es oportuno mencionar que el estudio se realizará en la Unidad de Calidad y Seguridad de la Información – UCSI.

UCSI es el departamento de la OTIC que se encarga del proceso de aseguramiento y control de calidad de los productos de software del MINEDU, a través de pruebas funcionales y de seguridad. La unidad está conformada por 28 personas, entre especialistas y analistas en certificación o carreras afines. De los cuales 26 analistas son los encargados de realizar las pruebas de software en sus distintos niveles y tipos. Todos cuentan con las aptitudes y habilidades para realizar pruebas manuales tanto en entorno tradicional, ágil e hibrido. La clasificación y asignación de un analista para un determinado proyecto depende mucho de la madurez que tenga técnicas, tipos de prueba, lenguajes de programación, automatización y otros aspectos. En el grafico 12, se detalla la estructura orgánica de la OTIC-MINEDU:

Figura 15

Estructura Organizacional de la Unidad de Calidad y Seguridad de la Información - UCSI



.

Nota. Tomada de *Organigrama OTIC-MINEDU* [Figura], MINEDU, 2019, Portal de Transparencia Web Site (http://www.minedu.gob.pe/transparencia/2020/pdf/RM-N-620-2019-MINEDU-Aprobar_el_Plan_de_Gobierno_Digital_del_Minedu_2019-2022.pdf).

3.5. Productos y Servicios

Principalmente el MINEDU ofrece servicios educativos a toda la comunidad educativa para garantizar una enseñanza inclusiva y de calidad. Muchas veces estos servicios están soportados en infraestructura tecnológica (Hardware, software y plataformas) propietarias y a demanda. En la figura 16, se especifica los principales sistemas de información que soportan los diferentes procesos operativos del MINEDU.

Figura 16

Sistemas de información de Procesos operativos - MINEDU

Tipo de Procesos	Procesos	Sistemas de Información
Procesos Operativos	PO01Gestionar el Curriculo Nacional, normas pedagógicas y modelos educativos	- SIAGIE - SISEVE - Identicole: - Ponte en Carrera - Juegos Florales Escolares - Sistema de Apoyo al Proceso Único de Admisión a los Colegios de Alto Rendimiento – COAR - Sistema de Gestión del Plan de Fortalecimiento de Educación Física y Deporte Escolar – SISEF
	PO02 Gestionar materiales educativos	- Observatorio Nacional de Textos Escolares - SIGMA - Materiales Educativos IIEE
	PO03 Dotar y gestionar el desarrollo profesional del personal en las instituciones educativas	- NEXUS, SUP, Escalafón, SIRA, Módulo de Constancias - PerúEduca: - Plataforma Moodle para la gestión de las capacitaciones de alcance nacional para docentes y trabajadores del Sector. Cuenta con servicio de Aula Virtual.
	PO04 Gestionar la infraestructura, mobiliario y equipamiento educativo	- Sistema de Información para el Registro de Declaraciones de Gastos de Mantenimiento Preventivo de Locales Educativos - Módulo de Declaración de Gastos por mantenimiento de locales educativos. - Módulo de Declaración de Gastos por mantenimiento de locales educativos PC.
	PO05 Dotar becas y crédito	- Sistema de Becas OBEC - Sistema de Créditos OBEC
	PO06 Gestionar la calidad de las instituciones educativas	- Sistema de Monitoreo PRONOEI. - Sistema Wasichay.

Nota. Tomada de *Sistemas de Información* [Figura], MINEDU, 2019, Portal de Transparencia Web Site (http://www.minedu.gob.pe/transparencia/2020/pdf/RM-N-620-2019-MINEDU-Aprobar_el_Plan_de_Gobierno_Digital_del_Minedu_2019-2022.pdf).

En la figura 17, se detallan los principales servicios digitales que ofrece el MINEDU, a través de su portal principal.

Figura 17

Principales Servicios Digitales - MINEDU

Servicios Docentes	Servicios de Gestión Educativa
- Concurso de Buenas Prácticas Docentes - Evaluación Docente - Gracias Profe - Palmas Magisteriales - Reforma Magisterial - Rutas de Aprendizaje – Kit de Evaluación - Registro Nacional de Docentes Bilingües en Lenguas Originarias - Somos Docentes	- Colegios de Alto Rendimiento (COAR) - Compromisos de Desempeño - Identicole - Educación superior Tecnológica - PerúEduca - Ponte en carrera - Sistema Especializado en Reporte de Casos de Violencia Escolar – SISEVE. - Superatec

Nota. Tomada de *Servicios Digitales* [Figura], MINEDU, 2019, Portal de Transparencia Web Site (http://www.minedu.gob.pe/transparencia/2020/pdf/RM-N-620-2019-MINEDU-Aprobar_el_Plan_de_Gobierno_Digital_del_Minedu_2019-2022.pdf).

3.6. Diagnostico Sectorial

El Ministerio de Educación del Perú como ente rector del sector, cuenta con su Plan Estratégico Sectorial Multianual (PESEM), este nace de las políticas establecidas en el Acuerdo Nacional. A partir del PESEM se desprenden las otras herramientas estratégicas del MINEDU como son: El Plan Estratégico Institucional (PEI) y el Plan de Gobierno Digital (PGD). Todas las herramientas mencionadas cuentan con una línea de trazabilidad y los objetivos que se plantean en cada una de ellas están

alineadas a los objetivos establecidos en el documento precedente de mayor jerarquía.

Por lo tanto, para el estudio nos enfocaremos en analizar el PGD, porque se plantea demostrar la mejora del proceso de pruebas de software mediante la ejecución de la automatización de pruebas. Con ello se asegura que la presente investigación responde y está alineado a los objetivos establecidos en el Plan de Gobierno digital. Para mayor detalle de la trazabilidad de los objetivos, ver la tabla 15.

En la figura 18, se visualiza la lista de los proyectos priorizados por el Ministerio de Educación del Perú para el periodo 2019-2022.

Figura 18

Listado de Proyectos de Gobierno Digital – MINEDU – 2019-2022

Nro.	Proyecto	Descripción	Objetivo Estratégico Institucional	Objetivo de Gobierno Digital
Digitalización de Procesos y Servicios				
1	Sistema Integrado de Gestión de Servidores Docentes y Administrativos (SISDA)	Desarrollar un sistema web integrado con base de datos centralizada, que soporte los procesos atendidos actualmente por los sistemas SUP, NEXUS y LEGIX. El mismo que debe abarcar todos los regímenes laborales que se encuentran a cargo del Sector Educación.	OEI.4 OEI.6	OGD.1 OGD.2 OGD.3
2	PerúEduca 4.0	Implementar la plataforma tecnológica educativa del MINEDU que permita responder a la demanda de servicios educativos digitales de la comunidad educativa (Docentes, Directores, Estudiantes, Familia y Aliados).	OEI.1 OEI.4	OGD.1 OGD.2 OGD.3
3	Implementación de la modalidad EBA en el SIAGIE (SIAGIE-EBA)	Implementar la modalidad de la Educación Básica Alternativa (EBA) en el SIAGIE, para los procesos de matrícula y evaluación de aprendizaje de la Educación Básica Alternativa.	OEI.6	OGD.2 OGD.3
4	Sistema de Informe de Progreso del Estudiante (SIPE)	Implementar un Sistema Web Multiplataforma que genere un vínculo estrecho entre el padre de familia y/o apoderado con la Institución	OEI.6	OGD.2 OGD.3

Nota. Tomada de *Listado de Proyectos Digitales*[Figura], MINEDU, 2019, Portal de Transparencia Web Site (http://www.minedu.gob.pe/transparencia/2020/pdf/RM-N-620-2019-MINEDU-Aprobar_el_Plan_de_Gobierno_Digital_del_Minedu_2019-2022.pdf).

CAPÍTULO IV: RESULTADOS

4.1. Marco metodológico (tipo y diseño de investigación, población, muestra, instrumentos).

Para el estudio se realizó una investigación aplicada, sustentada en la aplicabilidad de conocimientos de la automatización de pruebas para optimizar las pruebas de software; el diseño es preexperimental porque se realizó medición de la variable dependiente (Pruebas de software) antes y después de la aplicación de la variable independiente (Automatización de pruebas) y cuantificar el efecto en el proceso de control de calidad. La población considerada en el estudio fueron todos los trabajadores de la Unidad de Sistemas de Información y la muestra solamente los 26 analistas de calidad de la Unidad de Calidad y seguridad de la Información. Las herramientas tecnológicas utilizadas fueron Java, Selenium WebDriver, Cucumber, Junit y Gherkin. La instalación y configuración de las herramientas para diseñar la arquitectura de automatización están detalladas en el anexo1(ver anexos) y el modelo de framework utilizado fue Page Object Model - POM cuyo detalle esta explicado en el capítulo 3 marco teórico, sección definiciones.

En los siguientes acápites se detalla los resultados obtenidos en la medición de cada indicador por cada uno de los objetivos específicos.

4.2. Resultados

A continuación, por cada objetivo específico, se detallan los resultados cuantitativos que se obtuvieron antes y después de la aplicación de la variable independiente (Automatización de Pruebas).

OE1: Determinar como la rapidez de los scripts automatizados mejora la productividad de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022.

Tabla 17

Métrica del Indicador Tiempo de Ejecución de Pruebas

Dimensión	Indicador	Formula
Productividad	Tiempo de Ejecución de Pruebas	TEPF=TEPFM-TEPFA TEPF: Tiempo de ejecución de pruebas funcionales TEPFM: tiempo de ejecución de pruebas funcionales manuales TEPFA: Tiempo de ejecución de pruebas funcionales automatizadas

En la tabla 18, se detalla los tiempos obtenidos de la ejecución de pruebas funcionales manuales(antes) y automatizados(después) y un resumen (%) del beneficio positivo de la automatización de pruebas en la productividad del proceso de certificación.

Tabla 18

Resultado del indicador Tiempo de Ejecución de Pruebas

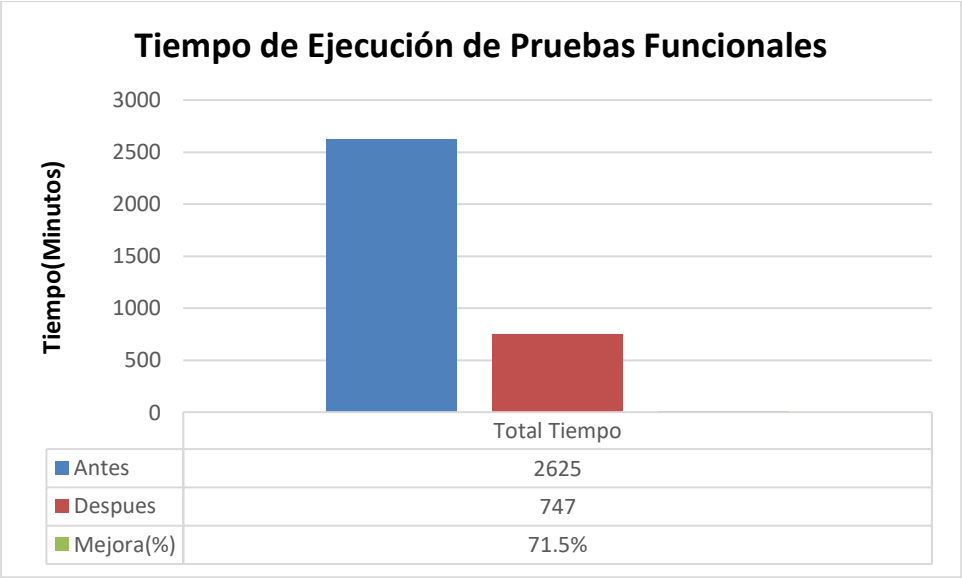
Automatización de pruebas funcionales							
Legacy	**Feature**	**Cantidad de Escenarios**	**Plataforma**	**Tiempo de ejecución de pruebas en minutos**			
				Prueba Manual	**Total, Tiempo PM**	**Prueba Automatizada**	**Total, Tiempo PA**
Portal PerúEduca v4.0	Iniciar Sesión	21	Google Chrome, Firefox y Edge	15	315	4	84
	Buscar Recurso	15		10	150	3	45
	Comentarios	9		12	108	4	36
	Analítica	24		8	192	3	72
	Registrar Usuario	18		19	342	6	108
	Streaming	27		11	297	4	108
	Registra ME	12		20	240	6	72

BackOffice PerúEduca v4.0	Registrar AR	18	Google Chrome, Firefox y Edge	25	450	5	90
	Transmisiones	12		18	216	5	60
	Colecciones	9		10	90	3	27
	Actualización de Datos	15		15	225	3	45
Total		**180**		**14.82**	**2625**	**4.81**	**747**
Cálculo de la Métrica							
TEPF=TEPFM-TEPFA **TEPF**=2625-747 **TEPF**=1878 Minutos % Reducción de tiempo en pruebas = [1- (TEPFA/TEPFM)]*100 % Reducción de tiempo en pruebas = [1- (747/2625)]*100 % Reducción de tiempo en pruebas = **71.5%**							
Análisis: Se observa que hay una mejora considerable en la reducción del tiempo de pruebas en un total del 71.5% y que impacta directamente de manera positiva en la productividad del proceso de control de pruebas.							

En la figura 19, se muestra de forma resumida el análisis de la tabla de resultados explicado en el acápite anterior.

Figura 19

Resumen Indicador "Tiempo de Ejecución de Pruebas Funcionales"



A continuación, en la figura 20 se explica los estadísticos inferenciales para validar la normalidad de los datos con respecto a este primer indicador. En el gráfico

se observa que el valor p (0.010) para el antes y el valor p (0.002) para el después son menores al criterio de significancia o error (0.05). Por consiguiente, se determina que los datos no tienen una distribución normal.

Figura 20

Prueba para determinar normalidad de los datos del indicador "Tiempo de Ejecución de Pruebas Funcionales" Pre y Post

Resumen de procesamiento de casos

	Casos					
	Válido		Perdidos		Total	
	N	Porcentaje	N	Porcentaje	N	Porcentaje
PreTest	26	100.0%	0	0.0%	26	100.0%
PosTest	26	100.0%	0	0.0%	26	100.0%

Descriptivos

			Estadístico	Error estándar
PreTest	Media		100.96	8.387
	95% de intervalo de confianza para la media	Límite inferior	83.69	
		Límite superior	118.23	
	Media recortada al 5%		97.98	
	Mediana		94.50	
	Varianza		1828.758	
	Desv. estándar		42.764	
	Mínimo		50	
	Máximo		216	
	Rango		166	
	Rango intercuartil		46	
	Asimetría		1.072	.456
	Curtosis		.686	.887

PosTest	Media		28.73	2.399
	95% de intervalo de confianza para la media	Límite inferior	23.79	
		Límite superior	33.67	
	Media recortada al 5%		27.83	
	Mediana		27.50	
	Varianza		149.645	
	Desv. estándar		12.233	
	Mínimo		15	
	Máximo		60	
	Rango		45	
	Rango intercuartil		14	
	Asimetría		1.141	.456
	Curtosis		1.144	.887

Pruebas de normalidad

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
PreTest	.204	26	.007	.891	26	.010
PosTest	.189	26	.017	.861	26	.002

a. Corrección de significación de Lilliefors

Nota. La figura representa el resultado que retorna el software *IBM SPSS Statistics Desktop*. Licencia estudiantil Gratuita por 30 días**.**

A continuación, se explica los estadísticos inferenciales para validar la hipótesis de investigación “La rapidez de los scripts automatizados mejora la productividad de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú” mediante el test no paramétrico de Wilcoxon, donde la hipótesis nula (**Ho**) es la no mejora de la productividad y la hipótesis alterna es la mejora de la productividad(**H_1**).

En la figura 21, se observa los resultados del test no paramétrico de Wilcoxon para el contraste de las hipótesis. Se visualiza que la significación (p) es “0.001”; y que es absolutamente inferior que el valor de alfa (α) de 0,05. Además, el resultado de “Z” muestra un valor de “-4.464”. De modo que se rechaza la hipótesis nula y se reconoce la hipótesis alterna con un nivel de confianza del 95%.

Figura 21

Test no paramétrico Wilcoxon del indicador "Tiempo de Ejecución de Pruebas Funcionales" Pre y Post

Prueba de rangos con signo de Wilcoxon

Rangos

		N	Rango promedio	Suma de rangos
PosTest - PreTest	Rangos negativos	26[a]	13.50	351.00
	Rangos positivos	0[b]	.00	.00
	Empates	0[c]		
	Total	26		

a. PosTest < PreTest

b. PosTest > PreTest

c. PosTest = PreTest

Estadísticos de prueba[a]

	PosTest - PreTest
Z	-4.464[b]
Sig. asin. (bilateral)	<.001

a. Prueba de rangos con signo de Wilcoxon

b. Se basa en rangos positivos.

Nota. La figura representa el resultado que retorna el software *IBM SPSS Statistics Desktop*. Licencia estudiantil Gratuito limitado.

OE2: Determinar como la fiabilidad de los scripts automatizados mejora la detección temprana de defectos de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022.

Tabla 19

Métrica del Indicador Cantidad de Defectos Reportados

Dimensión	Indicador	Formula
Detección Temprana de Defectos	Cantidad de Defectos Reportados	TB=TBPM-TBPA TB: Total de Bugs TBPM: Total Bugs con pruebas manuales TBPA: Total de Bugs con pruebas automatizadas

En la tabla número 20, se detalla la cantidad de defectos encontrados con pruebas funcionales manuales(antes) y automatizados(después) y un resumen (%) del beneficio positivo de la automatización de pruebas en la detección temprana de defectos en el proceso de certificación.

Tabla 20

Resultado del indicador "Cantidad de Defectos Reportados"

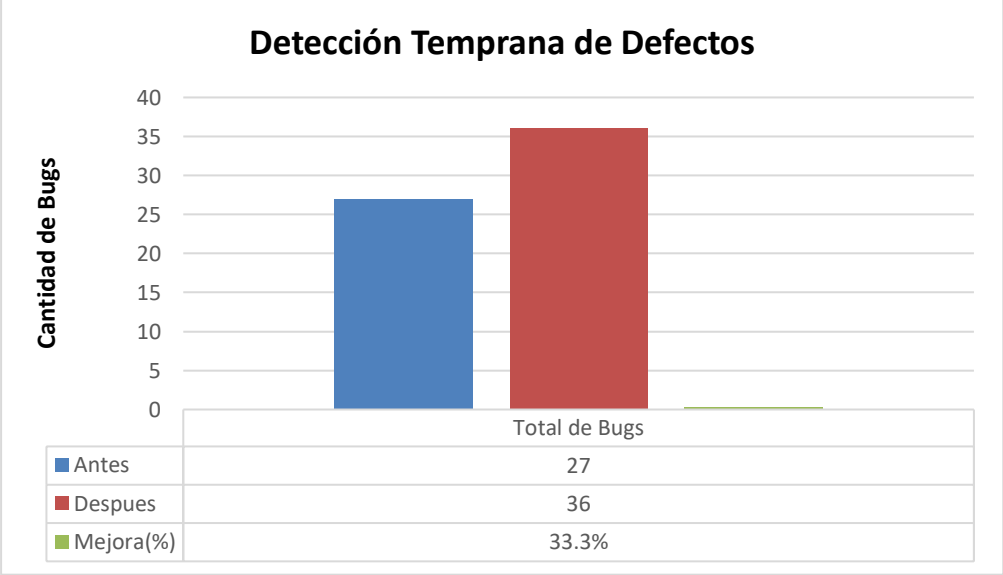
Proceso de detección de Pruebas	Bugs pruebas de Verificación	Bugs en Pruebas de Regresión	Total
Pruebas Manuales	16	11	27
Pruebas Automatizadas	21	15	36
TB= TBPA - TBPM TB=36-27=9 % de incremento de detección temprana de defectos= [(TBPA/TBPM) -1]*100 % de incremento de detección temprana de defectos= [(36/27) - 1]*100 % de incremento de detección temprana de defectos= 33%			

Análisis: Se observa que hay una mejora considerable en la cantidad de defectos detectados de manera temprana (33% más de efectividad) y que impacta de manera positiva en la resolución de los mismos y en reducir los costos de defectos del proceso de control de pruebas.

En la figura 22, se muestra de forma resumida el análisis de la tabla de resultados explicado en el acápite anterior.

Figura 22

Resumen Indicador "Cantidad de Defectos Reportados"



A continuación, en la figura 23 se explica los estadísticos inferenciales para validar la normalidad de los datos con respecto a este segundo indicador. En el gráfico se observa que el valor p (0.002) para el antes y el valor p (0.001) para el después son menores al criterio de significancia o error (0.05). Por consiguiente, se determina que los datos no tienen una distribución normal.

Figura 23

Prueba para determinar normalidad de los datos del indicador "Cantidad de Defectos Reportados"

Resumen de procesamiento de casos

	Casos					
	Válido		Perdidos		Total	
	N	Porcentaje	N	Porcentaje	N	Porcentaje
PreTest	26	96.3%	1	3.7%	27	100.0%
PosTest	26	96.3%	1	3.7%	27	100.0%

Descriptivos

			Estadístico	Error estándar
PreTest	Media		1.04	.180
	95% de intervalo de confianza para la media	Límite inferior	.67	
		Límite superior	1.41	
	Media recortada al 5%		.99	
	Mediana		1.00	
	Varianza		.838	
	Desv. estándar		.916	
	Mínimo		0	
	Máximo		3	
	Rango		3	
	Rango intercuartil		2	
	Asimetría		.597	.456
	Curtosis		-.272	.887
PosTest	Media		1.38	.201
	95% de intervalo de confianza para la media	Límite inferior	.97	
		Límite superior	1.80	
	Media recortada al 5%		1.29	
	Mediana		1.00	
	Varianza		1.046	
	Desv. estándar		1.023	
	Mínimo		0	
	Máximo		5	
	Rango		5	
	Rango intercuartil		1	
	Asimetría		1.796	.456
	Curtosis		5.434	.887

Pruebas de normalidad

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
PreTest	.248	26	<.001	.853	26	.002
PosTest	.300	26	<.001	.775	26	<.001

a. Corrección de significación de Lilliefors

Nota. La figura representa el resultado que retorna el software *IBM SPSS Statistics Desktop*. Licencia estudiantil Gratuita limitada.

A continuación, se explica los estadísticos inferenciales para validar la hipótesis de investigación “La fiabilidad de los scripts automatizados mejora la detección temprana de defectos de software en el proceso de control de calidad del Ministerio de Educación del Perú” mediante el test no paramétrico de Wilcoxon, donde la hipótesis nula (**Ho**) es la no mejora de la detección temprana de defectos y la hipótesis alterna es la mejora de la de la detección temprana de defectos (**H_1**).

En la figura 24, se observa los resultados del test no paramétrico de Wilcoxon para el contraste de las hipótesis. Se visualiza que la significación (p) es “0.029”; y que es absolutamente inferior que el valor de alfa (α) de 0,05. Además, el resultado de “Z” muestra un valor de “-2.189”. De modo que se rechaza la hipótesis nula y se reconoce la hipótesis alterna con un nivel de confianza del 95%.

Figura 24

Test no paramétrico Wilcoxon del indicador "Cantidad de defectos Reportados" Pre y Post

Prueba de rangos con signo de Wilcoxon

Rangos

		N	Rango promedio	Suma de rangos
PosTest - PreTest	Rangos negativos	6[a]	7.00	42.00
	Rangos positivos	13[b]	11.38	148.00
	Empates	7[c]		
	Total	26		

a. PosTest < PreTest

b. PosTest > PreTest

c. PosTest = PreTest

Estadísticos de prueba[a]

	PosTest - PreTest
Z	-2.189[b]
Sig. asin. (bilateral)	.029

a. Prueba de rangos con signo de Wilcoxon

b. Se basa en rangos negativos.

Nota. La figura representa el resultado que retorna el software *IBM SPSS Statistics Desktop*. Licencia estudiantil Gratuito limitado.

OE3: Determinar como la repetición de los scripts automatizados mejora la cobertura de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022.

Tabla 21

Métrica de la Dimensión Cobertura de Pruebas

Dimensión	Indicador	Formula
Cobertura de Pruebas	Cantidad de Pruebas de Regresión Ejecutadas	TPR=TPRA-TPRM TPR: Total de pruebas de regresión TPRM: Total pruebas de regresión manuales TPRA: Total pruebas de regresión automatizadas

En la tabla número 22, se detalla porcentaje de pruebas de regresión cubiertos con pruebas funcionales manuales(antes) y automatizados(después) y un resumen (%) del beneficio positivo de la automatización de pruebas en la cobertura de pruebas en el proceso de certificación.

Tabla 22

Resultado del indicador Cobertura de Pruebas de Regresión

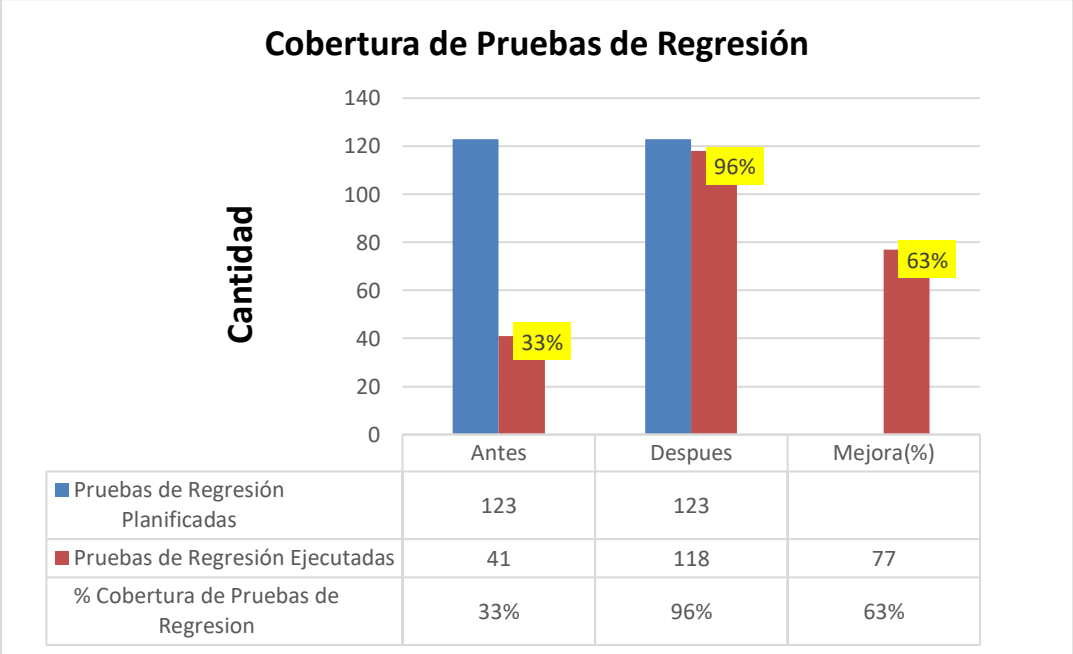
Legacy	Pruebas de Regresión Planificadas (PRP)	Cantidad de Pruebas de Regresión Manual – Ciclo 2	Cantidad de Pruebas de Regresión Automatizadas - Ciclo 3
Portal PerúEduca V4.0	68	23	68
BackOffice PerúEduca 4.0	55	18	50
Total	**123**	**41**	**118**

TPR=TPRA-TPRM
TPR= 118 – 41 = 77
% de cobertura de Pruebas Antes = [(TPRM/TPRA)]*100
% de cobertura de Pruebas Antes = [41/123]*100
% de cobertura de Pruebas Antes = 33%
% de cobertura de Pruebas Después = [(TPRM/TPRA)]*100
% de cobertura de Pruebas Después = [118/123]*100
% de cobertura de Pruebas Después = 96%
Análisis: Se observa que hay una mejora considerable en la cantidad de pruebas de regresión ejecutados (77 casos de pruebas adicionales) y que impacta de manera positiva en el incremento del porcentaje de cobertura de pruebas en un 63% adicional en el proceso de control de calidad

En la figura 25, se muestra de forma resumida el análisis de la tabla de resultados explicado en el acápite anterior.

Figura 25

Resumen indicador "Cantidad de Pruebas de Regresión Ejecutados"



A continuación, en la figura 26 se explica los estadísticos inferenciales para validar la normalidad de los datos con respecto a este segundo indicador. En el gráfico

se observa que el valor p (0.013) para el antes es menor al criterio de significancia o error (0.05) y por lo tanto estos datos no presenta una distribución normal y el valor p (0.162) para el después es mayor al criterio de significancia o error (0.05) y esto determina la normalidad de los datos. Sin embargo, ya que son muestras relacionadas, una de ellas no presenta normalidad y el tamaño de la muestra es <30 se utilizará el test con signo de Wilcoxon.

Figura 26

Prueba para determinar normalidad de los datos del indicador "Cantidad de Pruebas de Regresión Ejecutadas"

Resumen de procesamiento de casos

	Casos					
	Válido		Perdidos		Total	
	N	Porcentaje	N	Porcentaje	N	Porcentaje
PreTest	26	100.0%	0	0.0%	26	100.0%
PosTest	26	100.0%	0	0.0%	26	100.0%

Descriptivos

			Estadístico	Error estándar
PreTest	Media		1.58	.273
	95% de intervalo de confianza para la media	Límite inferior	1.02	
		Límite superior	2.14	
	Media recortada al 5%		1.49	
	Mediana		1.00	
	Varianza		1.934	
	Desv. estándar		1.391	
	Mínimo		0	
	Máximo		5	
	Rango		5	
	Rango intercuartil		3	
	Asimetría		.650	.456
	Curtosis		-.180	.887
PosTest	Media		4.54	.237
	95% de intervalo de confianza para la media	Límite inferior	4.05	
		Límite superior	5.03	
	Media recortada al 5%		4.54	
	Mediana		4.50	
	Varianza		1.458	
	Desv. estándar		1.208	
	Mínimo		2	
	Máximo		7	
	Rango		5	
	Rango intercuartil		1	
	Asimetría		-.024	.456
	Curtosis		-.421	.887

Pruebas de normalidad

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
PreTest	.199	26	.009	.897	26	.013
PosTest	.172	26	.046	.943	26	.162

a. Corrección de significación de Lilliefors

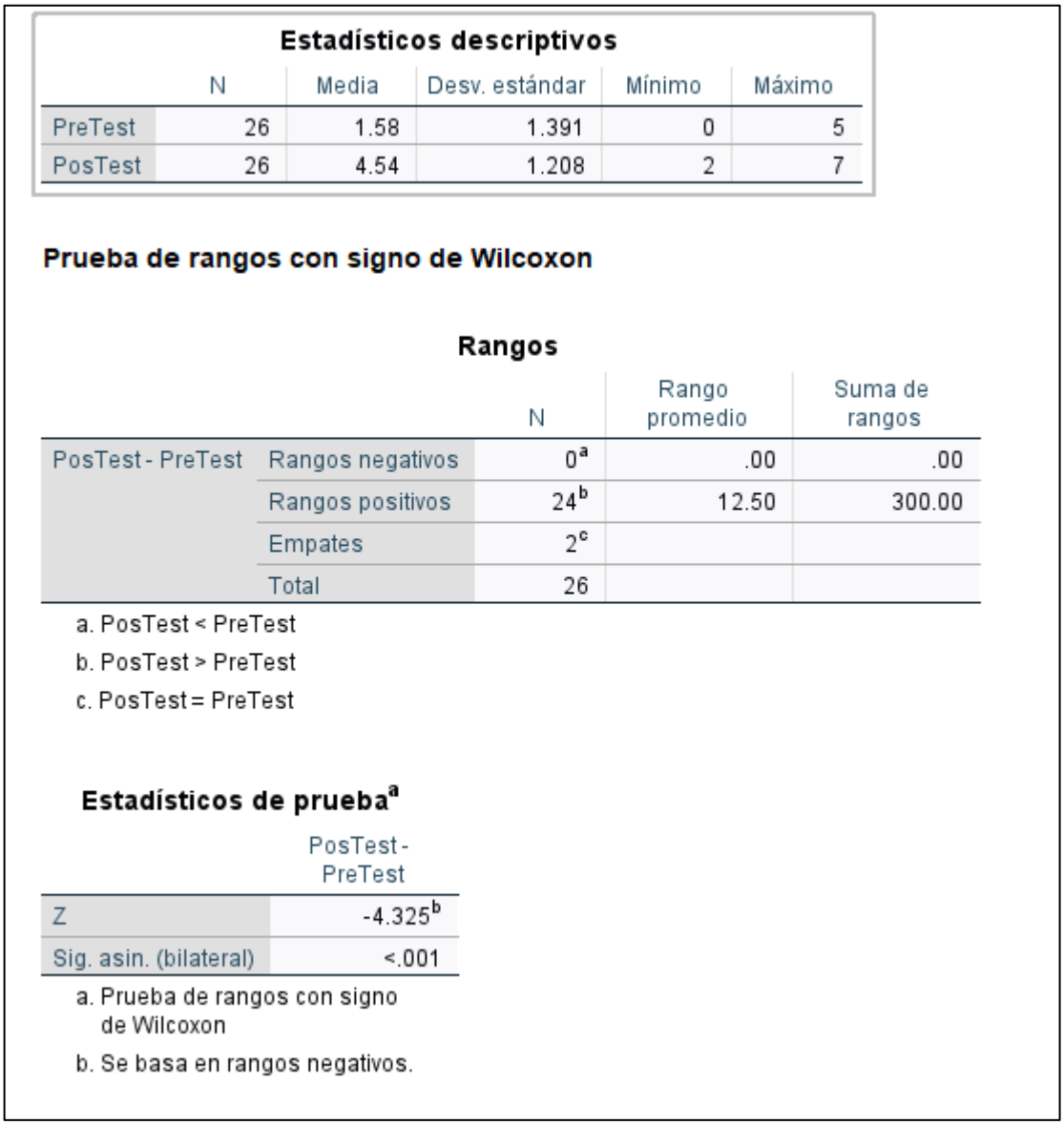
Nota. La figura representa el resultado que retorna el software *IBM SPSS Statistics Desktop*. Licencia estudiantil Gratuita limitada.

A continuación, se explica los estadísticos inferenciales para validar la hipótesis de investigación “La repetición de los scripts automatizados mejora la cobertura de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú” mediante el test no paramétrico de Wilcoxon, donde la hipótesis nula (**Ho**) es la no mejora de la cobertura de pruebas y la hipótesis alterna es la mejora de la cobertura de pruebas (**H_1**).

En la figura 27, se observa los resultados del test no paramétrico de Wilcoxon para el contraste de las hipótesis. Se visualiza que la significación (p) es “0.001”; y que es absolutamente inferior que el valor de alfa (α) de 0,05. Además, el resultado de “Z” muestra un valor de “-4.325”. De modo que se rechaza la hipótesis nula y se reconoce la hipótesis alterna con un nivel de confianza del 95%.

Figura 27

Test no paramétrico Wilcoxon del indicador "Cantidad de Pruebas de Pruebas de Regresión ejecutadas" Pre y Post



Estadísticos descriptivos

	N	Media	Desv. estándar	Mínimo	Máximo
PreTest	26	1.58	1.391	0	5
PosTest	26	4.54	1.208	2	7

Prueba de rangos con signo de Wilcoxon

Rangos

		N	Rango promedio	Suma de rangos
PosTest - PreTest	Rangos negativos	0[a]	.00	.00
	Rangos positivos	24[b]	12.50	300.00
	Empates	2[c]		
	Total	26		

a. PosTest < PreTest

b. PosTest > PreTest

c. PosTest = PreTest

Estadísticos de prueba[a]

	PosTest - PreTest
Z	-4.325[b]
Sig. asin. (bilateral)	<.001

a. Prueba de rangos con signo de Wilcoxon

b. Se basa en rangos negativos.

Nota. La figura representa el resultado que retorna el software *IBM SPSS Statistics Desktop*. Licencia estudiantil Gratuito limitado.

OE4: Determinar como la reusabilidad de los scripts automatizados mejora el rendimiento del diseño de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022.

Tabla 23

Métrica del Indicador Diseño de Pruebas Automatizados

Dimensión	Indicador	Formula
Diseño de Pruebas	Tasa de Diseño Casos de Prueba Automatizados	TDCPA=(TCPAR/TCPAI) *100 TDCPA= Tasa (%) de diseño de casos de prueba automatizados TCPAI= Total de casos de prueba automatizados iniciales TCPAR= Total de casos de prueba diseñados con reutilización

En la tabla número 24, se detalla el porcentaje de los casos de prueba automatizados gracias a la reutilización del código de los scripts en el proceso de certificación.

Tabla 24

Resultado del indicador "Tasa de diseño de casos de prueba automatizados"

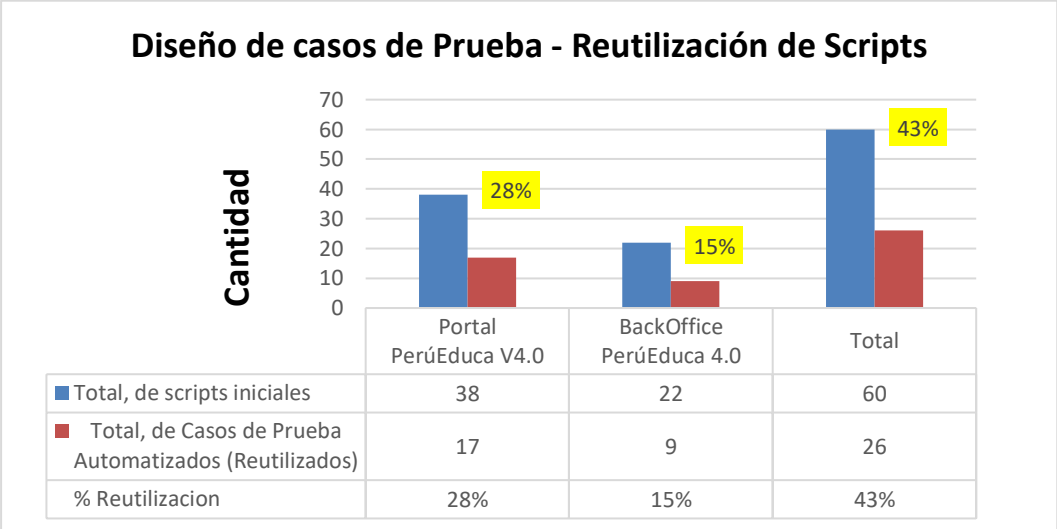
Legacy	Total, de scripts iniciales	Total, de Casos de Prueba Automatizados (Reutilizados)
Portal PerúEduca V4.0	38	17
BackOffice PerúEduca 4.0	22	9
Total	**60**	26
TDCPA = (TCPAR/TCPAI) *100 TDCPA = (26/60) *100 TDCPA = 43%		

Análisis: Se observa que hay un 43% más de casos de pruebas automatizados diseñados y que benefician en aspectos como la mejora en la productividad y el incremento en la cobertura de pruebas en el proceso de control de calidad gracias a la reutilización de los scripts.

En la figura 28, se muestra de forma resumida el análisis de la tabla de resultados explicado en el acápite anterior.

Figura 28

Resumen Indicador "Tasa de diseño de casos de prueba automatizados"



A continuación, en la figura 29 se explica los estadísticos inferenciales para validar la normalidad de los datos con respecto a este cuarto indicador. En el gráfico se observa que el valor p (0.032) para el antes y el valor p (0.019) para el después son menores al criterio de significancia o error (0.05). Por consiguiente, se determina que los datos no tienen una distribución normal.

Figura 29

Prueba para determinar la normalidad de los datos del indicador "Tasa de Diseño de Casos de prueba automatizados"

Resumen de procesamiento de casos

	Casos					
	Válido		Perdidos		Total	
	N	Porcentaje	N	Porcentaje	N	Porcentaje
PreTest	26	100.0%	0	0.0%	26	100.0%
PosTest	26	100.0%	0	0.0%	26	100.0%

Descriptivos

			Estadístico	Error estándar
PreTest	Media		6.92	.411
	95% de intervalo de confianza para la media	Límite inferior	6.08	
		Límite superior	7.77	
	Media recortada al 5%		6.84	
	Mediana		7.00	
	Varianza		4.394	
	Desv. estándar		2.096	
	Mínimo		4	
	Máximo		12	
	Rango		8	
	Rango intercuartil		4	
	Asimetría		.364	.456
	Curtosis		-.493	.887
PosTest	Media		3.00	.396
	95% de intervalo de confianza para la media	Límite inferior	2.18	
		Límite superior	3.82	
	Media recortada al 5%		2.93	
	Mediana		2.50	
	Varianza		4.080	
	Desv. estándar		2.020	
	Mínimo		0	
	Máximo		7	
	Rango		7	
	Rango intercuartil		3	
	Asimetría		.726	.456
	Curtosis		-.335	.887

Pruebas de normalidad

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
PreTest	.167	26	.061	.914	26	.032
PosTest	.190	26	.017	.904	26	.019

a. Corrección de significación de Lilliefors

Nota. La figura representa el resultado que retorna el software *IBM SPSS Statistics Desktop*. Licencia estudiantil Gratuita limitada.

A continuación, se explica los estadísticos inferenciales para validar la hipótesis de investigación "*La reusabilidad de los scripts automatizados mejora el rendimiento del diseño de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú*" mediante el test no paramétrico de Wilcoxon, donde la hipótesis nula (**Ho**) es la no mejora del rendimiento del diseño de las pruebas de software y la hipótesis alterna es la mejora del rendimiento del diseño de las pruebas de software (**H_1**).

En la figura 30, se observa los resultados del test no paramétrico de Wilcoxon para el contraste de las hipótesis. Se visualiza que la significación (p) es "0.001"; y que es absolutamente inferior que el valor de alfa (α) de 0,05. Además, el resultado de "Z" muestra un valor de "-4.231". De modo que se rechaza la hipótesis nula y se reconoce la hipótesis alterna con un nivel de confianza del 95%.

Figura 30

Test no paramétrico Wilcoxon del indicador "Tasa de Diseño de Casos de prueba automatizados" Pre y Post

Estadísticos descriptivos

	N	Media	Desv. estándar	Mínimo	Máximo
PreTest	26	6.92	2.096	4	12
PosTest	26	3.00	2.020	0	7

Prueba de rangos con signo de Wilcoxon

Rangos

		N	Rango promedio	Suma de rangos
PosTest - PreTest	Rangos negativos	23[a]	12.00	276.00
	Rangos positivos	0[b]	.00	.00
	Empates	3[c]		
	Total	26		

a. PosTest < PreTest

b. PosTest > PreTest

c. PosTest = PreTest

Estadísticos de prueba[a]

	PosTest - PreTest
Z	-4.231[b]
Sig. asin. (bilateral)	<.001

a. Prueba de rangos con signo de Wilcoxon

b. Se basa en rangos positivos.

Nota. La figura representa el resultado que retorna el software *IBM SPSS Statistics Desktop*. Licencia estudiantil Gratuita limitada.

OE5: Determinar como la programación de los scripts automatizados mejora la eficiencia de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022.

Tabla 25

Métrica del Indicador Eficiencia en la Detección del Defecto

Dimensión	Indicador	Formula
Eficiencia	Eficiencia en la Detección del defecto	EDD=(DS/DS+DP) *100 EDD: Eficiencia en la detección del defecto DS: Defectos detectados en pruebas del sistema DP: Defectos detectados en Producción

En la tabla número 26, se detalla la cantidad de defectos ocultos, que terminan en fallas en producción y que son atendidos mediante tickets.

Tabla 26

Resultado del Indicador Eficiencia en la Detección del Defecto

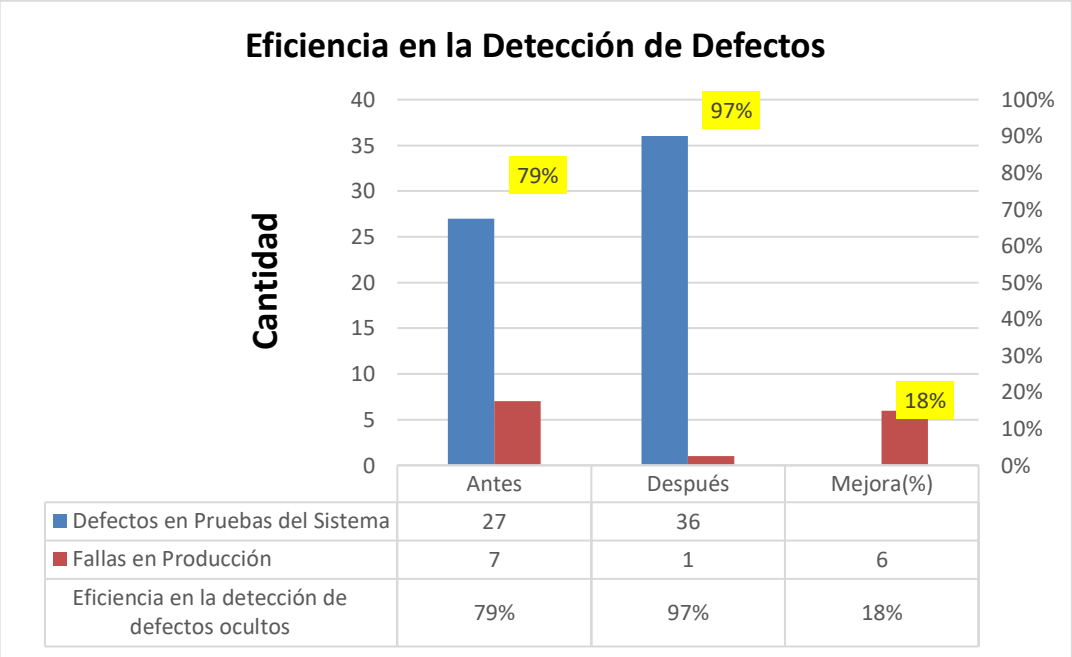
Tipo de Pruebas	Defectos en Pruebas de Sistemas	Fallas en Producción	Total
Pruebas Manuales	27	7	34
Pruebas Automatizadas	36	1	37
Total	**63**	7	
Eficiencia Antes EDD = (DS/DS+DP) *100 EDD = (27/27+7) *100 EDD = 79% ---------- Eficiencia Después EDD = (DS/DS+DP) *100 EDD = (36/36+1) *100 EDD = 97%			

Análisis: Se observa que hay un incremento en la eficiencia en la detección de defectos de un 18% adicional gracias a las pruebas automatizadas; y que incrementan la confiabilidad del producto a prueba en un 97% en el proceso de control de calidad.

En la figura 31, se muestra de forma resumida el análisis de la tabla de resultados explicado en el acápite anterior.

Figura 31

Resumen indicador "Eficiencia en la Detección de Defectos"



	Antes	Después	Mejora(%)
Defectos en Pruebas del Sistema	27	36	
Fallas en Producción	7	1	6
Eficiencia en la detección de defectos ocultos	79%	97%	18%

A continuación, en la figura 32 se explica los estadísticos inferenciales para validar la normalidad de los datos con respecto a este segundo indicador. En el gráfico se observa que el valor p (0.001) para el antes y el valor p (0.001) para el después son menores al criterio de significancia o error (0.05). Por consiguiente, se determina que los datos no tienen una distribución normal.

Figura 32

Prueba para determinar normalidad de los datos del indicador "Eficiencia en la Detección del Defecto"

Resumen de procesamiento de casos

	Casos					
	Válido		Perdidos		Total	
	N	Porcentaje	N	Porcentaje	N	Porcentaje
PreTest	26	100.0%	0	0.0%	26	100.0%
PosTest	26	100.0%	0	0.0%	26	100.0%

Descriptivos

			Estadístico	Error estándar
PreTest	Media		.27	.089
	95% de intervalo de confianza para la media	Límite inferior	.09	
		Límite superior	.45	
	Media recortada al 5%		.24	
	Mediana		.00	
	Varianza		.205	
	Desv. estándar		.452	
	Mínimo		0	
	Máximo		1	
	Rango		1	
	Rango intercuartil		1	
	Asimetría		1.105	.456
	Curtosis		-.850	.887
PosTest	Media		.04	.038
	95% de intervalo de confianza para la media	Límite inferior	-.04	
		Límite superior	.12	
	Media recortada al 5%		.00	
	Mediana		.00	
	Varianza		.038	
	Desv. estándar		.196	
	Mínimo		0	
	Máximo		1	
	Rango		1	
	Rango intercuartil		0	
	Asimetría		5.099	.456
	Curtosis		26.000	.887

Pruebas de normalidad

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
PreTest	.455	26	<.001	.557	26	<.001
PosTest	.539	26	<.001	.198	26	<.001

a. Corrección de significación de Lilliefors

Nota. La figura representa el resultado que retorna el software *IBM SPSS Statistics Desktop*. Licencia estudiantil Gratuita limitada.

A continuación, se explica los estadísticos inferenciales para validar la hipótesis de investigación “La programación de los scripts automatizados mejora la eficiencia de las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú” mediante el test no paramétrico de Wilcoxon, donde la hipótesis nula (**Ho**) es la no mejora de la eficiencia de las pruebas y la hipótesis alterna es la mejora de la eficiencia de las pruebas (**H_1**).

En la figura 33, se observa los resultados del test no paramétrico de Wilcoxon para el contraste de las hipótesis. Se visualiza que la significación (p) es “0.001”; y que es absolutamente inferior que el valor de alfa (α) de 0,05. Además, el resultado de “Z” muestra un valor de “-2.325”. De modo que se rechaza la hipótesis nula y se reconoce la hipótesis alterna con un nivel de confianza del 95%.

Figura 33

Test no paramétrico Wilcoxon del indicador "Eficiencia en la detección del Defecto"

Estadísticos descriptivos

	N	Media	Desv. estándar	Mínimo	Máximo
PreTest	26	.27	.452	0	1
PosTest	26	.04	.196	0	1

Prueba de rangos con signo de Wilcoxon

Rangos

		N	Rango promedio	Suma de rangos
PosTest - PreTest	Rangos negativos	7[a]	4.50	31.50
	Rangos positivos	1[b]	4.50	4.50
	Empates	18[c]		
	Total	26		

a. PosTest < PreTest

b. PosTest > PreTest

c. PosTest = PreTest

Estadísticos de prueba[a]

	PosTest - PreTest
Z	-2.121[b]
Sig. asin. (bilateral)	.034

a. Prueba de rangos con signo de Wilcoxon

b. Se basa en rangos positivos.

Nota. La figura representa el resultado que retorna el software *IBM SPSS Statistics Desktop*. Licencia estudiantil Gratuito limitado.

CAPÍTULO V: CONCLUSIONES Y RECOMENDACIONES O SUGERENCIAS

5.1. Conclusiones

Gracias a la automatización de las pruebas funcionales se logró mejorar de manera considerable las pruebas de software en el proceso de control de calidad del Ministerio de Educación del Perú en el año 2022.

Las pruebas de calidad han tenido una mejora neta del 71.5% en su productividad con respecto a los tiempos, gracias a la rapidez de ejecución de los scripts automatizados se permitió reducir los tiempos de ejecución de casos de prueba. Antes se tenía un tiempo promedio de ejecución de casos de prueba de 14.82 minutos; y ahora el tiempo promedio es de 4.81 minutos.

De manera considerable se logró incrementar en 33% la detección temprana de defectos, gracias a la fiabilidad de los scripts automatizados disminuyendo los errores en el proceso de control de calidad. Con pruebas manuales se detectaron 27 defectos y con pruebas automatizadas 36 defectos.

Se logró incrementar al 96% la tasa de cobertura de pruebas de regresión gracias a la repetibilidad de los scripts automatizados, logrando una mejora adicional del 63% de cobertura con la automatización de pruebas. Antes se tenía una cobertura del 33%.

Se logró mejorar el rendimiento en el diseño de casos de prueba automatizados en 43%, gracias a la reusabilidad de los scripts automatizados que permiten reutilizar bloques de código para construir nuevos casos de prueba disminuyendo el tiempo empleado en el diseño los scripts de prueba.

Se logró mejorar la eficiencia en la detección de defectos al 97%, logrando un incremento del 18% gracias a la programabilidad de los scripts automatizados. Antes se tenía una tasa de eficiencia en la detección de defectos de 79% y con las pruebas

automatizadas se tiene una tasa del 97% lo que permitió aumentar la confiabilidad en el producto y reducir los costos de fallas.

5.2. Recomendaciones o sugerencias

A continuación, se mencionan recomendaciones importantes relacionados al marco teórico del estudio y a los resultados obtenidos:

La automatización de pruebas funcionales debería de implementarse después de que la institución madure su marco de trabajo para la certificación de software. Es lo que recomiendan los expertos citados en este estudio.

Implementar la cultura DevOps en toda la institución para mejorar la capacidad de respuesta y generar mayor valor institucional reduciendo los tiempos de respuesta y la puesta en producción de los productos de software a través de la automatización de pruebas de calidad.

Extender las pruebas automatizadas como una tarea clave en cada uno de los grupos de pipelines (procesos automatizados) de los proyectos de PerúEduca con la finalidad de explotar el máximo potencial del despliegue continuo que ayude a mejorar el flujo de trabajo y a obtener un software más confiable y seguro. Todo ello debe especificarse como un estándar en el proceso de construcción de software y direccionarse a través de la Oficina de Tecnologías de la Información y Comunicación.

Incorporar al patrón de automatización de pruebas explicado en esta investigación, librerías de código abierto adicionales que ayuden a fortalecer la arquitectura de pruebas automatizadas con la finalidad de mejorar la mantenibilidad, escalabilidad, reusabilidad, y rapidez de los scripts para reducir los tiempos y costos de diseño e incrementar la eficiencia del proceso de control de calidad.

Bibliografía

Borio, J. I., & Paterno, R. J. (2021). *Automatización de pruebas de regresión [Tesis de grado, Universidad Nacional de la Plata].* Repositorio Institucional de la UNLP, De la plata. http://sedici.unlp.edu.ar/handle/10915/119477

Cabrera Serna, J., & Pareja Verastegui, C. E. (2021). *Automatización de Pruebas Funcionales Web para mejorar el Área de Calidad de Software en una empresa del rubro de Retails en el año 2021 [Tesis de Grado, Universidad Tecnológica del Perú].* UTP-Institucional, Lima, Perú. https://hdl.handle.net/20.500.12867/5634

Capcha Coronado, E. (2018). *Implementación del Framework de automatización de proceso de QA en un proyecto de diseño de software en una consultora [Tesis de grado, Universidad Nacional Mayor de San Marcos].* UNMSM-Tesis, Lima, Perú. https://hdl.handle.net/20.500.12672/10210

Cortés Pabón, Á. M. (2020). *Automatización de pruebas de regresión para reducción de tiempo de entrega de nuevas versiones de software [Tesis de postgrado, Universidad de Chile].* Repositorio UChile, Santiago, Chile. https://repositorio.uchile.cl/handle/2250/177640

Crespo, A. (11 de Enero de 2018). *Automatización de Pruebas*. Excentia Site Web: https://www.excentia.es/automatizacion-de-pruebas

Fernandez Avalos, J. L. (2018). *Automatización de Procesos para mejorar las Pruebas de Software en el área de calidad del Banco de Crédito [Tesis de maestría, Universidad Cesar Vallejo].* UCV-Institucional, Lima, Perú. https://hdl.handle.net/20.500.12692/23871

Fewster, M., & Graham, D. (2000). *Software Test Automation Effective use of test execution tools* (First ed.). England: Pearson Education.

Gerena, L. (2018). *Investigacion Aplicada.* Calameo WebSite: https://www.calameo.com/books/004243589cb44e615e1ef

GLENFORD, J., BADGETT, T., & SANDLER, C. (2012). *The Art of Software Testing* (Third ed.). New Jersey, EE.UU.: JohnWiley & Sons, Inc.,.

Gobierno del Perú. (31 de 10 de 2020). *Historia del Ministerio de Educación*. Gobierno del Perú Web Site: https://www.gob.pe/institucion/minedu/informes-publicaciones/1307863-historia-del-ministerio-de-educacion

Gobierno del Perú. (s.f.). *Organización del Ministerio de Educación*. Gobierno del Perú Web Site: https://www.gob.pe/institucion/minedu/organizacion

Goméz, M. (2006). *Introducción a la metodología de la investigación científica.* Córdova, Argentina: Editorial Brujas.

Hayes, L. (1995). *The Automated Testing Handbook.* Dallas, Texas: Software Testing Institute.

Hérnandez, R., Fernandéz, C., & Baptista, M. (2014). *Metodología de la Investigación* (6ta ed.). México D.F., Santa Fe, México: McGRAW-HILL / INTERAMERICANA EDITORES, S.A.

IBM. (2022). *¿Qué es la prueba de software y cómo funciona?* IBM Corp. Web Site: https://www.ibm.com/es-es/topics/software-testing

ISTQB, I. S. (2018). *Probador Certificado ISTQB - Programa de Estudio de Nivel Básico.* (S. T. Board, Trad.)

Medina Yacupoma, M. (2020). *Automatización de pruebas para proyectos ágiles aplicando el desarrollo dirigido por comportamiento para una compañía de líneas de belleza [Tesis de grado, Universidad Tecnológica del Perú].* UTP-Institucional, Lima, Perú. https://hdl.handle.net/20.500.12867/3166

MINEDU. (2015). *Organigrama Ministerio de Educacion del Perú[Gráfico].* Portal Transparencia Web site: http://www.minedu.gob.pe/p/xtras/organigrama_minedu.pdf

MINEDU. (2019). *Organigrama OTIC - MINEDU[Gráfico].* Transparencia Minedu Web Site: http://www.minedu.gob.pe/transparencia/2020/pdf/RM-N-620-2019-MINEDU-Aprobar_el_Plan_de_Gobierno_Digital_del_Minedu_2019-2022.pdf

Patton, R. (2001). *Software Testing.* United States of America: Sams Publishing.

Pressman, R. (2010). Ingeniería del Software un Enfoque Práctico. En *Aseguramiento de la Calidad de Software* (Séptima ed., págs. 371- 400). México: McGRAW-HILL INTERAMERICANA EDITORES, S.A.

Rivera Martínez, C. A. (2018). *Automatización de pruebas de regresión [Tesis de postgrado, Universidad de Chile].* Repositorio Uchile, Santiago, Chile. https://repositorio.uchile.cl/handle/2250/165608

Sarco, P. (2019). *Testing Automatizado.* Blog Pablo Sarco: https://josepablosarco.wordpress.com/2009/09/29/testing-automatizado-automated-testing/

Siroky, D. (2017). *The glitch economy: Counting the cost of software failures*. Cloud Computing News: https://www.cloudcomputing-news.net/news/2017/oct/30/glitch-economy-counting-cost-software-failures/

Toledo, F. (Febrero de 2022). *Tendencias de testing en Latam para 2022 - opinión de expertos*. Federico Toledo Blog: https://www.federico-toledo.com/tendencias-de-testing-en-latam-para-2022-opinion-de-expertos/

Toledo, F., Curcio, A., & Scuoteguazza, G. (2014). Introducción a las Pruebas de Sistemas de Información. En *Automatización de Pruebas Funcionales* (págs. 93-129). Montevideo: Abstracta.

Anexos

Anexo 1 *Manual de instalación y configuración del patrón Page Object Model*

Configuración De

Ambiente Para

Automatización De Pruebas

De Aplicaciones Web

Índice

1 Glosario de Términos

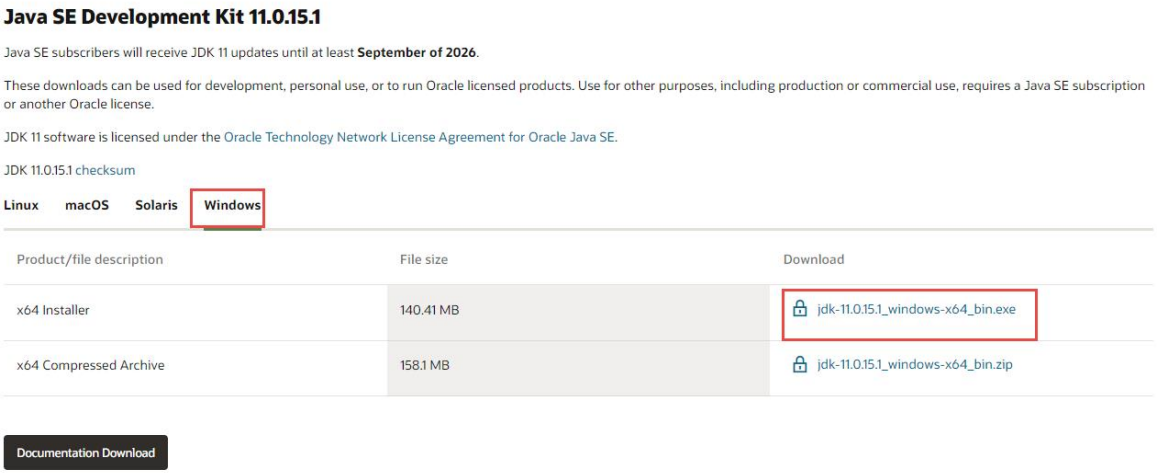
Aplicación	ámbito	
IntelliJ IDEA	Entorno de desarrollo Integrado (IDE) Para el desarrollo de Programas informáticos en lenguaje Java.	
JDK – Java Development Kit	Es el Kit de desarrollo de Java, para desarrollar aplicaciones en este lenguaje. Contiene herramientas, utilidades, documentos y ejemplos.	
Selenium WebDriver	Es un framework de automatización web que le permite ejecutar sus pruebas contra diferentes navegadores	
Junit	Se trata de un Framework Open Source de java para la automatización de las pruebas (tanto unitarias, como de integración)	
TestNG	Es un framework para pruebas y testing que trabaja con Java y está basado en Junit (para Java) y NUnit (para .NET), pero introduciendo nuevas funcionalidades que lo hacen más potente.	

Cucumber	Es una herramienta que se utiliza para automatizar pruebas basadas en BDD (Behavior Driven Development). Utiliza el lenguaje Gherkins	
Apache Maven	Es una herramienta que se utiliza en la gestión y construcción de software. Posee la capacidad de realizar ciertas tareas claramente definidas desde la compilación hasta el despliegue. Para el manejo de dependencias utiliza el archivo POM (Proyect Object Model) basado en formato XML	
Gradle	Gradle es un sistema de automatización de construcción de código de software que construye sobre los conceptos de Apache Ant y Apache Maven e introduce un lenguaje específico del dominio (DSL) basado en Groovy en vez de la forma XML utilizada por Apache Maven para declarar la configuración de proyecto	

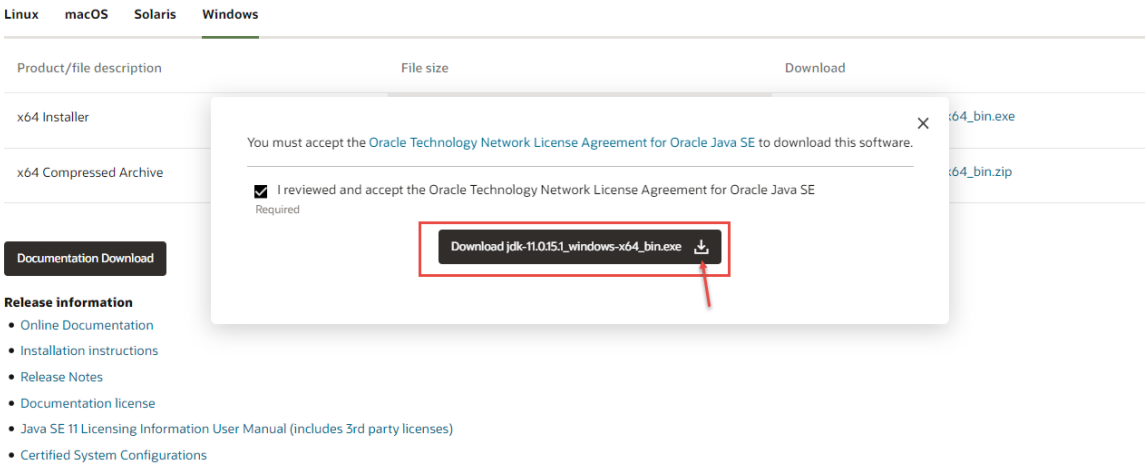
2 Descargar, Instalar y Configurar JDK de JAVA 8.0 o Superior

2.1 Descargar JDK de Java

1. Descargar el kit de desarrollo de java desde la página de Oracle o haciendo clic en el siguiente link: https://www.oracle.com/java/technologies/downloads/#java11-windows.
2. Se puede cambiar el paquete a descargar de acuerdo al sistema operativo donde se desea configurar el JDK. Para nuestro caso es Windows[1].



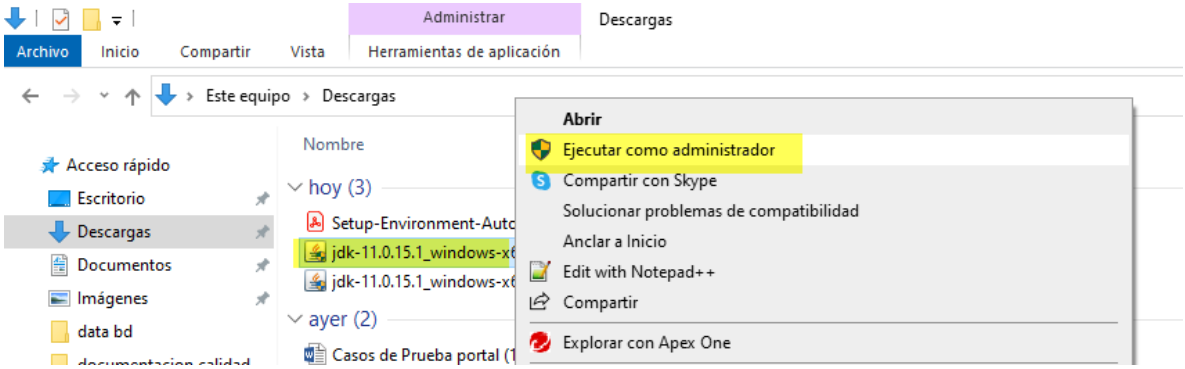
3. Clic en descargar y aceptamos los términos y condiciones.



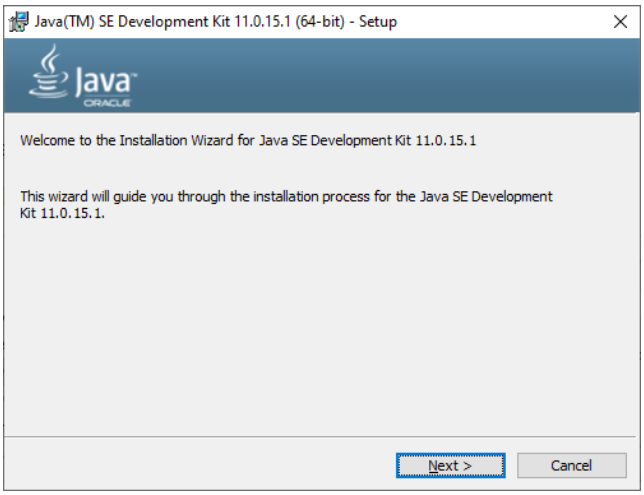
[1] Para descargar el JDK de java se tiene que tener una cuenta en Oracle. Sino la tuviera lo puede crear en el mismo proceso de la descarga.

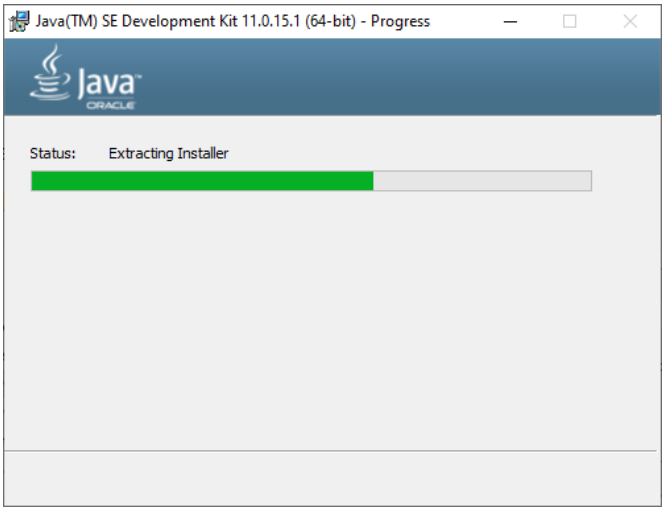
2.2 **Instalar JDK de Java**

1. Ejecutamos la aplicación de java en modo administrador



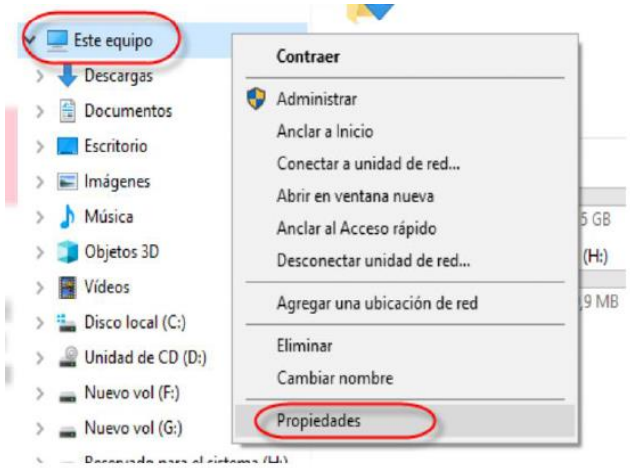
2. Se instala con las opciones por defecto



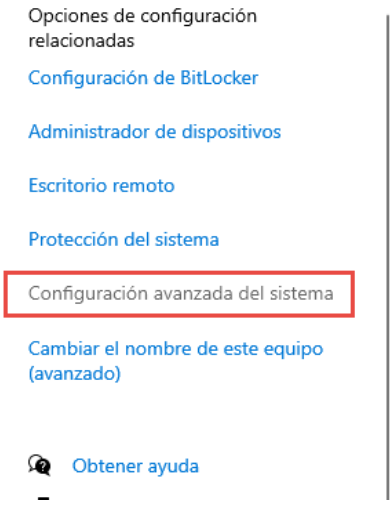


2.3 Configuración de variables de entorno

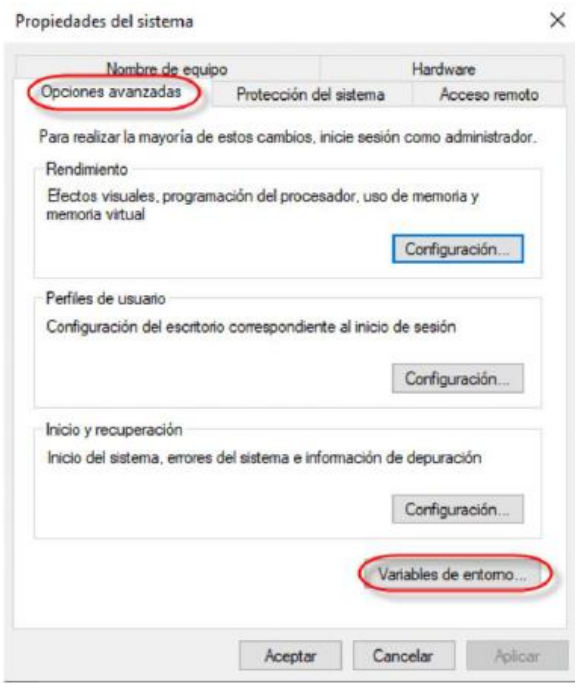
1. Ir al explorador de carpetas. Clic derecho sobre equipo. Clic sobre propiedades



2. Clic en configuración avanzada del sistema

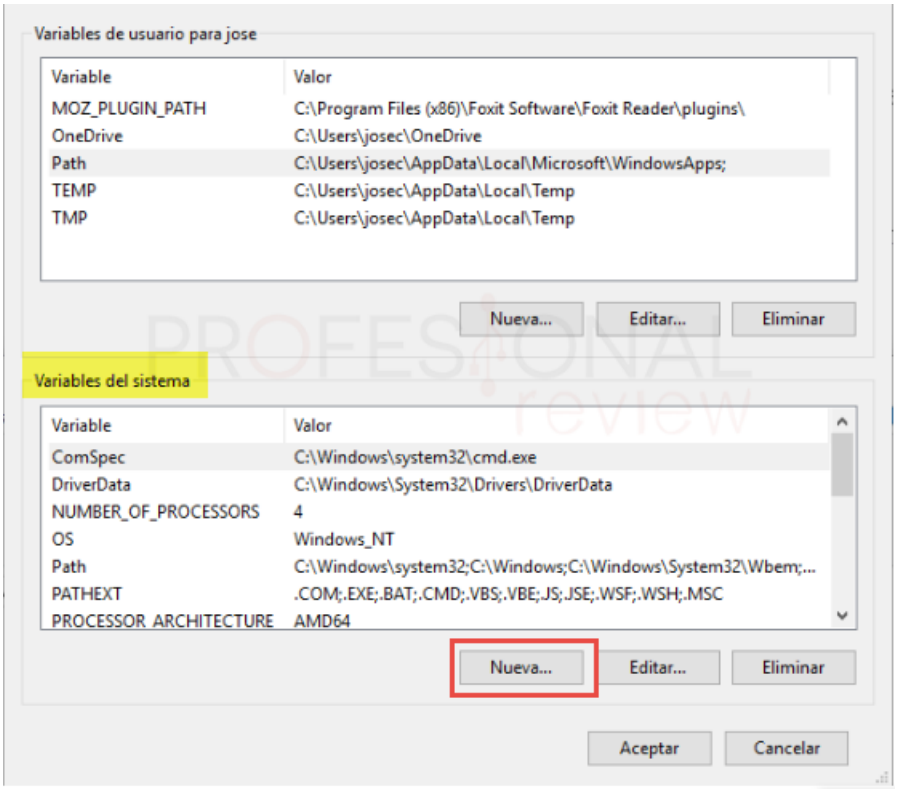


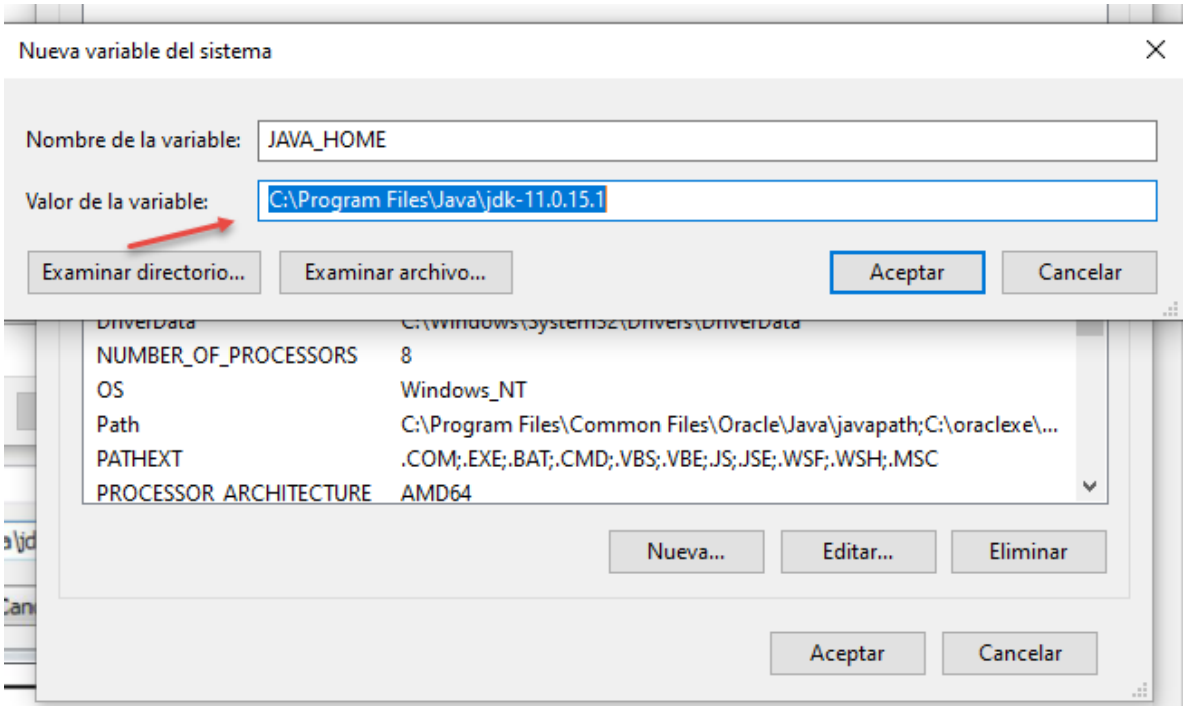
3. En la ventana de propiedades del sistema, ubicarse en la pestaña “opciones avanzadas” y hacer clic en variables de entorno.



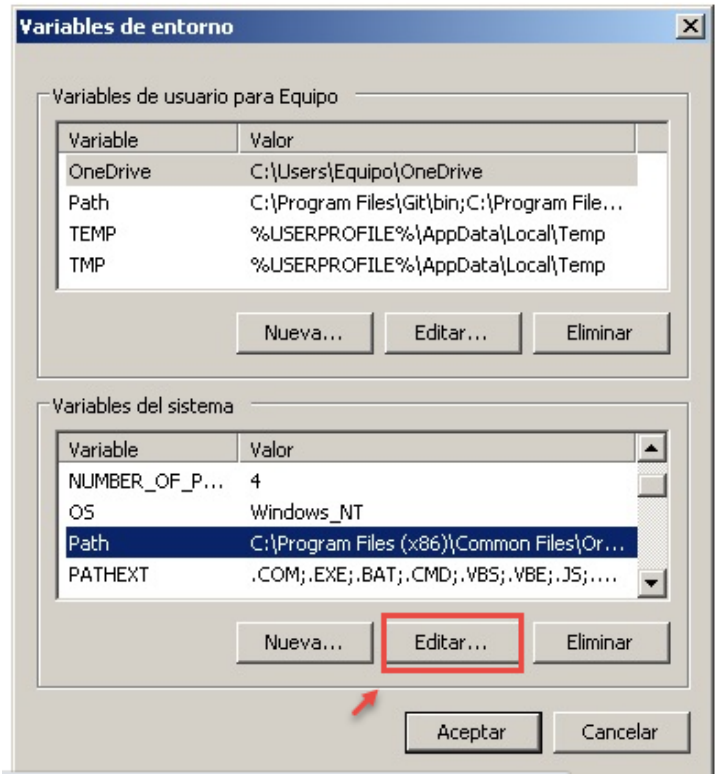
4. Para añadir una nueva variable de entorno, pulsamos sobre “Nuevo”. Nos aparecerá una nueva línea en donde tendremos que introducir la variable JAVA_HOME que debe de contener la ruta del JDK de java. Luego de Setear la ruta del JDK hacemos clic en aceptar.

   JAVA_HOME= C:\Program Files\Java\jdk-11.0.15.1

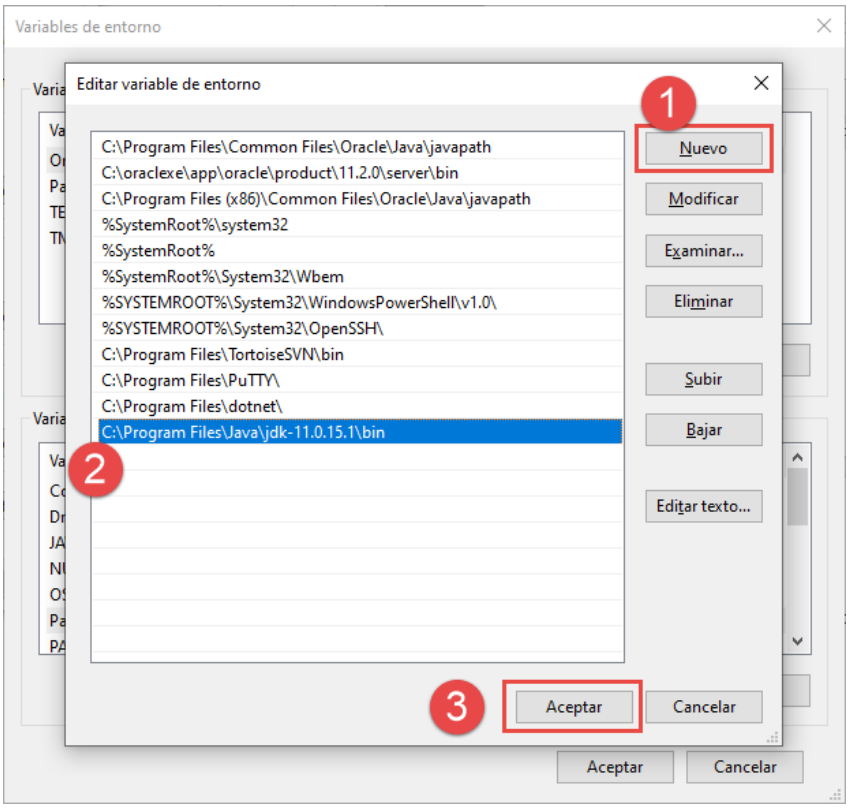




5. Seleccionamos la variable “Path” y clic en el botón “Editar”

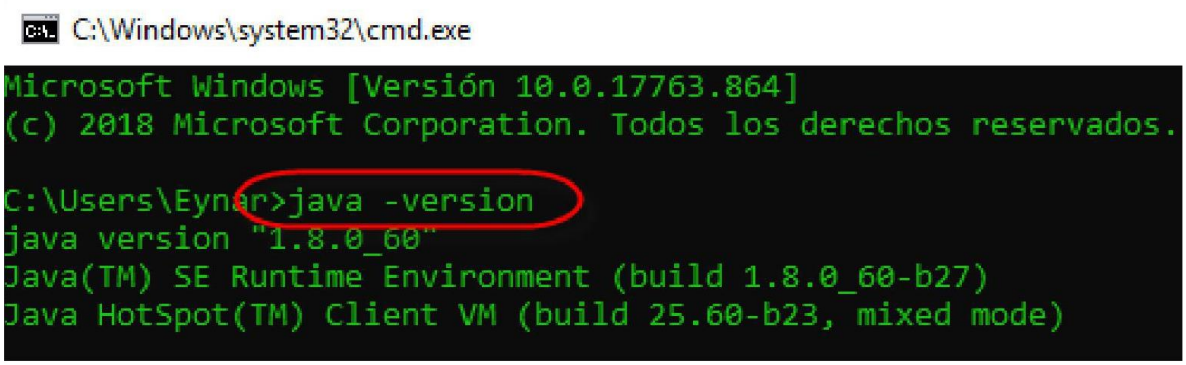


6. En la ventana agregar la ruta del JDK/bin. Para ello hacer clic en el botón “Nuevo”, direccionar a la ruta donde está instalado el JDK, hasta el bin o copiamos directamente la ruta **“C:\Program Files\Java\jdk-11.0.15.1\bin”**. Luego clic en aceptar



2.4 Verificar la configuración de Java

1. Abrir una consola y ejecutar el comando “java -versión. Si sale la versión del java, entonces esta todo correcto.

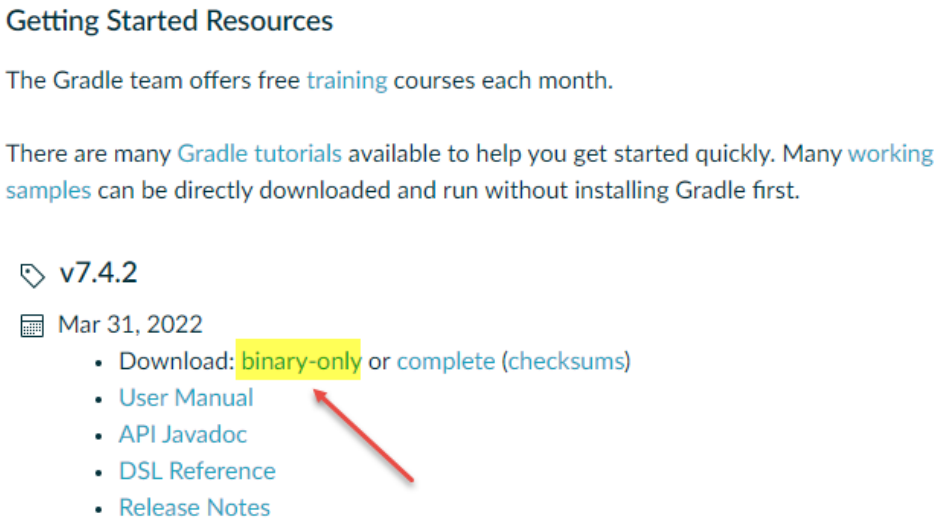


3 Descargar, Instalar y Configurar Gradle 5.5 o Superior

3.1 Descargar Gradle

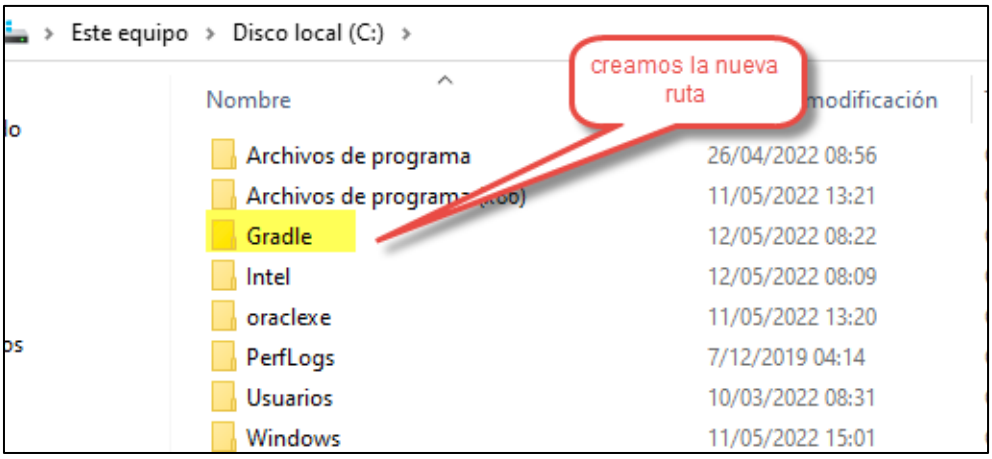
1. Descargar Gradle desde la misma página: https://gradle.org/install. Hacemos clic en download.

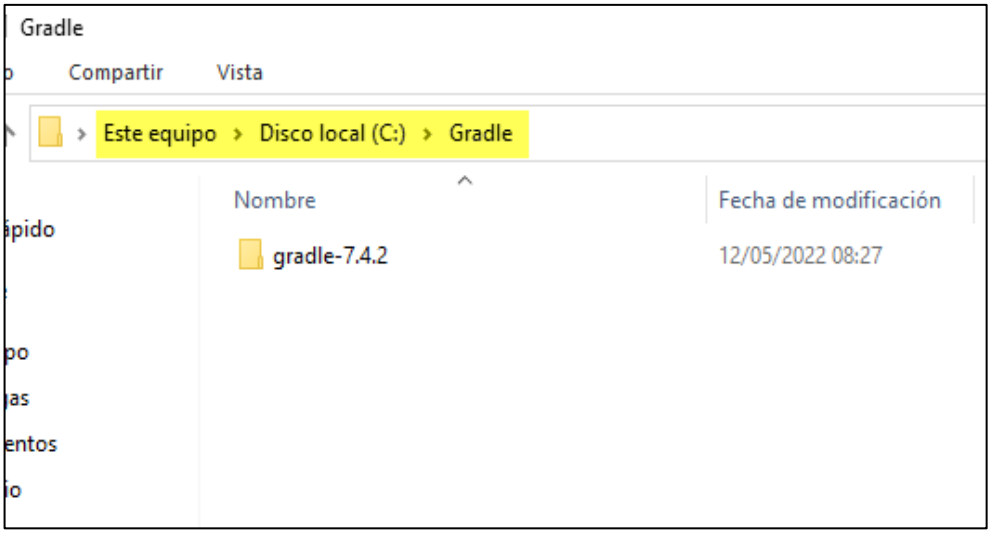
2. Elegimos la versión y hacemos clic en “Binary-only” Esperamos a que se termine la descarga.



3.2 Instalar Gradle

1. Crear un nuevo directorio en “**C:\Gradle**”. Luego ir a la ruta donde Gradle fue descargado, descomprimir el contenido y copiar todo a la nueva ruta.

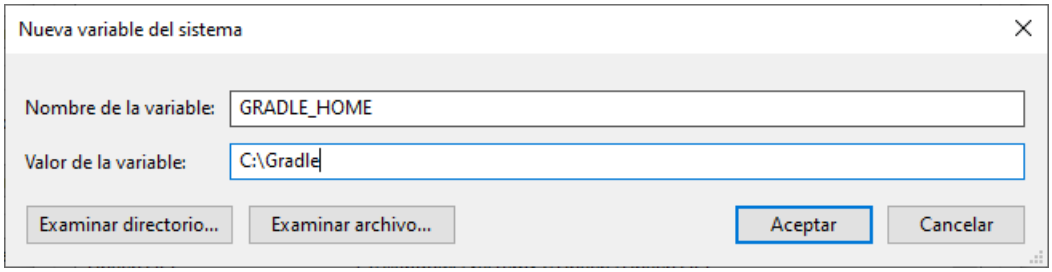




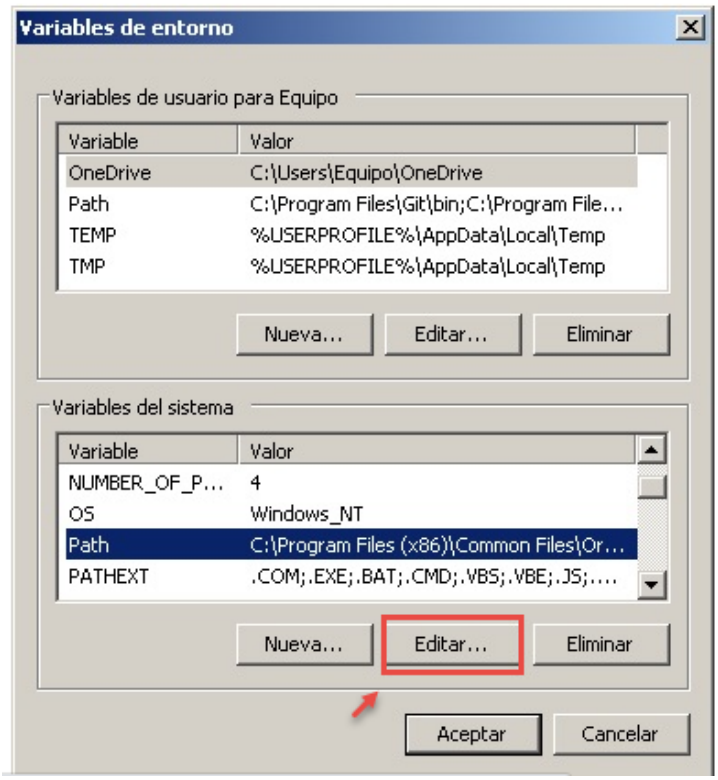
3.3 Configurar Gradle

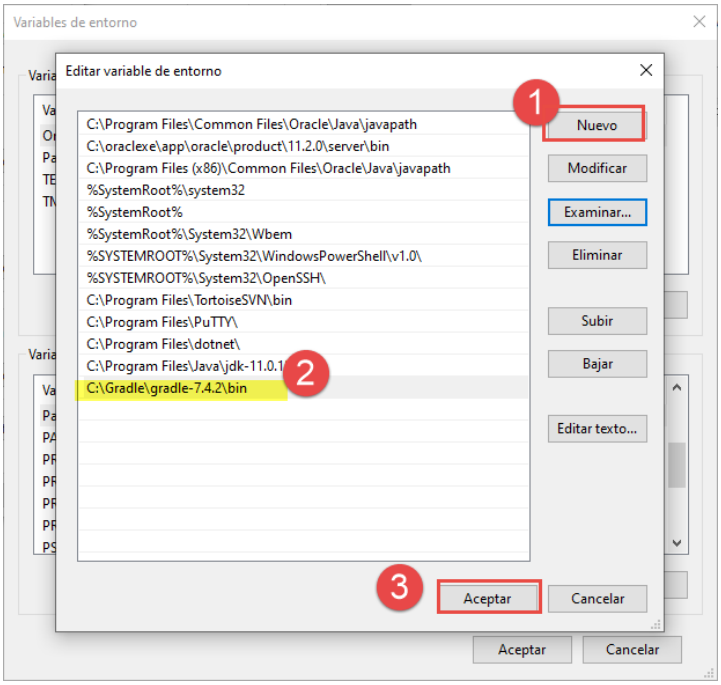
1. Hacer los mismos pasos cuando se configuraron las variables de entorno para java. Para ello, agregar la variable "**GRADLE_HOME**" en variables de entorno la ruta de la carpeta donde se instaló Gradle.

   GRADLE_HOME= C:\Gradle



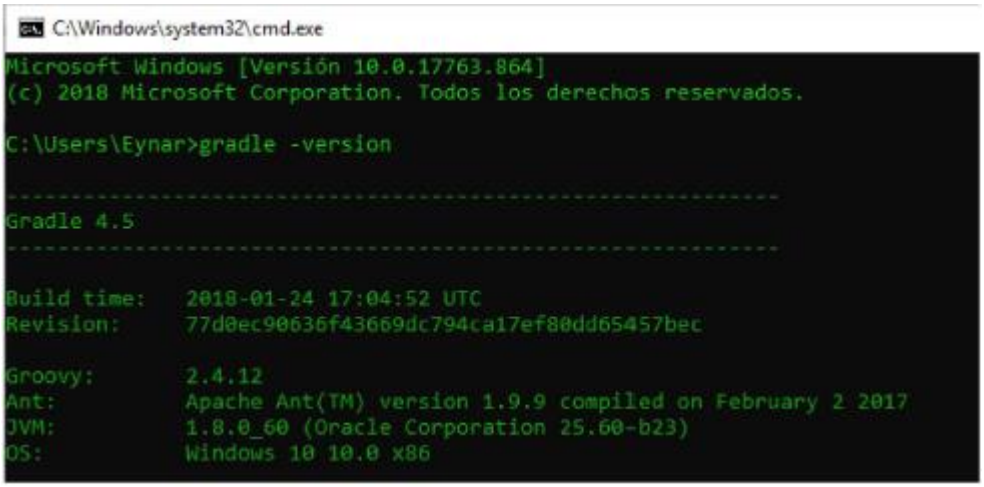
2. Agregar la ruta de GRADLE/bin en la variable de entorno Path.


Variables de entorno
Variables de usuario para Equipo
Variable
Valor
OneDrive
C:\Users\Equipo\OneDrive
Path
C:\Program Files\Git\bin;C:\Program File...
TEMP
%USERPROFILE%\AppData\Local\Temp
TMP
%USERPROFILE%\AppData\Local\Temp
Nueva...
Editar...
Eliminar
Variables del sistema
Variable
Valor
NUMBER_OF_P...
4
OS
Windows_NT
Path
C:\Program Files (x86)\Common Files\Or...
PATHEXT
.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;....
Nueva...
Editar...
Eliminar
Aceptar
Cancelar


Variables de entorno
Editar variable de entorno
C:\Program Files\Common Files\Oracle\Java\javapath
C:\oraclexe\app\oracle\product\11.2.0\server\bin
C:\Program Files (x86)\Common Files\Oracle\Java\javapath
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\TortoiseSVN\bin
C:\Program Files\PuTTY\
C:\Program Files\dotnet\
C:\Program Files\Java\jdk-11.0.1
C:\Gradle\gradle-7.4.2\bin
1
2
3
Nuevo
Modificar
Examinar...
Eliminar
Subir
Bajar
Editar texto...
Aceptar
Cancelar
Aceptar
Cancelar

3.4 **Verificar la configuración de Gradle**

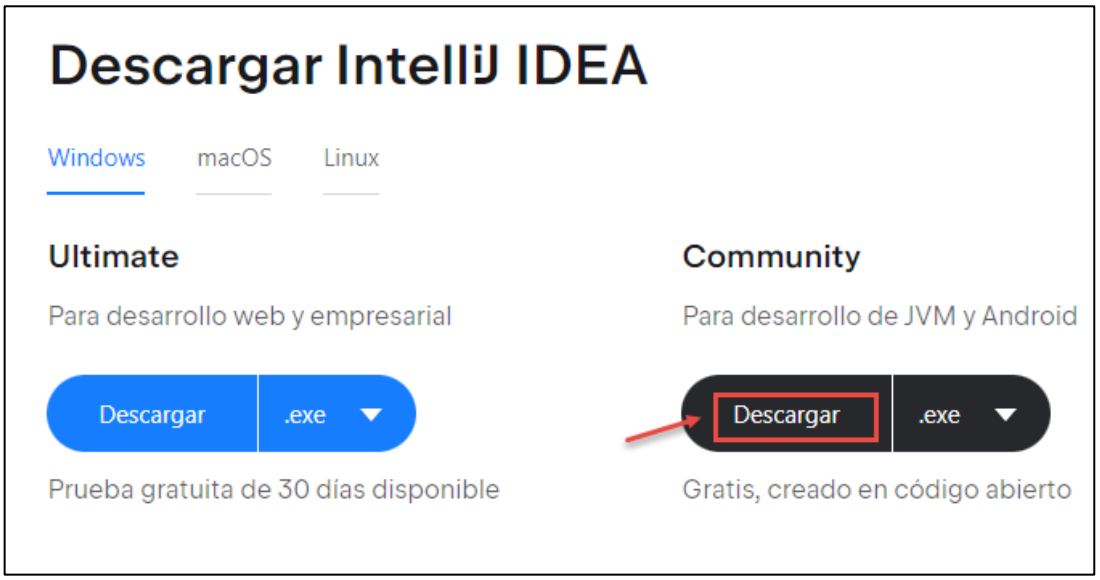
1. Abrir consola de CMD, y ejecutar el siguiente comando “Gradle -versión”



4 Descargar, Instalar y Configurar IntelliJ IDEA

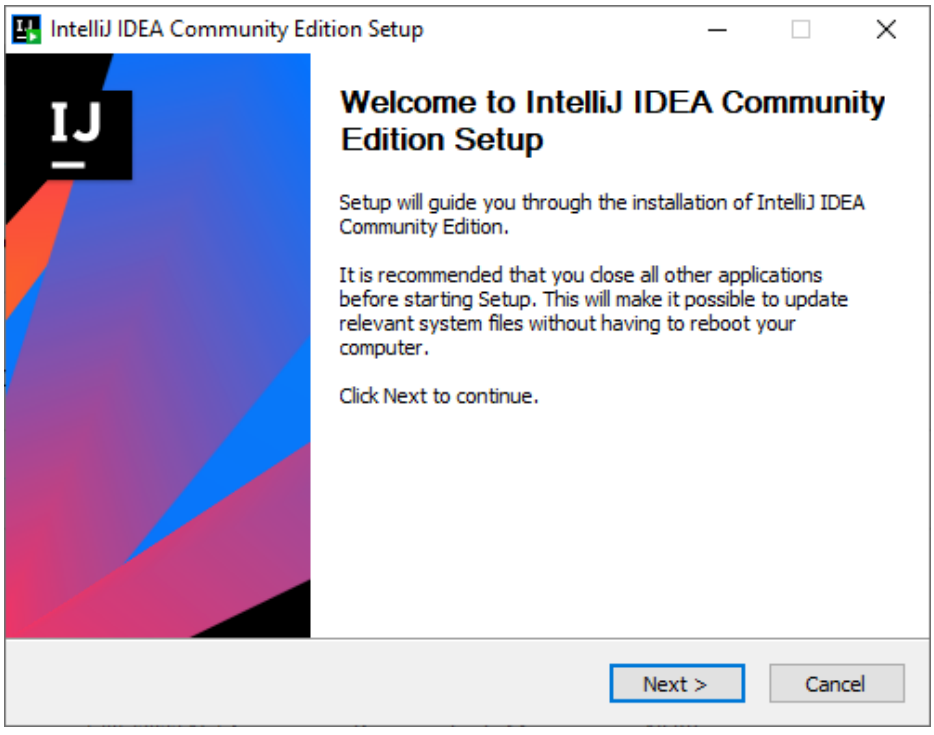
4.1 Descargar IntelliJ IDEA (community)

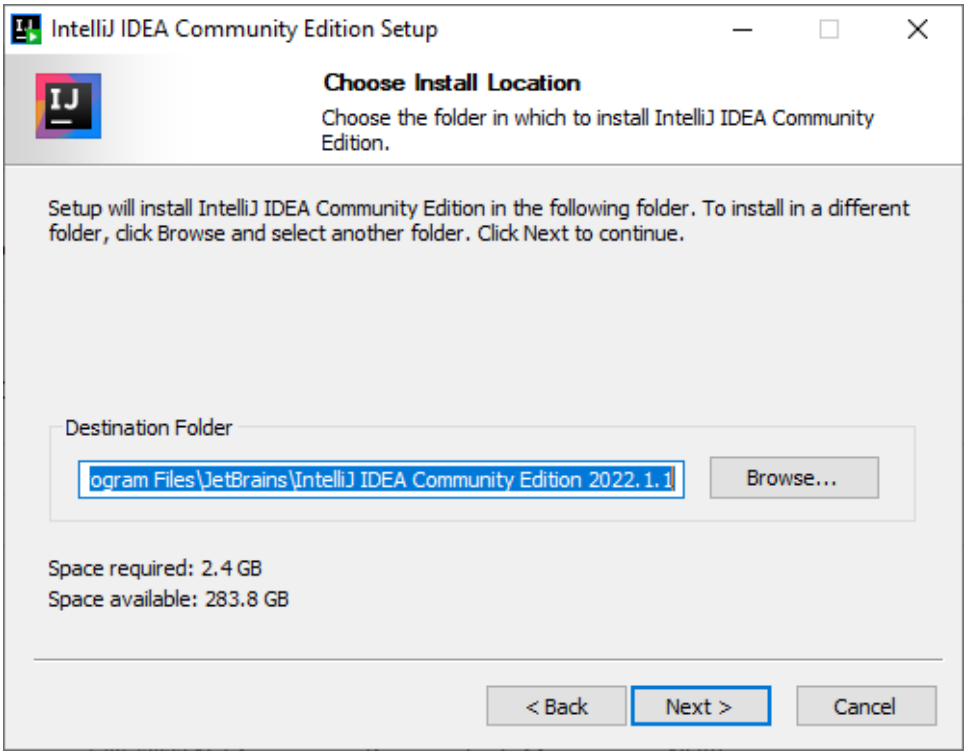
1. Descargar IntelliJ IDEA(Community) desde la misma página: https://www.jetbrains.com/es-es/idea/download/#section=windows. Clic en el botón descargar.



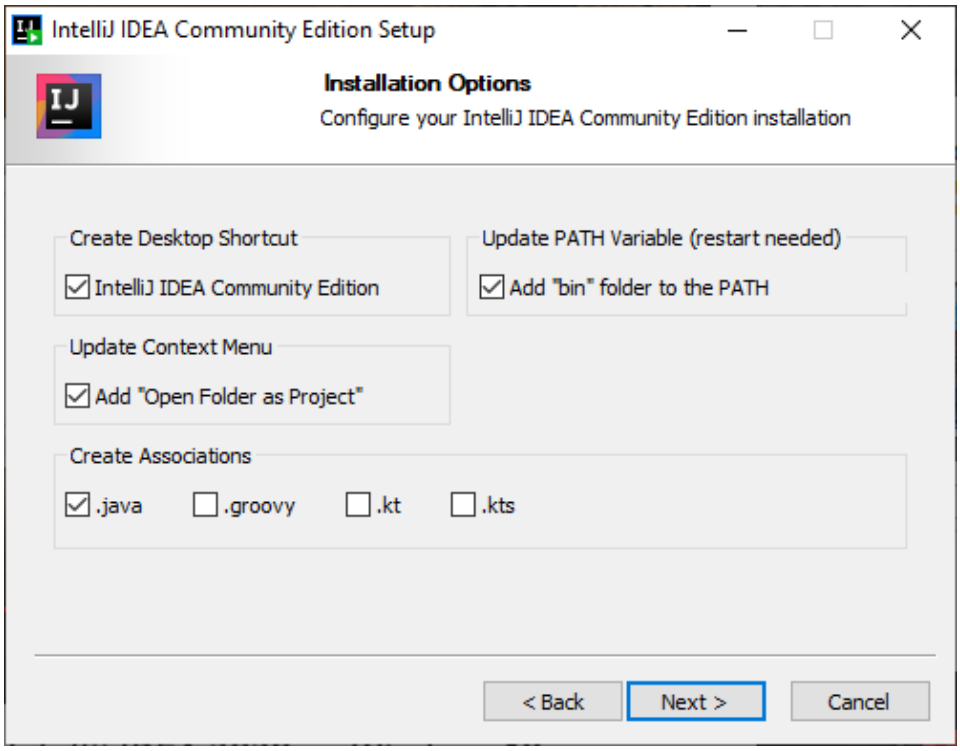
4.2 Instalar IntelliJ IDEA(Community)

1. Descargar IntelliJ IDEA(Community) desde la misma página: https://www.jetbrains.com/es-es/idea/download/#section=windows. Clic en el botón descargar.

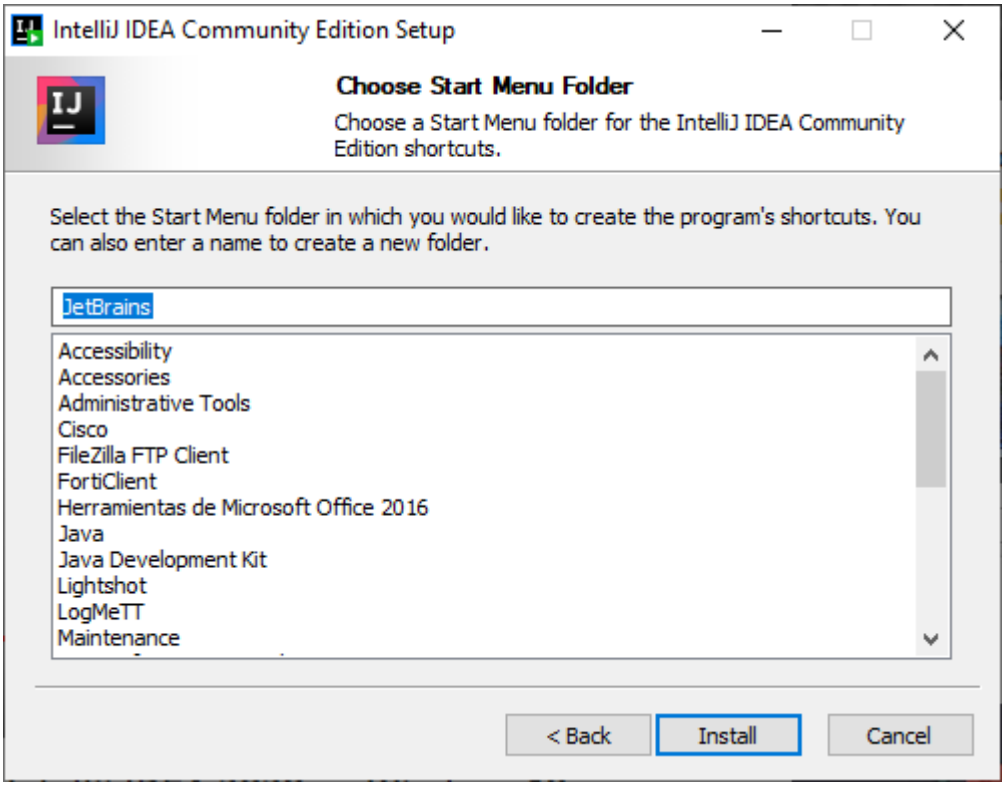


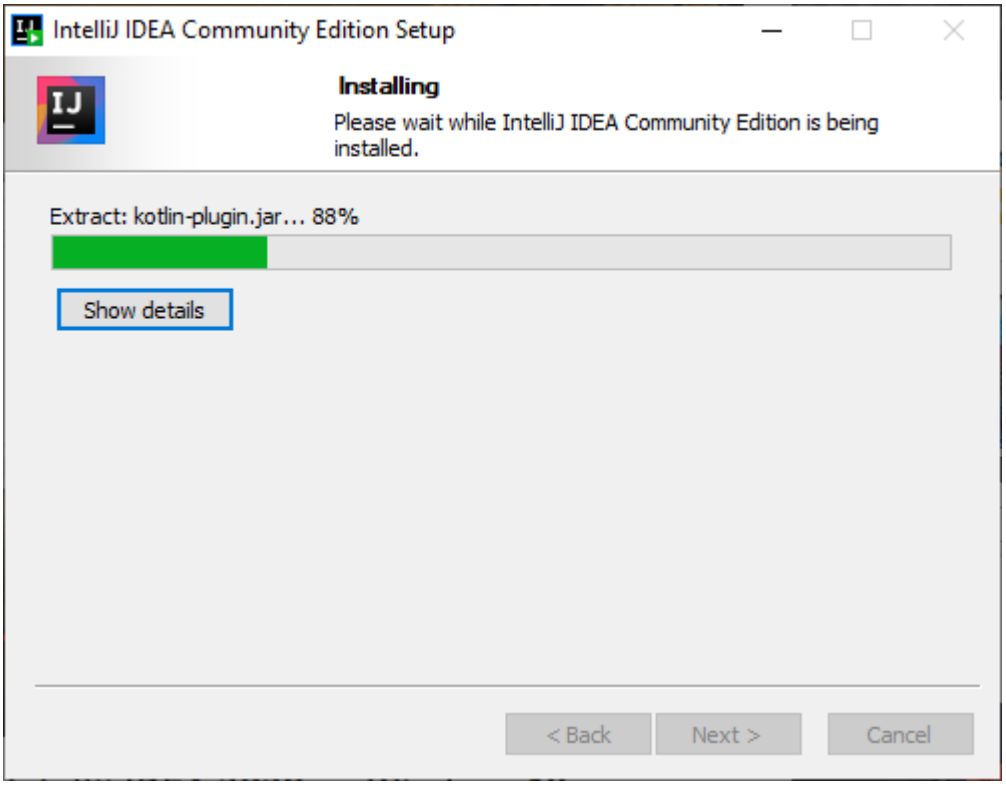


2. En esta ventana seleccionar las opciones marcadas en la imagen[2]. Y clic en continuar.

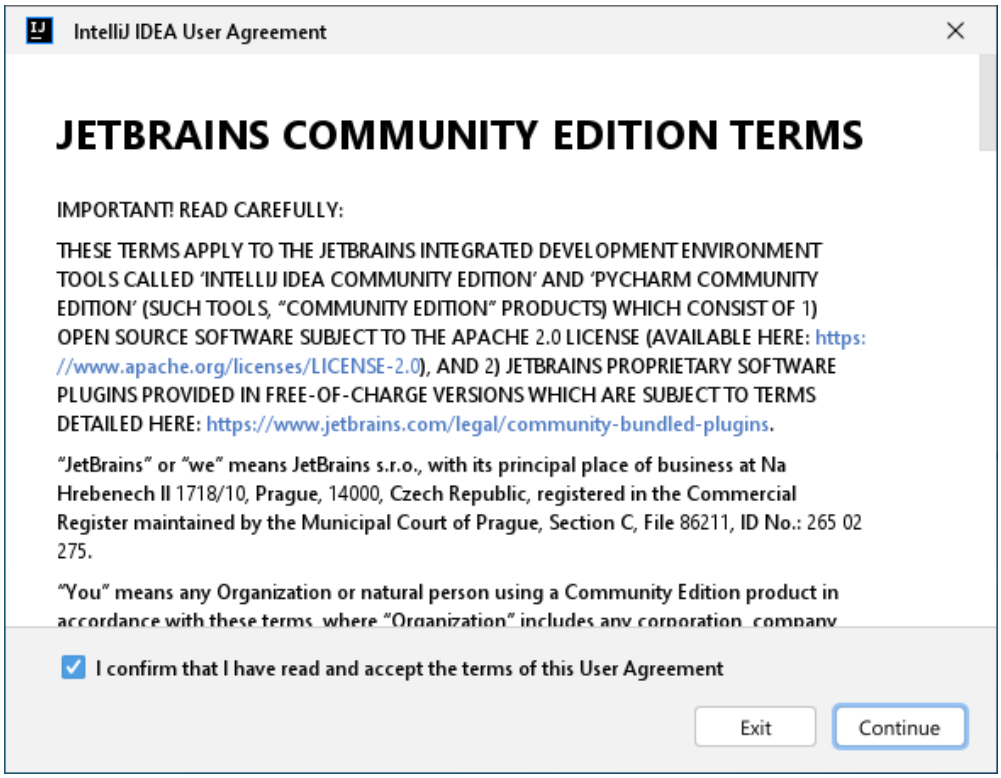


[2] Se crea un acceso directo en el escritorio del programa. Se agrega la ruta en la variable Path. Agrega un menú contextual para abrir una carpeta con IntelliJ haciendo clic derecho y asocia extensiones de archivo java al IDE.

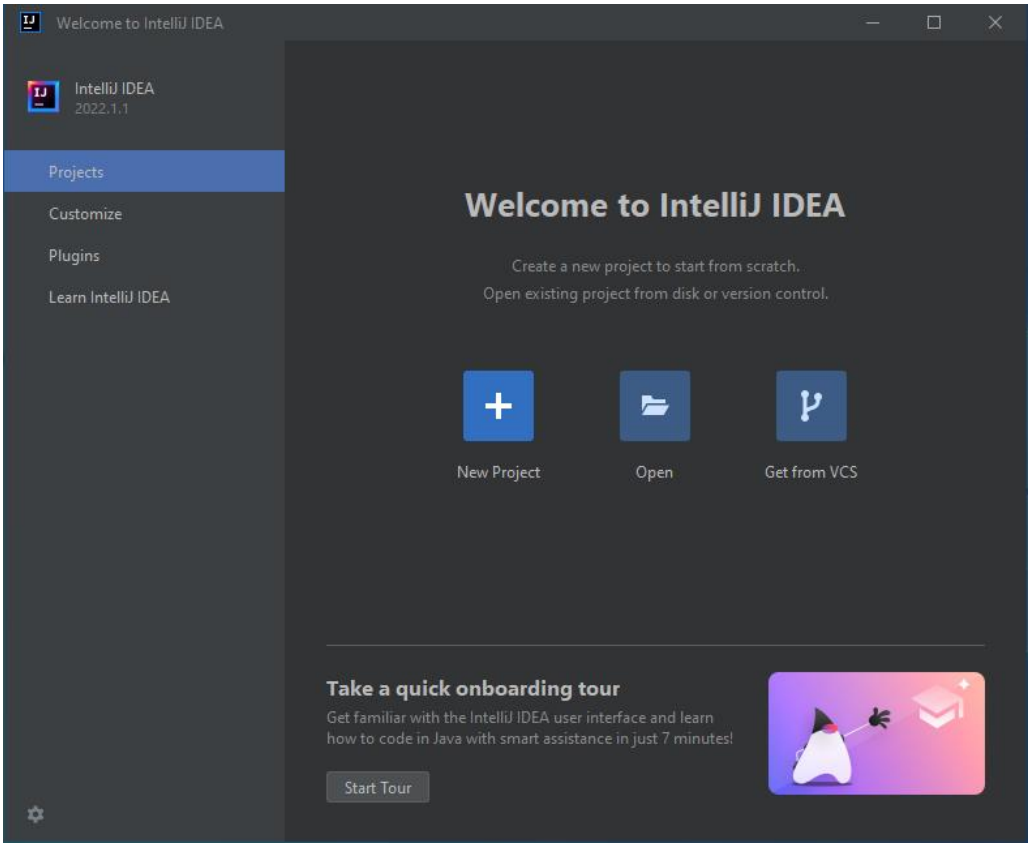

IntelliJ IDEA Community Edition Setup
Choose Start Menu Folder
Choose a Start Menu folder for the IntelliJ IDEA Community Edition shortcuts.
Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.
JetBrains
Accessibility
Accessories
Administrative Tools
Cisco
FileZilla FTP Client
FortiClient
Herramientas de Microsoft Office 2016
Java
Java Development Kit
Lightshot
LogMeTT
Maintenance
< Back
Install
Cancel


IntelliJ IDEA Community Edition Setup
Installing
Please wait while IntelliJ IDEA Community Edition is being installed.
Extract: kotlin-plugin.jar... 88%
Show details
< Back
Next >
Cancel

3. Reiniciar la computadora y continuar con la instalación.



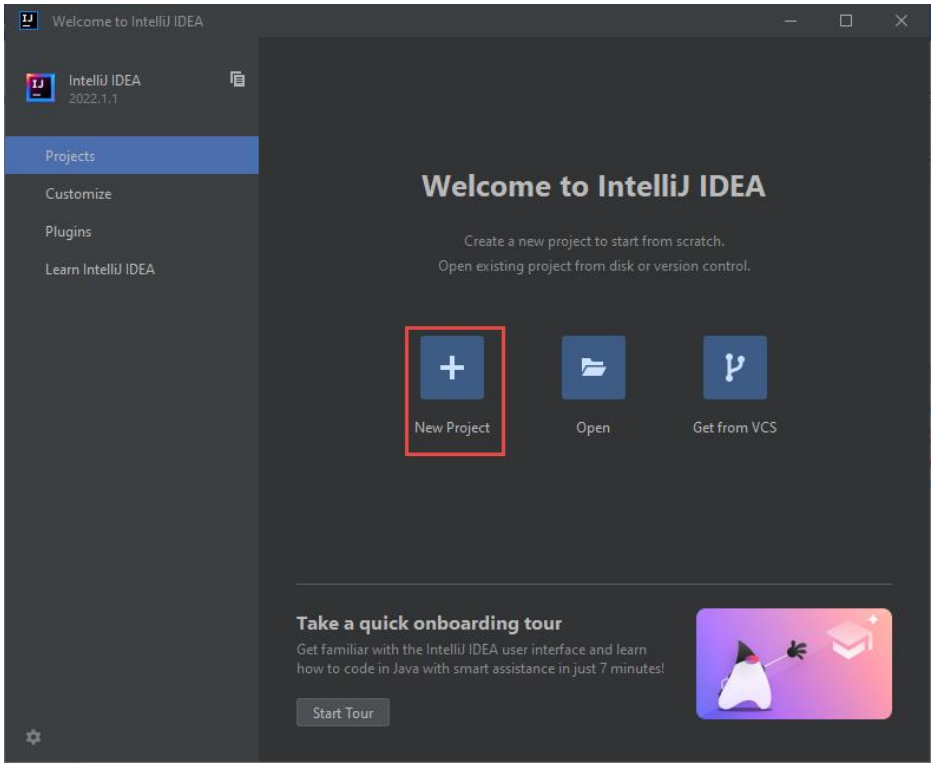

IntelliJ IDEA
Community 2022.1
JET
BRAINS

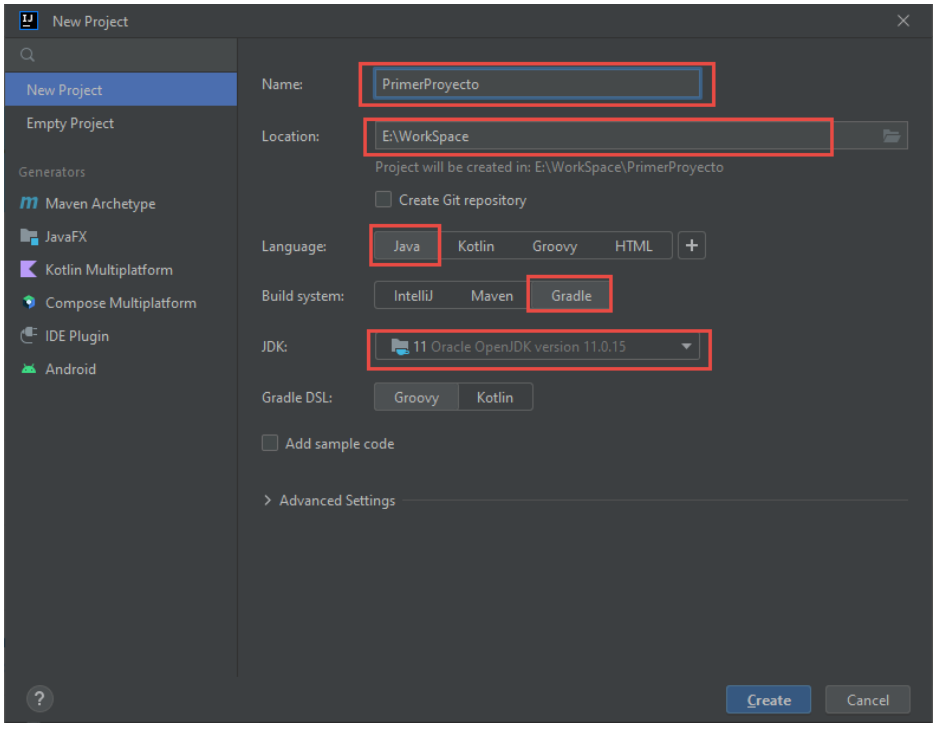

Welcome to IntelliJ IDEA
IntelliJ IDEA
2022.1.1
Projects
Customize
Plugins
Learn IntelliJ IDEA
Welcome to IntelliJ IDEA
Create a new project to start from scratch.
Open existing project from disk or version control.
New Project
Open
Get from VCS
Take a quick onboarding tour
Get familiar with the IntelliJ IDEA user interface and learn how to code in Java with smart assistance in just 7 minutes!
Start Tour

5 Primeros pasos con IntelliJ IDEA

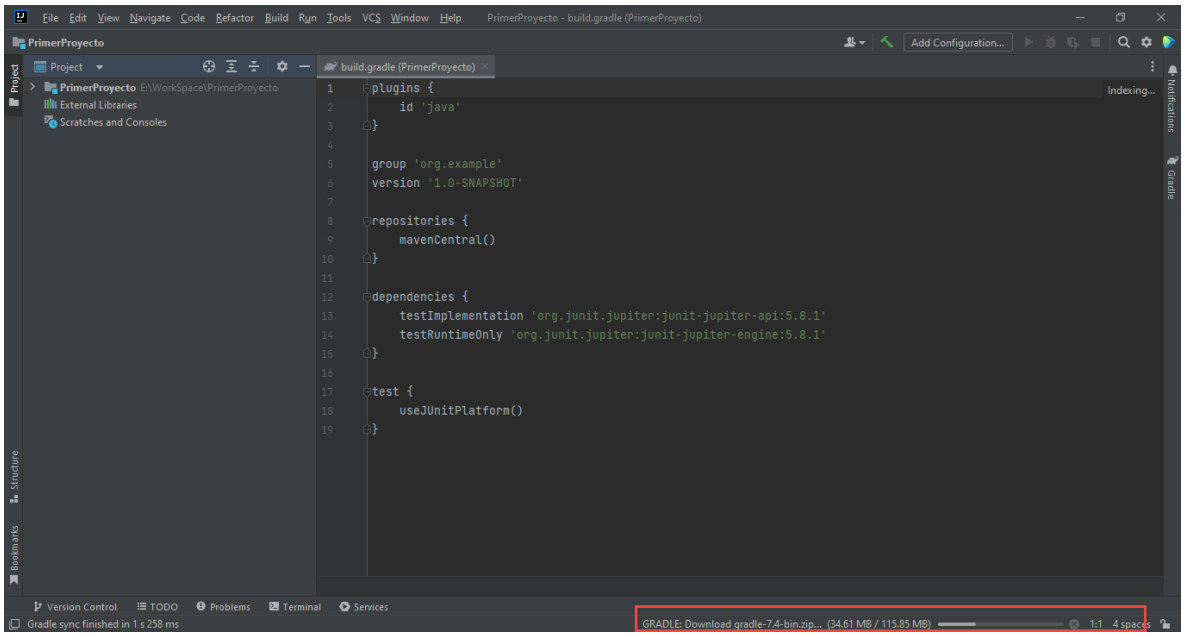
5.1 Crear proyecto JAVA Gradle

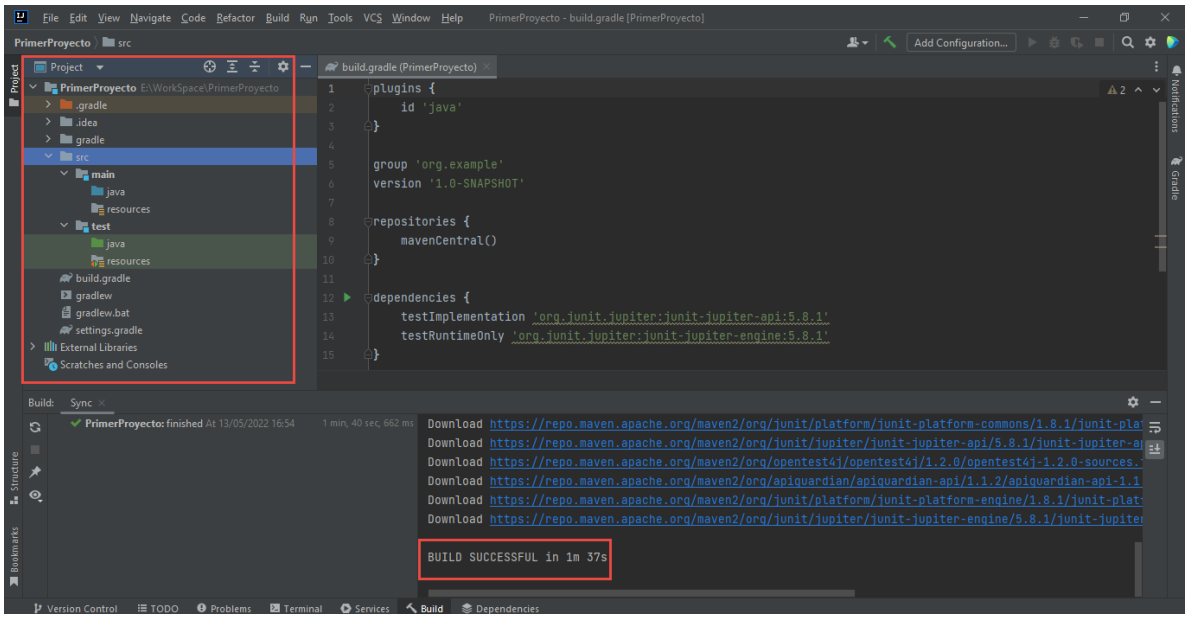
1. Abrir IntelliJ IDEA. Y hace clic en nuevo proyecto. Especificamos el espacio de trabajo, el nombre del proyecto, el lenguaje, el tipo de proyecto y el JDK.





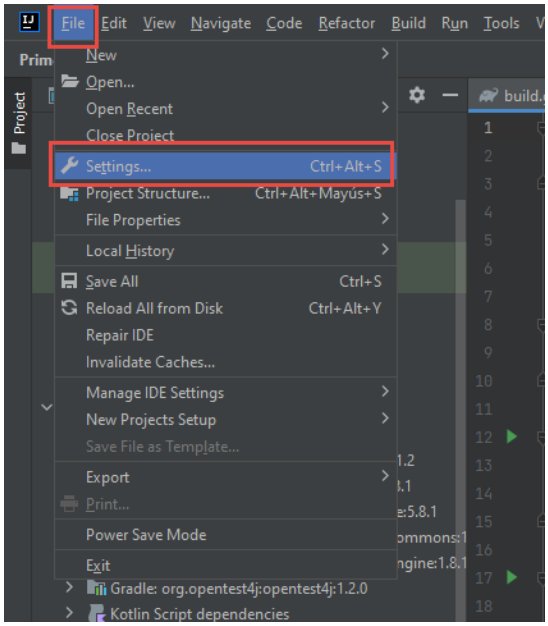
2. Clic en crear. IntelliJ IDEA cargara la estructura del proyecto java con Gradle. Ello tomará unos segundos. Gradle ejecuta un proceso Daemon que construirá la estructura inicial del proyecto y cargará las dependencias iniciales por defecto que están en el archivo build. Gradle.

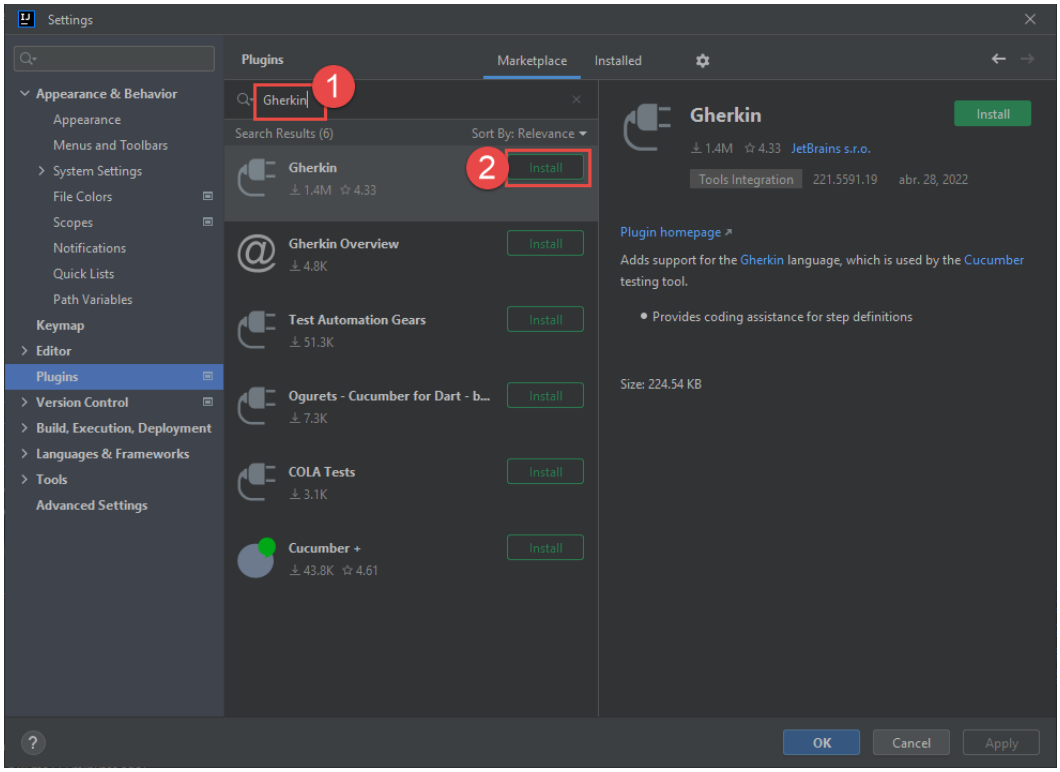




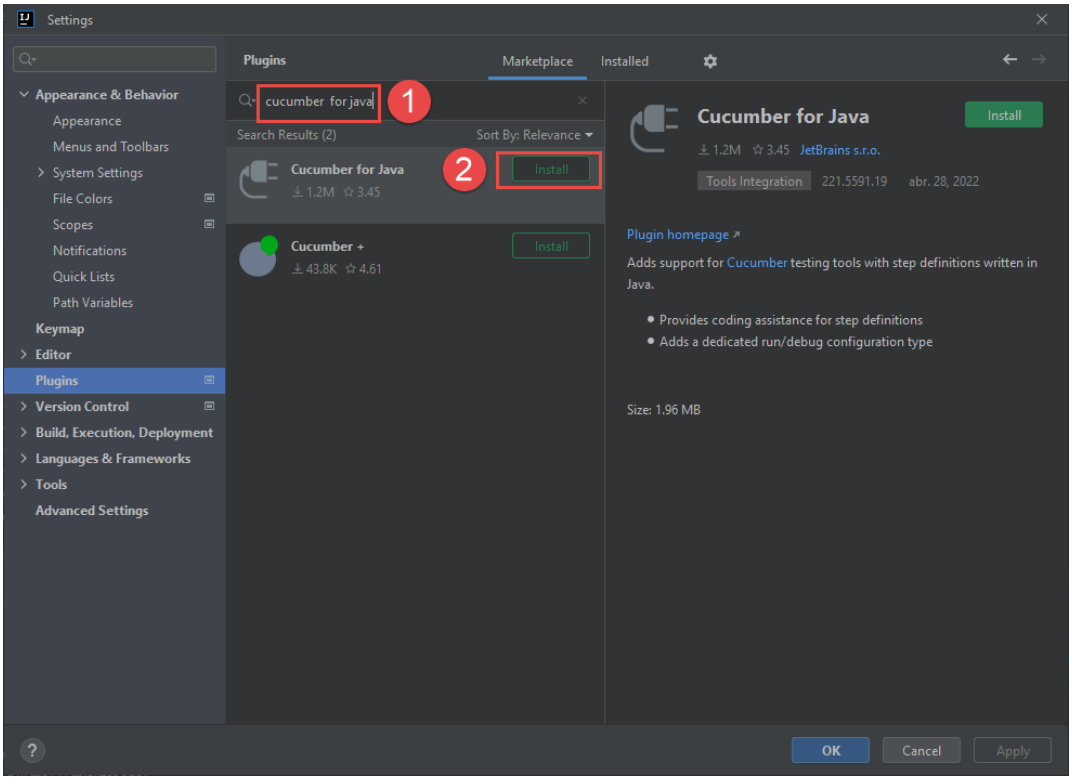
5.2 Instalar plugins de “Gherkin” y “Cucumber for Java”

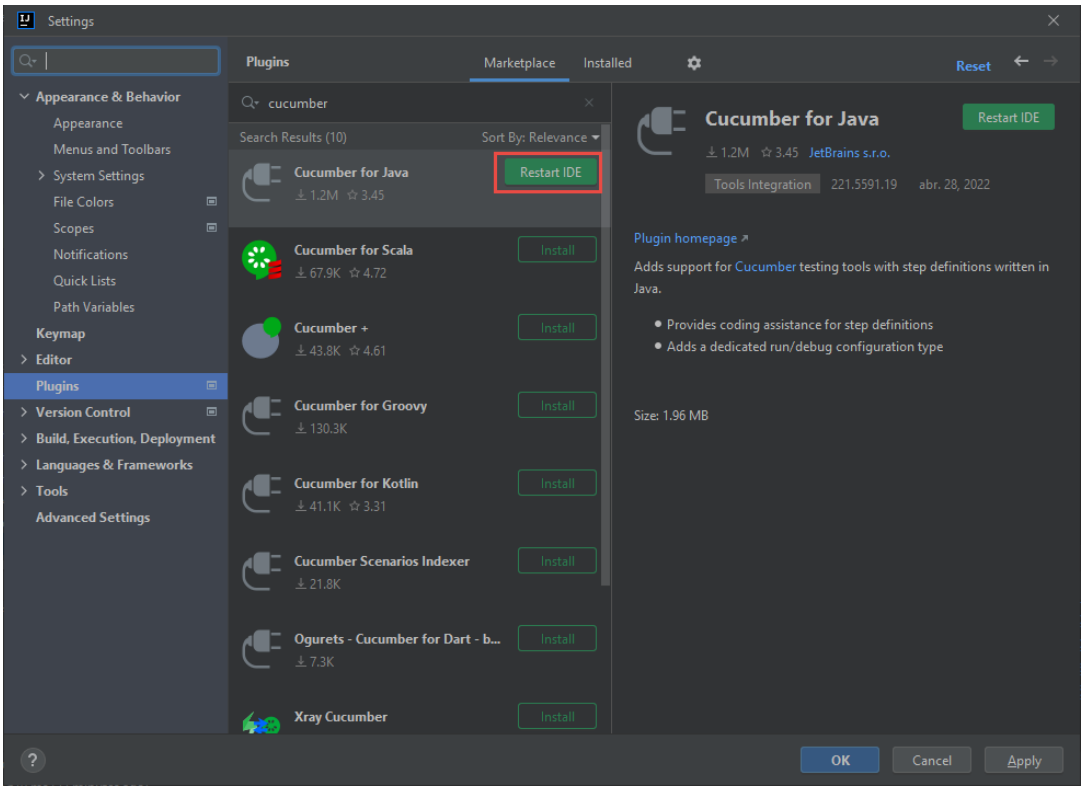
1. Instalar plugin “Gherkin”. Clic en File>>Settings>>Plugins. Digitar “Gherkin” y clic en install.





2. Instalar plugin “Cucumber for java”. Clic en File>>Settings>>Plugins. Digitar “cucumber for java” y clic en install. Luego reiniciar IDE

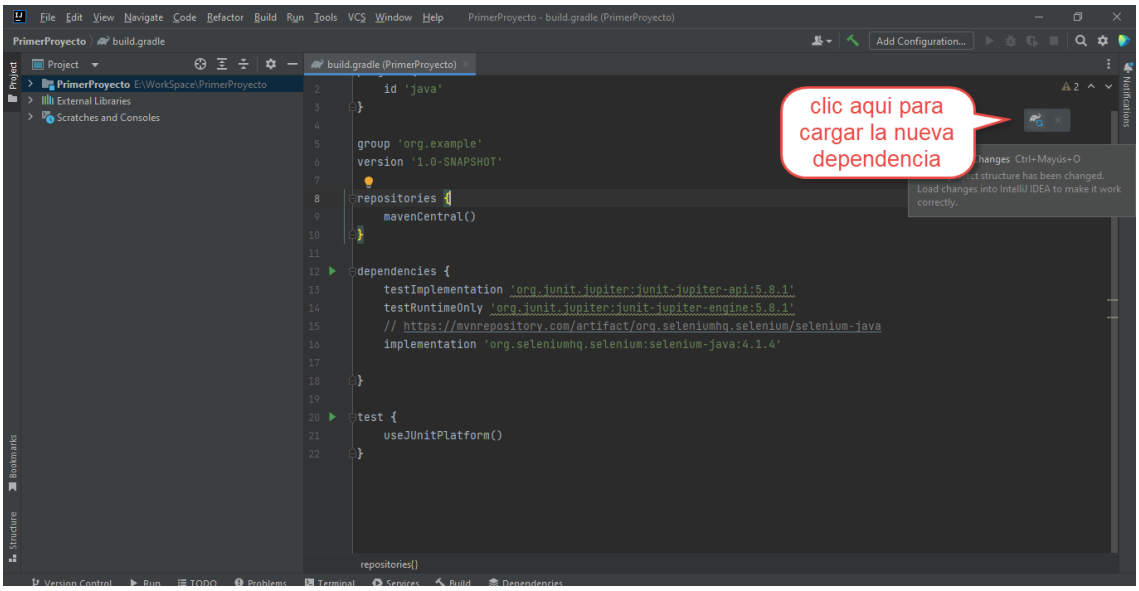

Settings
Plugins
Marketplace
Installed
cucumber for java
1
Search Results (2)
Sort By: Relevance
Cucumber for Java
1.2M 3.45
2
Install
Cucumber +
43.8K 4.61
Install
Cucumber for Java
Install
1.2M 3.45 JetBrains s.r.o.
Tools Integration 221.5591.19 abr. 28, 2022
Plugin homepage
Adds support for Cucumber testing tools with step definitions written in Java.
Provides coding assistance for step definitions
Adds a dedicated run/debug configuration type
Size: 1.96 MB
Appearance & Behavior
Appearance
Menus and Toolbars
System Settings
File Colors
Scopes
Notifications
Quick Lists
Path Variables
Keymap
Editor
Plugins
Version Control
Build, Execution, Deployment
Languages & Frameworks
Tools
Advanced Settings
OK
Cancel
Apply


Settings
Plugins
Marketplace
Installed
Reset
cucumber
Search Results (10)
Sort By: Relevance
Cucumber for Java
1.2M 3.45
Restart IDE
Cucumber for Scala
67.9K 4.72
Install
Cucumber +
43.8K 4.61
Cucumber for Groovy
130.3K
Cucumber for Kotlin
41.1K 3.31
Cucumber Scenarios Indexer
21.8K
Ogurets - Cucumber for Dart - b...
7.3K
Xray Cucumber
Cucumber for Java
Restart IDE
1.2M 3.45 JetBrains s.r.o.
Tools Integration 221.5591.19 abr. 28, 2022
Plugin homepage
Adds support for Cucumber testing tools with step definitions written in Java.
Provides coding assistance for step definitions
Adds a dedicated run/debug configuration type
Size: 1.96 MB
Appearance & Behavior
Appearance
Menus and Toolbars
System Settings
File Colors
Scopes
Notifications
Quick Lists
Path Variables
Keymap
Editor
Plugins
Version Control
Build, Execution, Deployment
Languages & Frameworks
Tools
Advanced Settings
OK
Cancel
Apply

5.1 Configurar las dependencias de Junit, Cucumber y Selenium

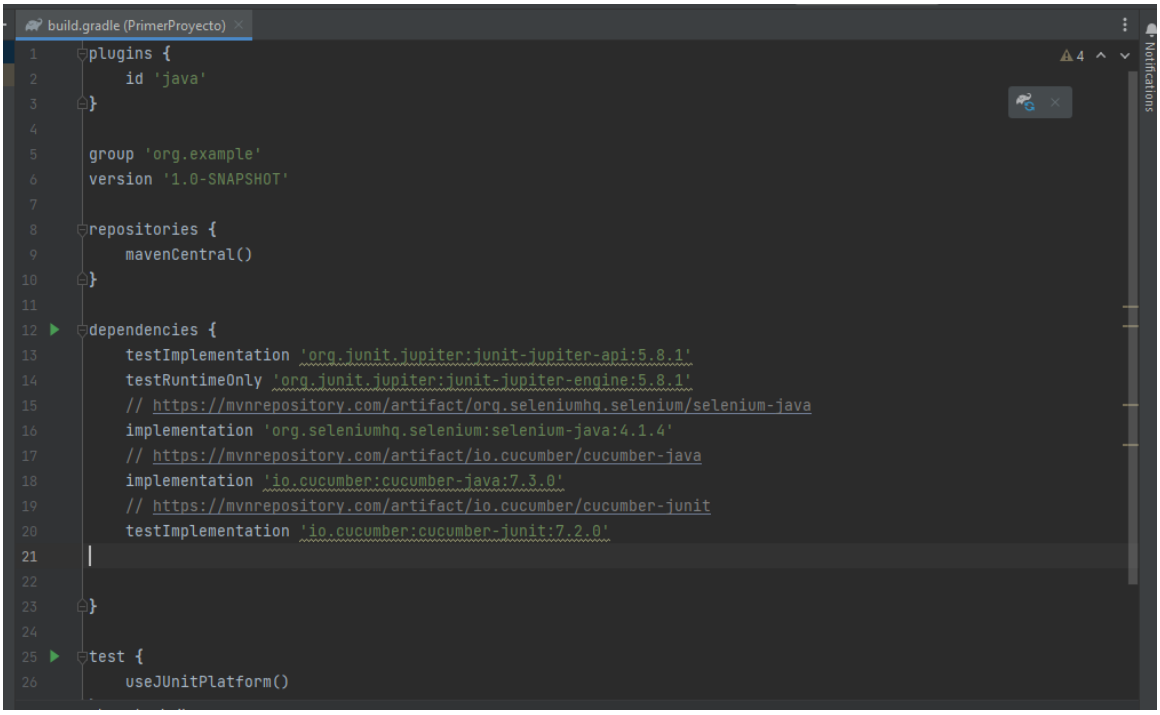
1. **Dependencia de Selenium:** Copiar en el archivo “build.gradle” del proyecto la dependencia para Selenium. Eso lo obtenemos desde la página de Maven repositorio:

   Siempre hay que descargar la última versión y la más estable.



2. **Dependencia de Cucumber:** Copiar en el archivo “build. gradle” del proyecto la dependencia para Cucumber. Eso lo obtenemos desde la página de Maven repositorio:

   Siempre hay que descargar la última versión y la más estable.[3]

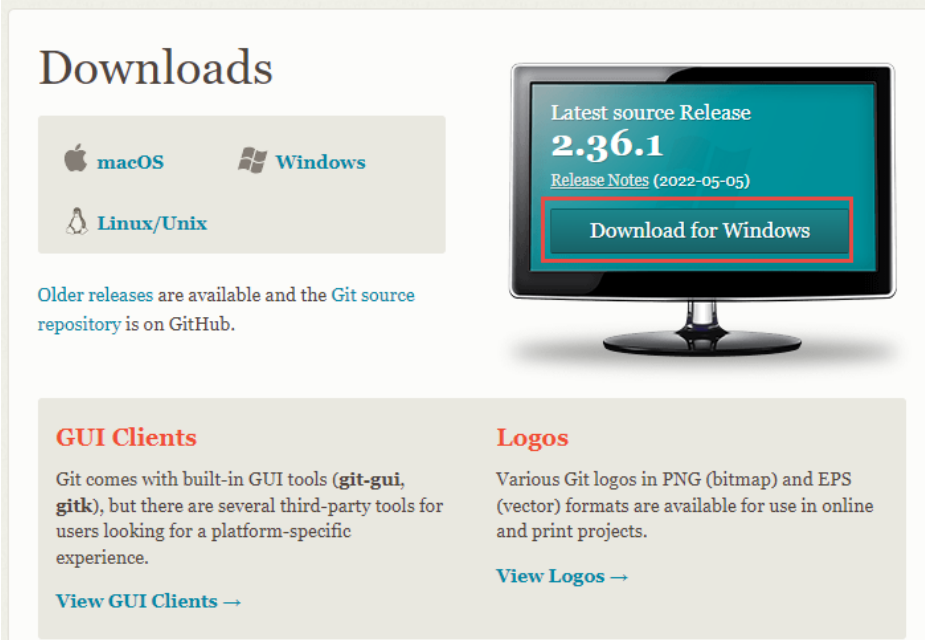


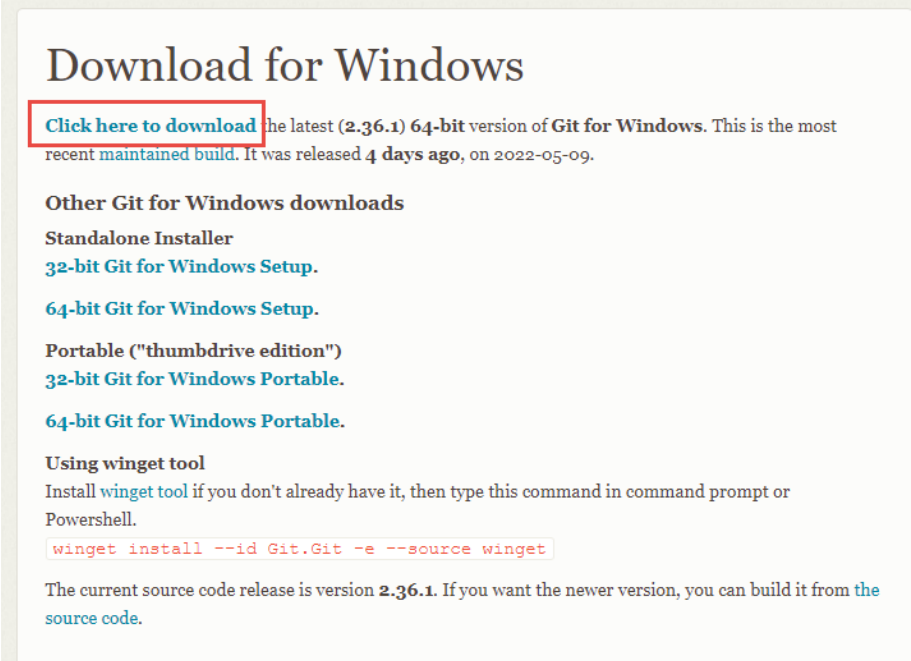
[3] Se procede de la misma manera para agregar cualquier dependencia externa al proyecto. Gradle lo descargará de manera automática y estará listo para usar en nuestro proyecto.

6 Descargar, Instalar Git

6.1 Descargar Git

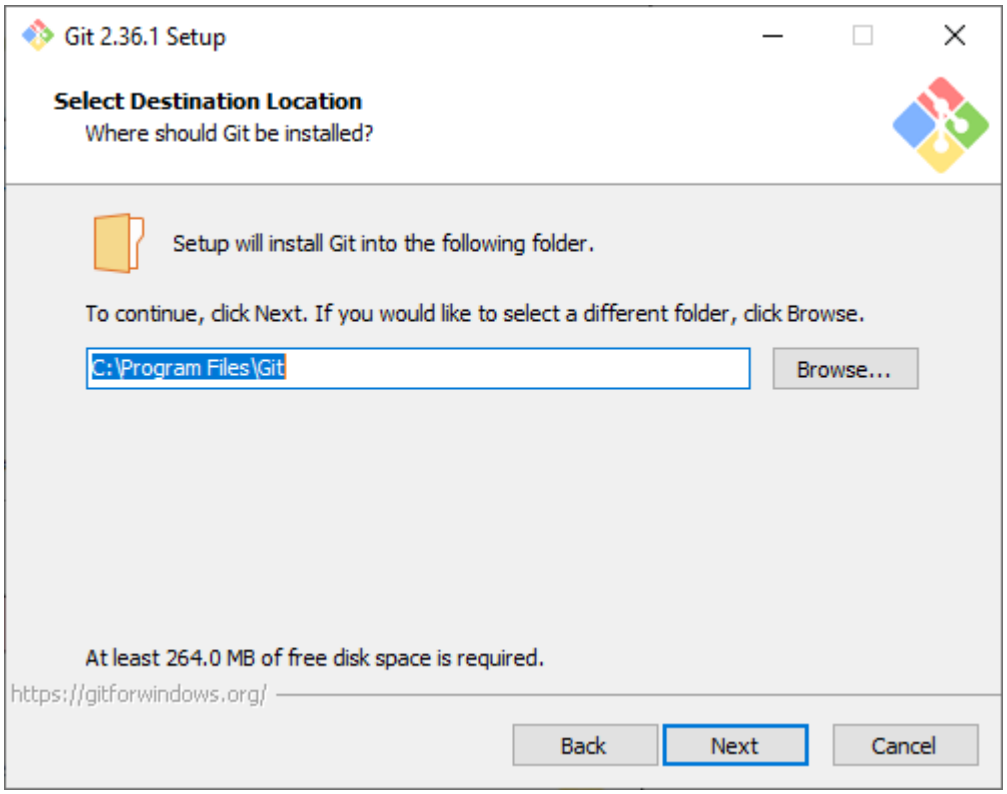
3. Descargar Git desde la misma página: https://git-scm.com/download/win. Clic en el botón descargar.

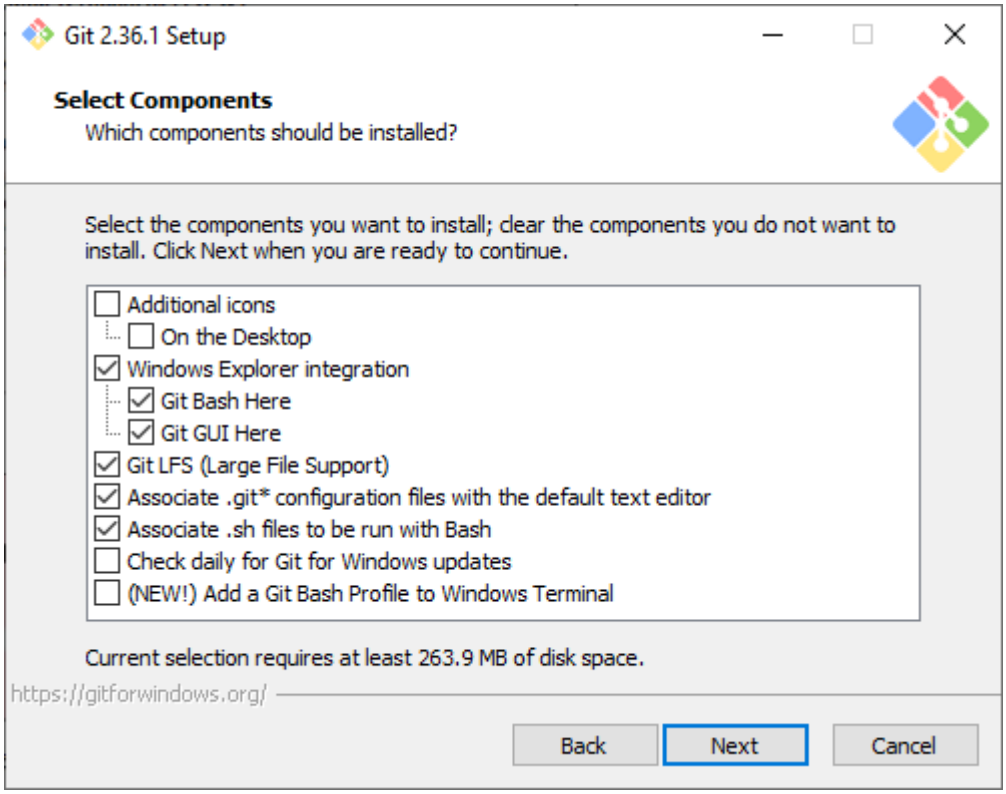


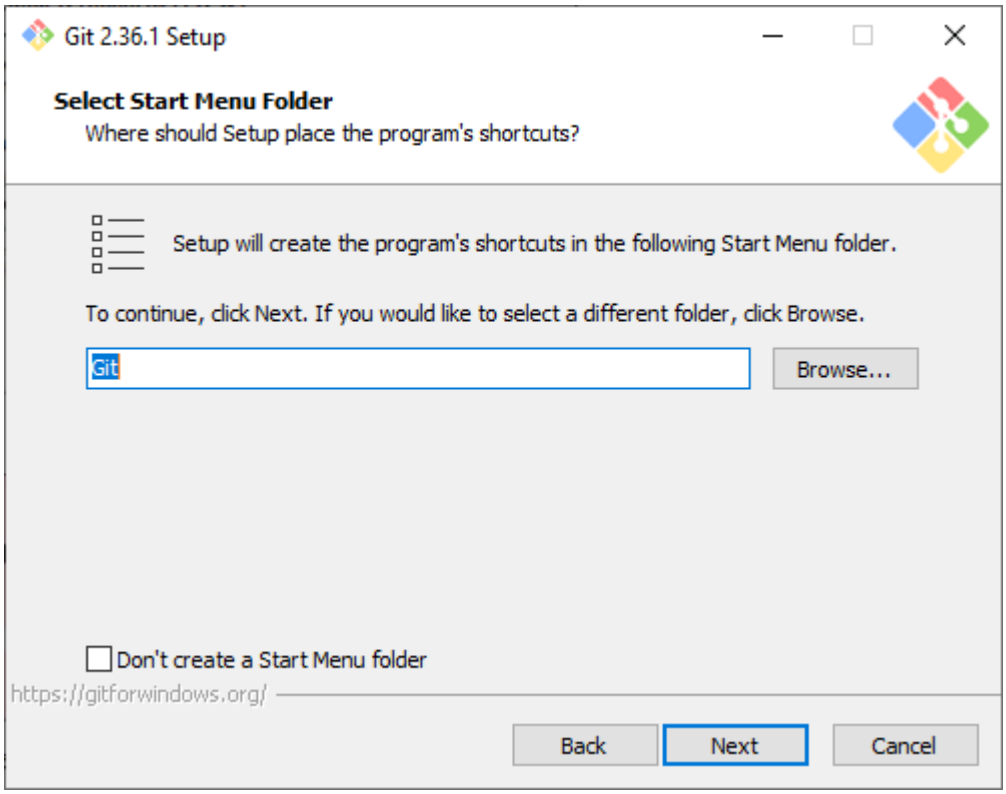


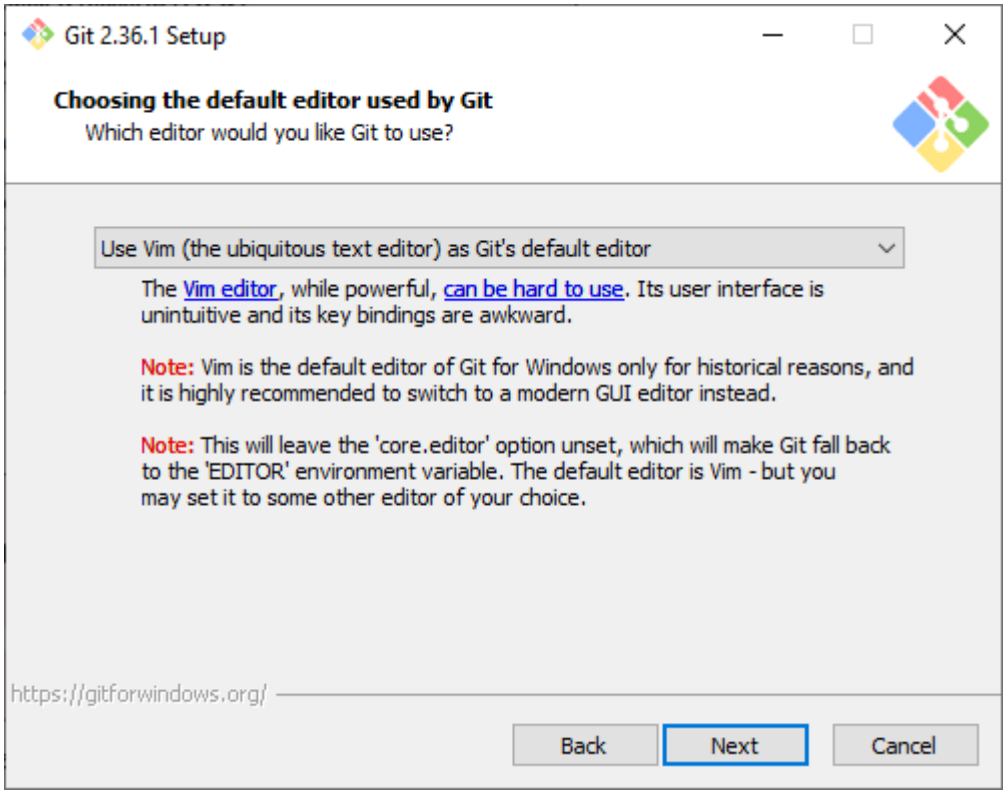
6.2 **Instalar Git**

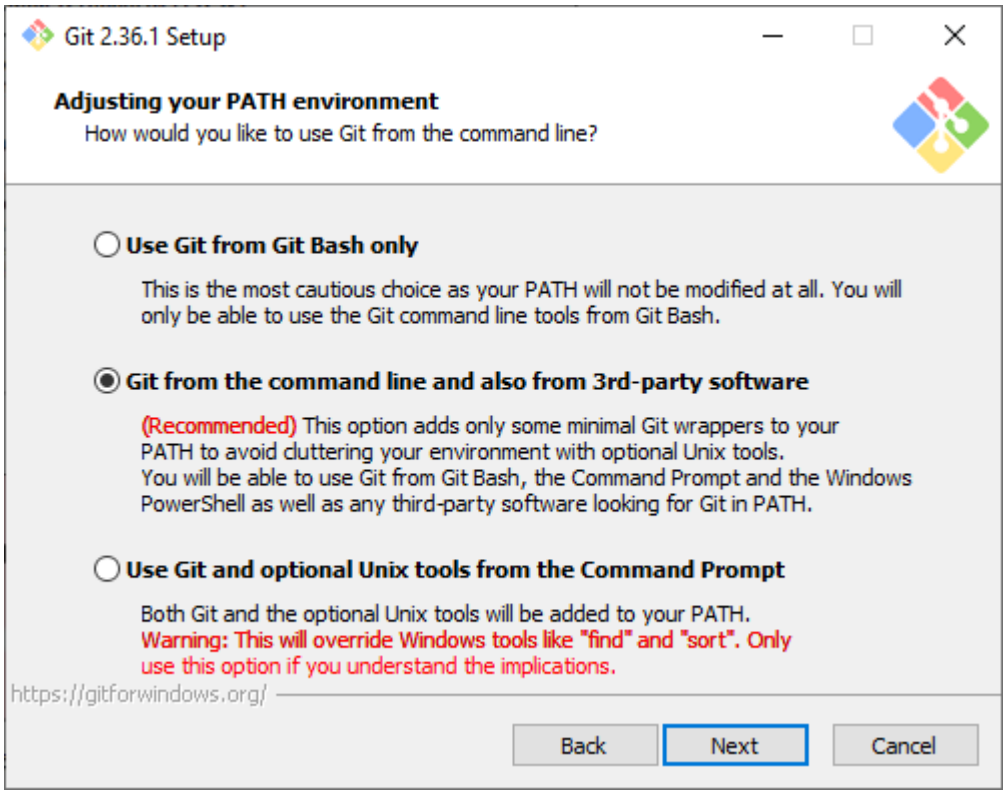
1. Proceso de Instalación de Git

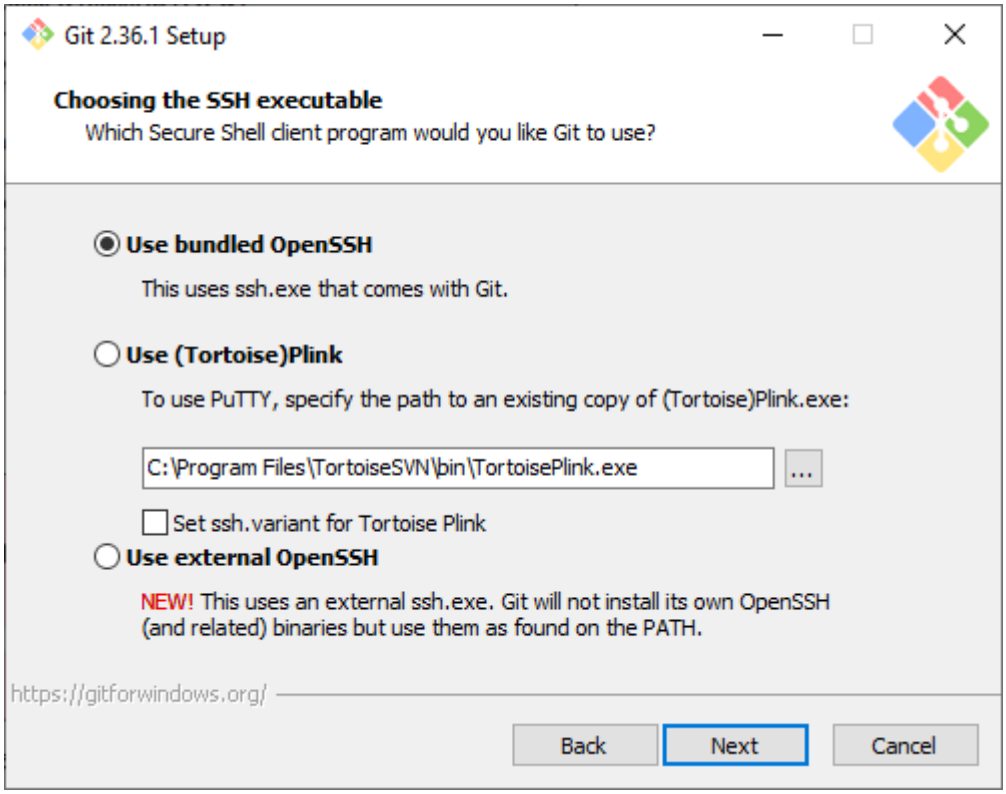


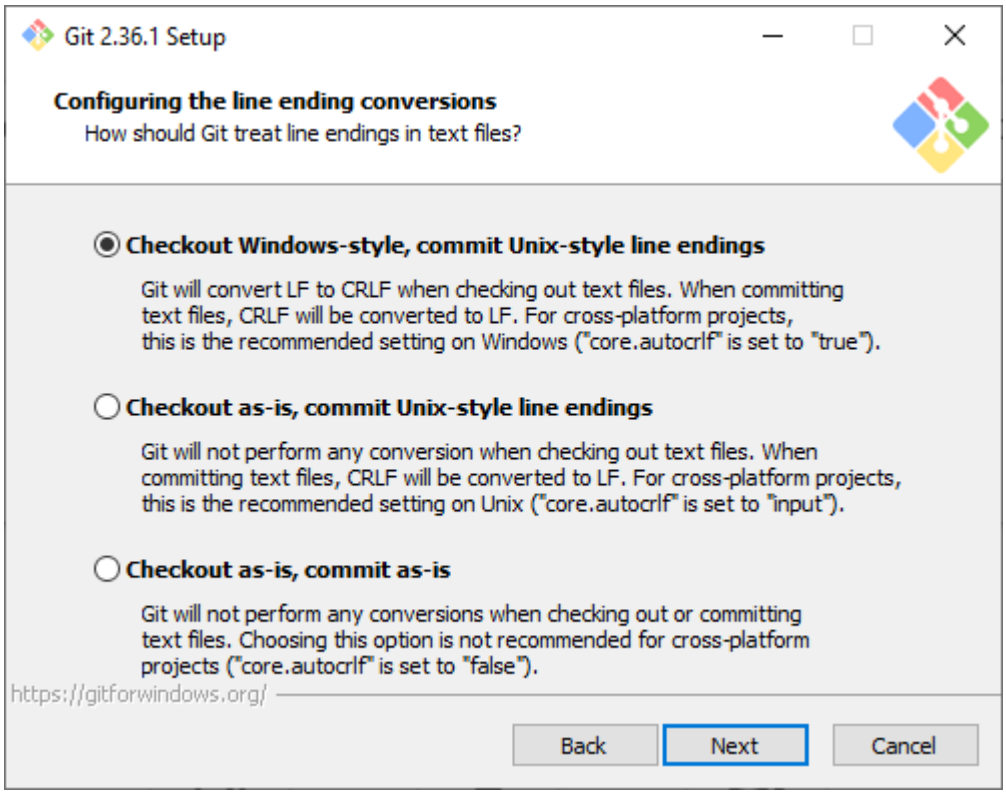

Git 2.36.1 Setup
Select Components
Which components should be installed?
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.
Additional icons
On the Desktop
Windows Explorer integration
Git Bash Here
Git GUI Here
Git LFS (Large File Support)
Associate .git* configuration files with the default text editor
Associate .sh files to be run with Bash
Check daily for Git for Windows updates
(NEW!) Add a Git Bash Profile to Windows Terminal
Current selection requires at least 263.9 MB of disk space.
https://gitforwindows.org/
Back
Next
Cancel

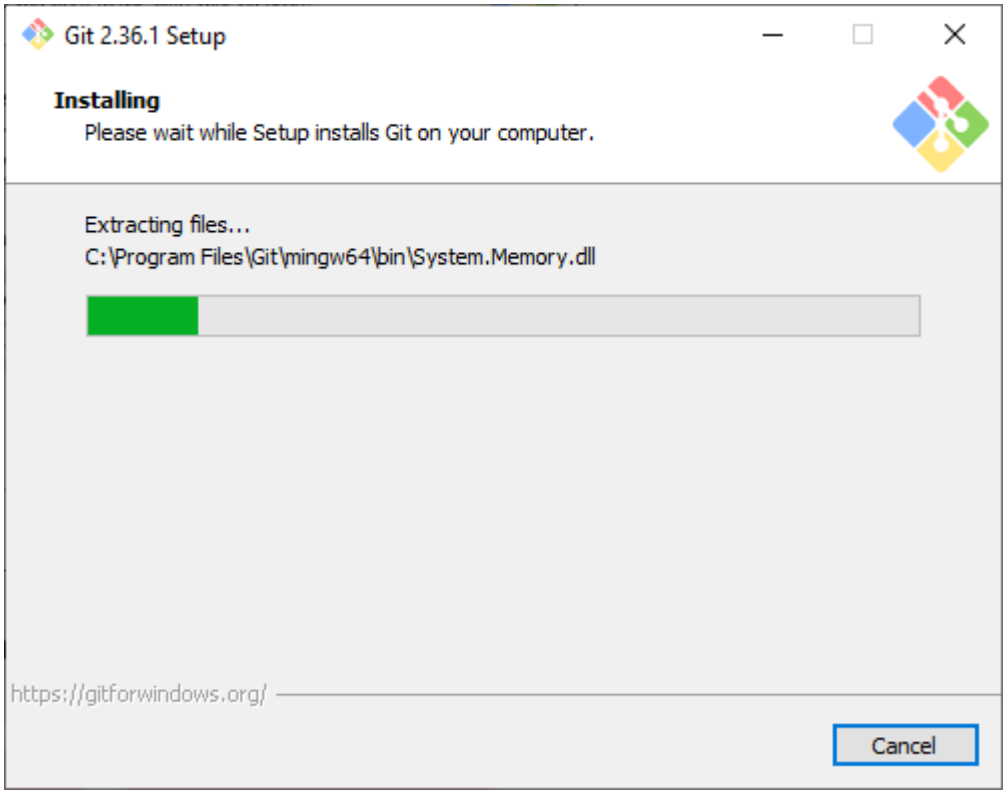

Git 2.36.1 Setup
Select Start Menu Folder
Where should Setup place the program's shortcuts?
Setup will create the program's shortcuts in the following Start Menu folder.
To continue, click Next. If you would like to select a different folder, click Browse.
Git
Browse...
Don't create a Start Menu folder
https://gitforwindows.org/
Back
Next
Cancel

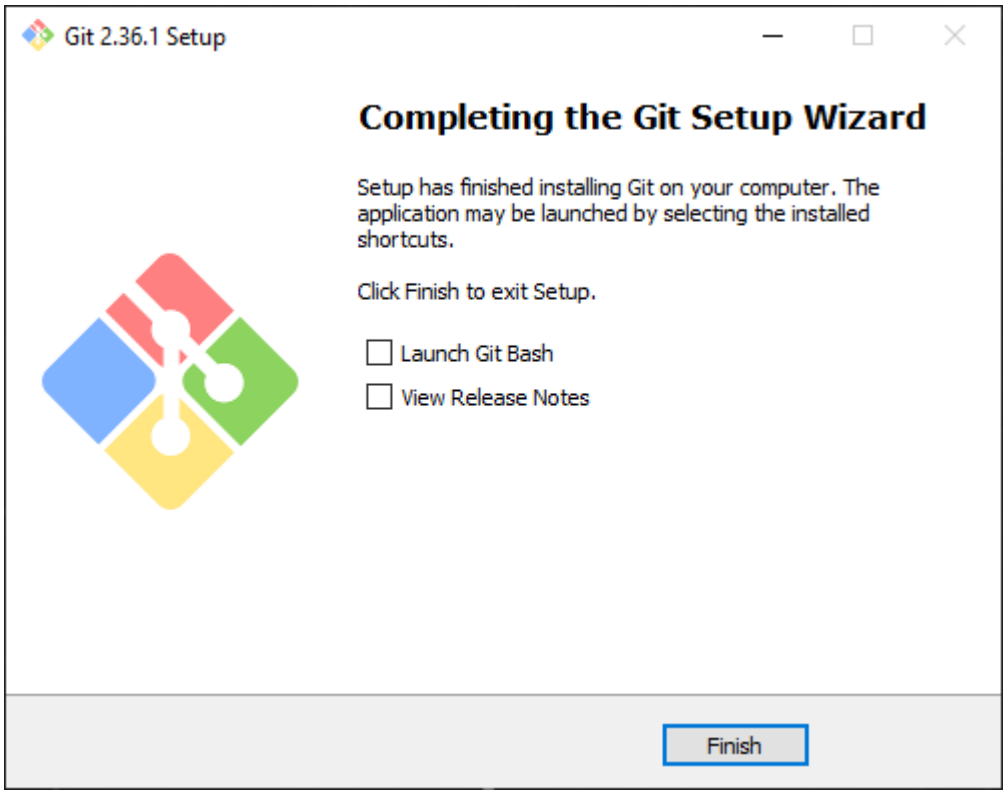

Git 2.36.1 Setup
Choosing the default editor used by Git
Which editor would you like Git to use?
Use Vim (the ubiquitous text editor) as Git's default editor
The Vim editor, while powerful, can be hard to use. Its user interface is unintuitive and its key bindings are awkward.
Note: Vim is the default editor of Git for Windows only for historical reasons, and it is highly recommended to switch to a modern GUI editor instead.
Note: This will leave the 'core.editor' option unset, which will make Git fall back to the 'EDITOR' environment variable. The default editor is Vim - but you may set it to some other editor of your choice.
https://gitforwindows.org/
Back
Next
Cancel


Git 2.36.1 Setup
Adjusting your PATH environment
How would you like to use Git from the command line?
Use Git from Git Bash only
This is the most cautious choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.
Git from the command line and also from 3rd-party software
(Recommended) This option adds only some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools.
You will be able to use Git from Git Bash, the Command Prompt and the Windows PowerShell as well as any third-party software looking for Git in PATH.
Use Git and optional Unix tools from the Command Prompt
Both Git and the optional Unix tools will be added to your PATH.
Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.
https://gitforwindows.org/
Back
Next
Cancel


Git 2.36.1 Setup
Choosing the SSH executable
Which Secure Shell client program would you like Git to use?
Use bundled OpenSSH
This uses ssh.exe that comes with Git.
Use (Tortoise)Plink
To use PuTTY, specify the path to an existing copy of (Tortoise)Plink.exe:
C:\Program Files\TortoiseSVN\bin\TortoisePlink.exe
...
Set ssh.variant for Tortoise Plink
Use external OpenSSH
NEW! This uses an external ssh.exe. Git will not install its own OpenSSH (and related) binaries but use them as found on the PATH.
https://gitforwindows.org/
Back
Next
Cancel

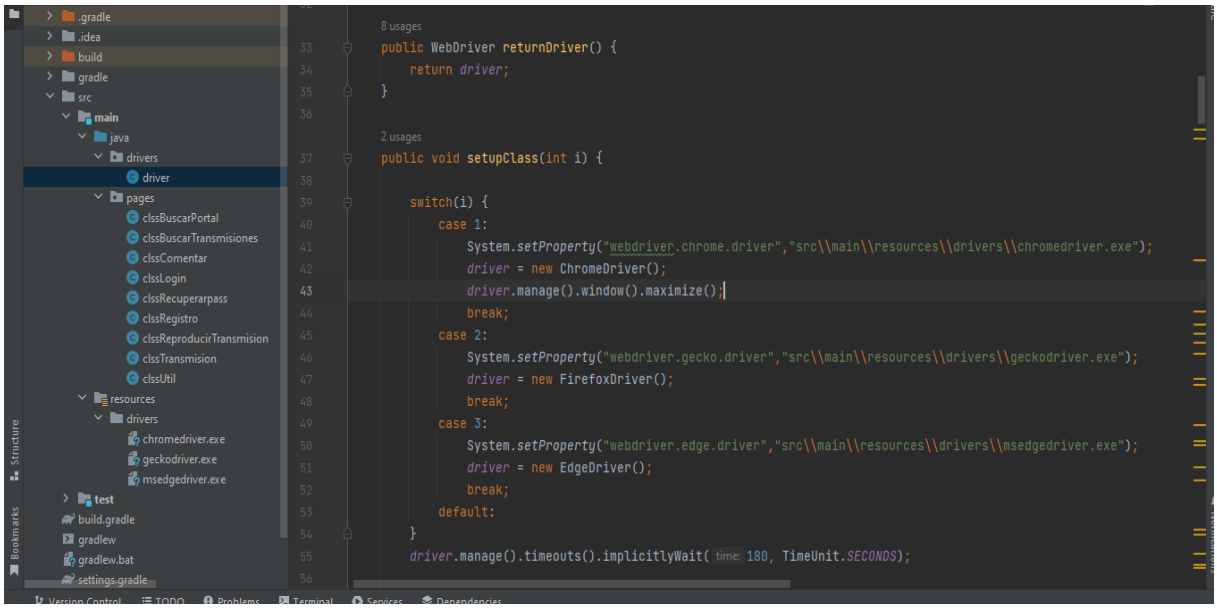

Git 2.36.1 Setup
Configuring the line ending conversions
How should Git treat line endings in text files?
Checkout Windows-style, commit Unix-style line endings
Git will convert LF to CRLF when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Windows ("core.autocrlf" is set to "true").
Checkout as-is, commit Unix-style line endings
Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").
Checkout as-is, commit as-is
Git will not perform any conversions when checking out or committing text files. Choosing this option is not recommended for cross-platform projects ("core.autocrlf" is set to "false").
https://gitforwindows.org/
Back
Next
Cancel


Git 2.36.1 Setup
Installing
Please wait while Setup installs Git on your computer.
Extracting files...
C:\Program Files\Git\mingw64\bin\System.Memory.dll
https://gitforwindows.org/
Cancel

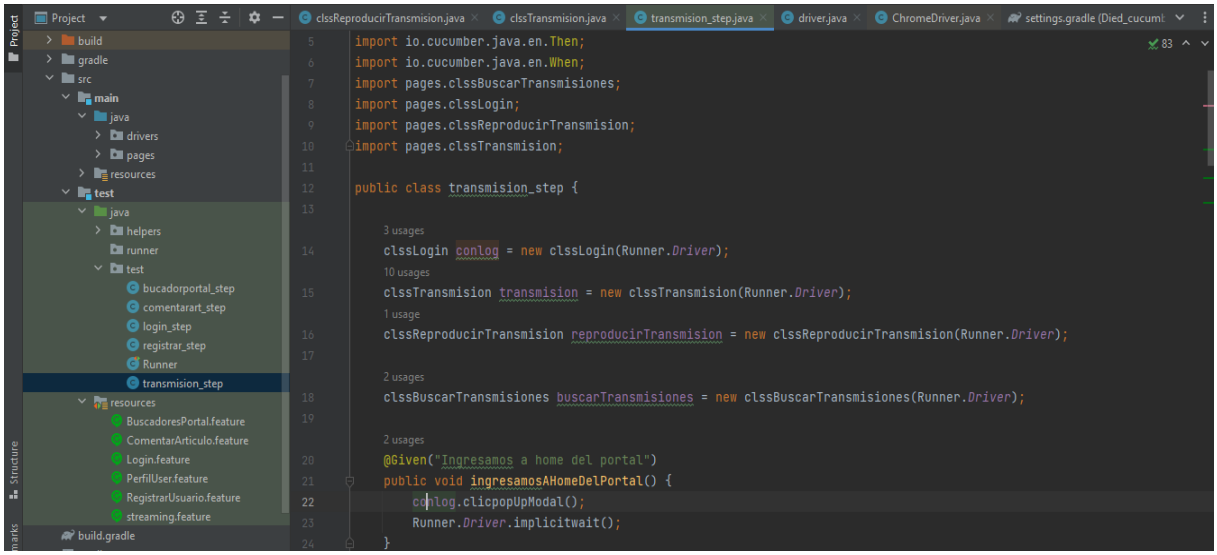

Git 2.36.1 Setup
Completing the Git Setup Wizard
Setup has finished installing Git on your computer. The application may be launched by selecting the installed shortcuts.
Click Finish to exit Setup.
Launch Git Bash
View Release Notes
Finish

Anexo 2 *Estructura del Patrón de Diseño de Page Object Model – POM*

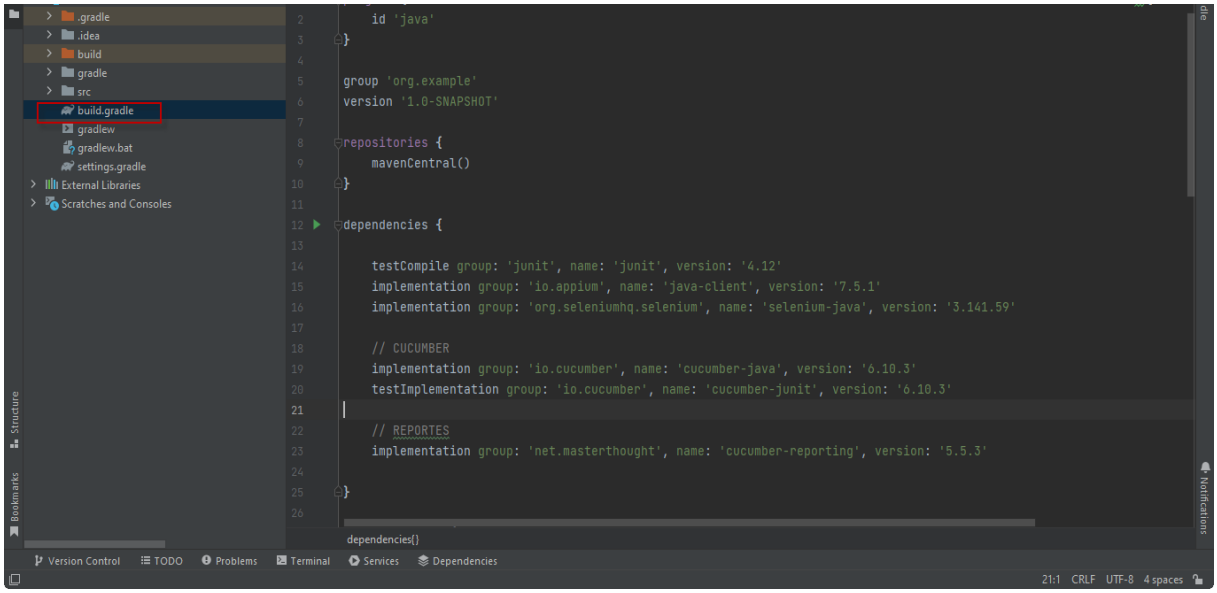
Estructura de objetos Main



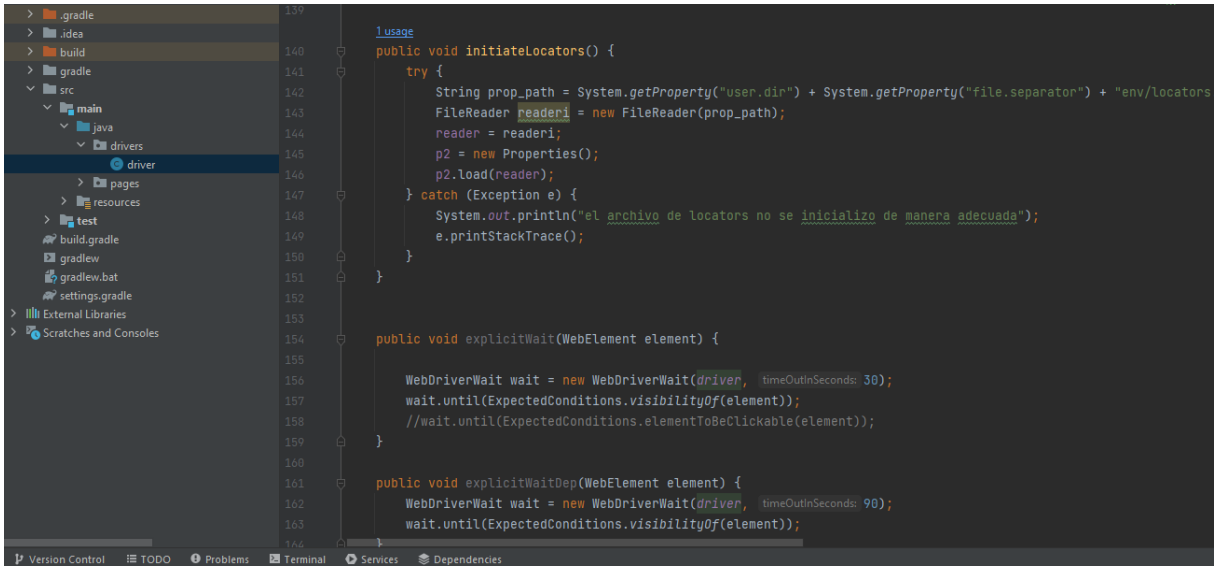
Estructura de Objetos Test



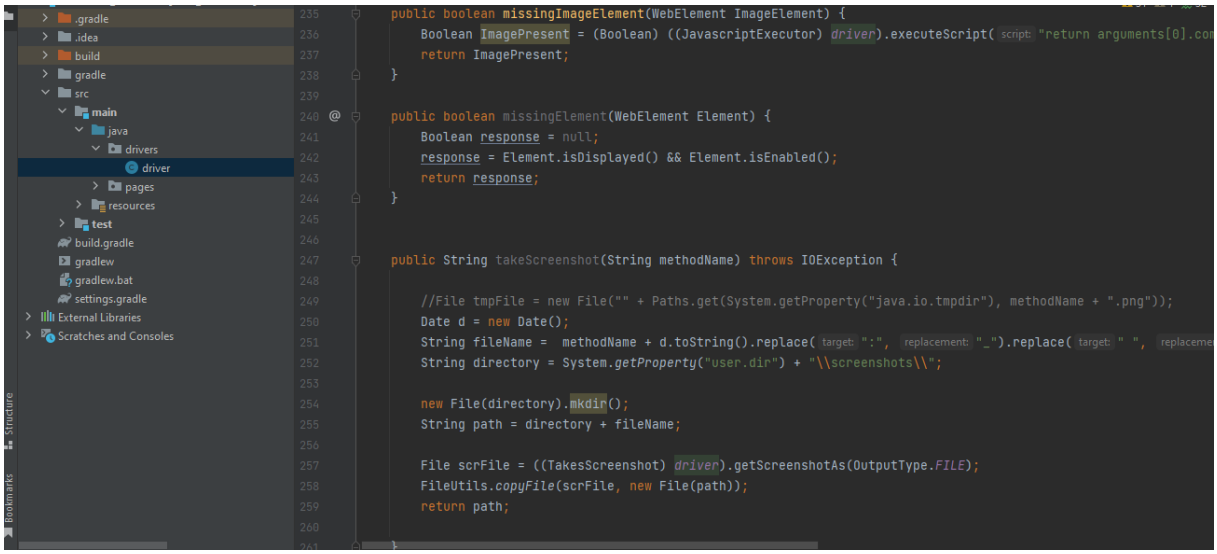
Anexo 3 *Manejo de dependencias java con Gradle*



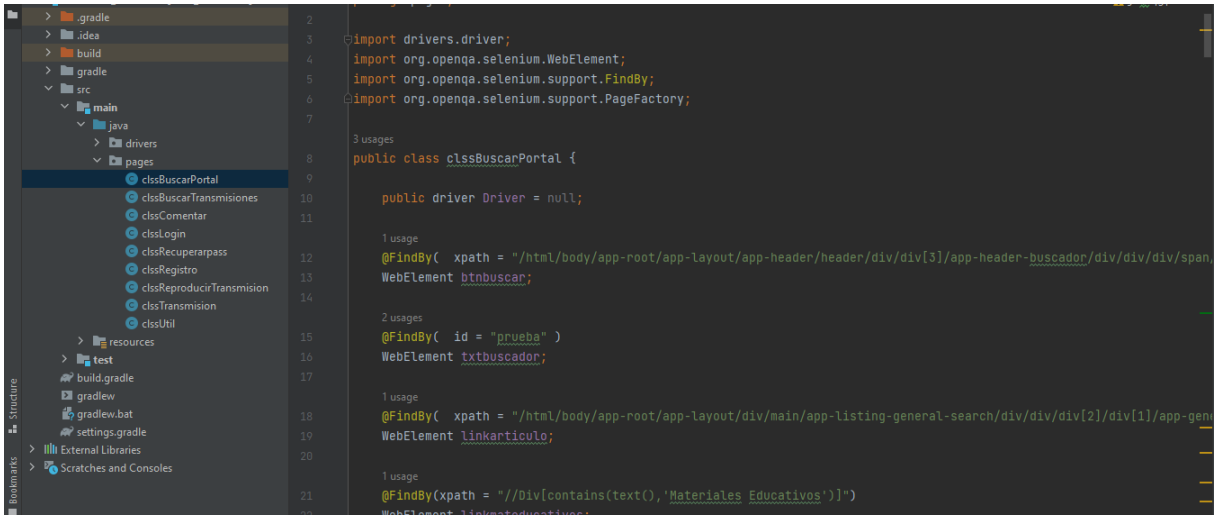
Anexo 4 *Código de ejecución multiplataforma*


.gradle
.idea
build
gradle
src
main
java
drivers
driver
pages
resources
test
build.gradle
gradlew
gradlew.bat
settings.gradle
External Libraries
Scratches and Consoles
1 usage
public void initiateLocators() {
try {
String prop_path = System.getProperty("user.dir") + System.getProperty("file.separator") + "env/locators
FileReader readeri = new FileReader(prop_path);
reader = readeri;
p2 = new Properties();
p2.load(reader);
} catch (Exception e) {
System.out.println("el archivo de locators no se inicializo de manera adecuada");
e.printStackTrace();
}
}
public void explicitWait(WebElement element) {
WebDriverWait wait = new WebDriverWait(driver, timeOutInSeconds: 30);
wait.until(ExpectedConditions.visibilityOf(element));
//wait.until(ExpectedConditions.elementToBeClickable(element));
}
public void explicitWaitDep(WebElement element) {
WebDriverWait wait = new WebDriverWait(driver, timeOutInSeconds: 90);
wait.until(ExpectedConditions.visibilityOf(element));
Version Control
TODO
Problems
Terminal
Services
Dependencies


.gradle
.idea
build
gradle
src
main
java
drivers
driver
pages
resources
test
build.gradle
gradlew
gradlew.bat
settings.gradle
External Libraries
Scratches and Consoles
1 usage
public void initiateLocators() {
try {
String prop_path = System.getProperty("user.dir") + System.getProperty("file.separator") + "env/locator
FileReader readeri = new FileReader(prop_path);
reader = readeri;
p2 = new Properties();
p2.load(reader);
} catch (Exception e) {
System.out.println("el archivo de locators no se inicializo de manera adecuada");
e.printStackTrace();
}
}
public void explicitWait(WebElement element) {
WebDriverWait wait = new WebDriverWait(driver, timeOutInSeconds: 30);
wait.until(ExpectedConditions.visibilityOf(element));
//wait.until(ExpectedConditions.elementToBeClickable(element));
}
public void explicitWaitDep(WebElement element) {
WebDriverWait wait = new WebDriverWait(driver, timeOutInSeconds: 90);
wait.until(ExpectedConditions.visibilityOf(element));

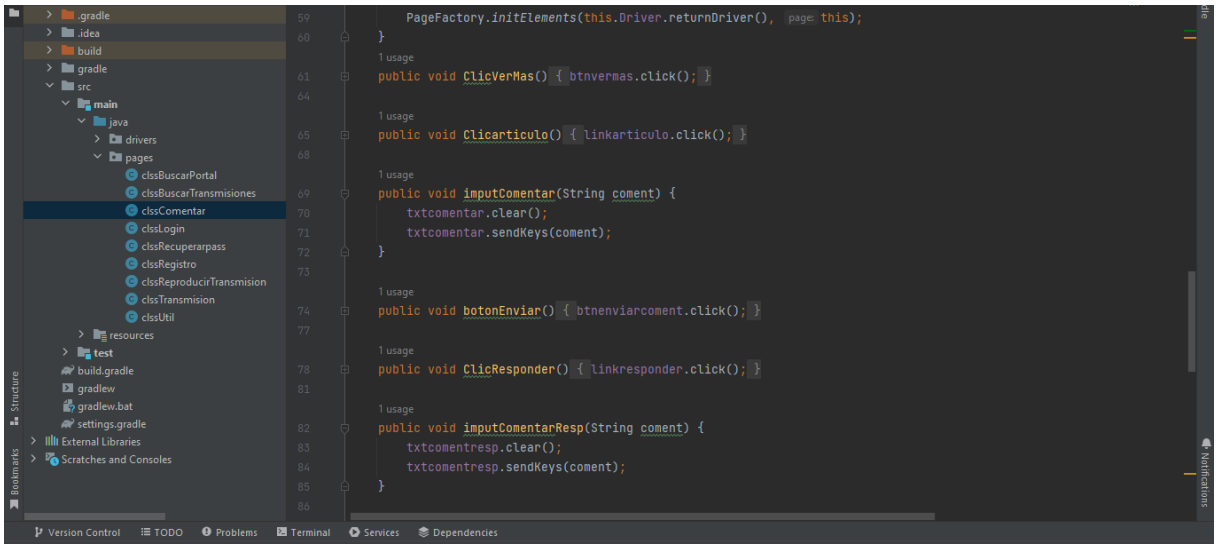

.gradle
.idea
build
gradle
src
main
java
drivers
driver
pages
resources
test
build.gradle
gradlew
gradlew.bat
settings.gradle
External Libraries
Scratches and Consoles
public boolean missingImageElement(WebElement ImageElement) {
Boolean ImagePresent = (Boolean) ((JavascriptExecutor) driver).executeScript(script: "return arguments[0].com
return ImagePresent;
}
public boolean missingElement(WebElement Element) {
Boolean response = null;
response = Element.isDisplayed() && Element.isEnabled();
return response;
}
public String takeScreenshot(String methodName) throws IOException {
//File tmpFile = new File("" + Paths.get(System.getProperty("java.io.tmpdir"), methodName + ".png"));
Date d = new Date();
String fileName = methodName + d.toString().replace(target: ":", replacement: "_").replace(target: " ",
String directory = System.getProperty("user.dir") + "\\screenshots\\";
new File(directory).mkdir();
String path = directory + fileName;
File scrFile = ((TakesScreenshot) driver).getScreenshotAs(OutputType.FILE);
FileUtils.copyFile(scrFile, new File(path));
return path;
Structure
Bookmarks

Anexo 5 *Lista y código de los Pages de automatización*

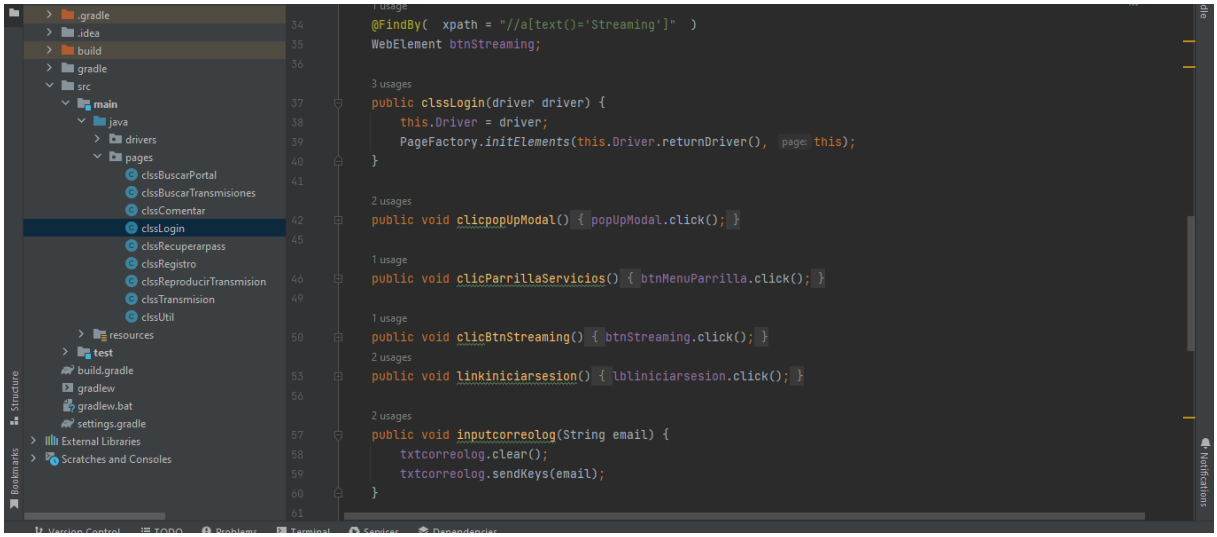


```
import drivers.driver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;

3 usages
public class clssBuscarPortal {

    public driver Driver = null;

    1 usage
    @FindBy(  xpath = "/html/body/app-root/app-layout/app-header/header/div/div[3]/app-header-buscador/div/div/div/span
    WebElement btnbuscar;

    2 usages
    @FindBy(  id = "prueba" )
    WebElement txtbuscador;

    1 usage
    @FindBy(  xpath = "/html/body/app-root/app-layout/div/main/app-listing-general-search/div/div/div[2]/div[1]/app-gen
    WebElement linkarticulo;

    1 usage
    @FindBy(xpath = "//Div[contains(text(),'Materiales Educativos')]")
```

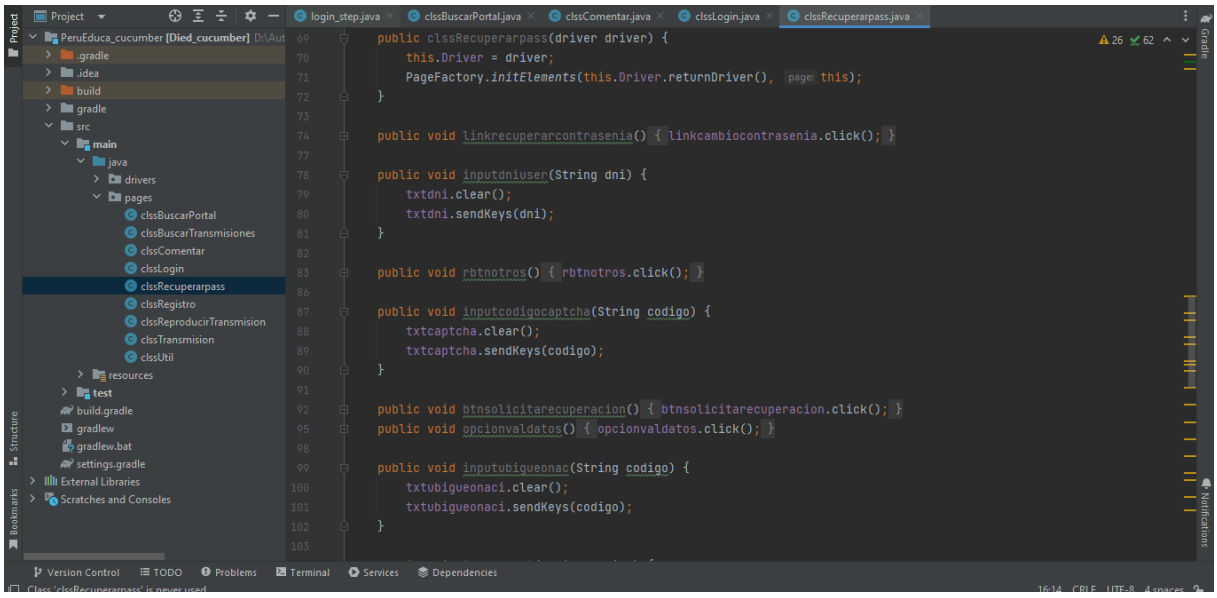
```
    1 usage
    @FindBy(xpath = "/html/body/app-root/app-layout/div/main/app-streaming/app-search-card/div[1]/div/div[2]/div/div/for
    WebElement cboDireccion;
    1 usage
    @FindBy(css = "div>div.play")
    List<WebElement> lstDireccion;
    1 usage
    @FindBy(xpath = "//button[@class='btn secondary']")
    WebElement btnBuscar;


    1 usage
    public clssBuscarTransmisiones(driver driver) {
        this.Driver = driver;
        PageFactory.initElements(this.Driver.returnDriver(),  page: this);
    }

    1 usage
    public void clicCboDireccion(String direccion) {
        cboDireccion.click();
        for (WebElement e : lstDireccion) {
            if (e.getText().equals(direccion)) {
                e.click();
                break;
            }
        }
```



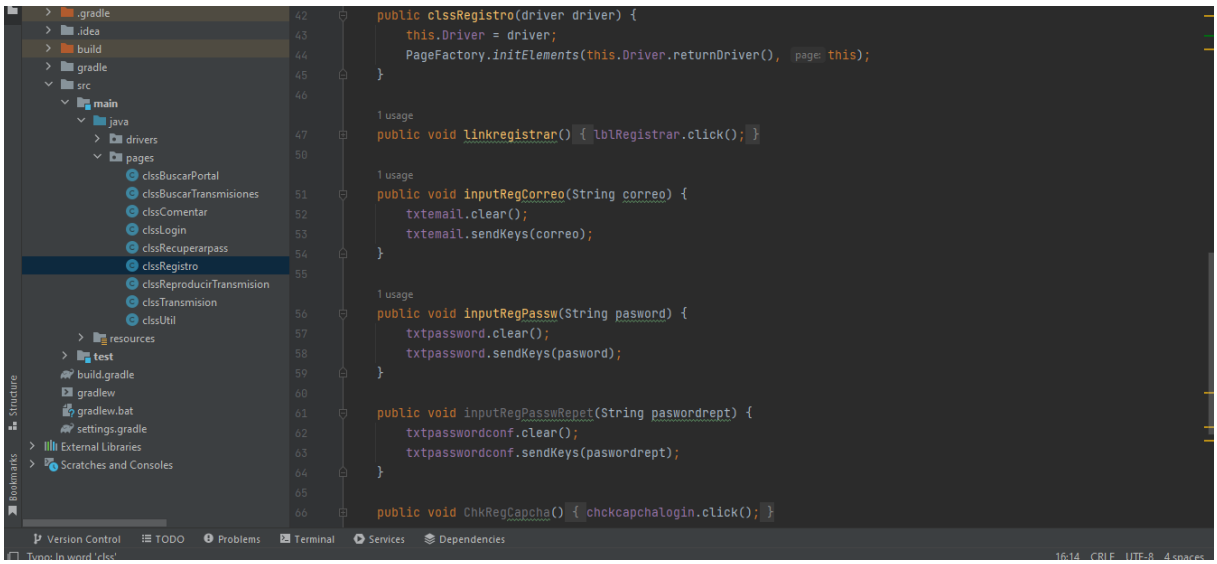
```
        PageFactory.initElements(this.Driver.returnDriver(),  page: this);
    }
    1 usage
    public void ClicVerMas() { btnvermas.click(); }

    1 usage
    public void Clicarticulo() { linkarticulo.click(); }

    1 usage
    public void imputComentar(String coment) {
        txtcomentar.clear();
        txtcomentar.sendKeys(coment);
    }

    1 usage
    public void botonEnviar() { btnenviarcoment.click(); }

    1 usage
    public void ClicResponder() { linkresponder.click(); }

    1 usage
    public void imputComentarResp(String coment) {
        txtcomentresp.clear();
        txtcomentresp.sendKeys(coment);
    }
```



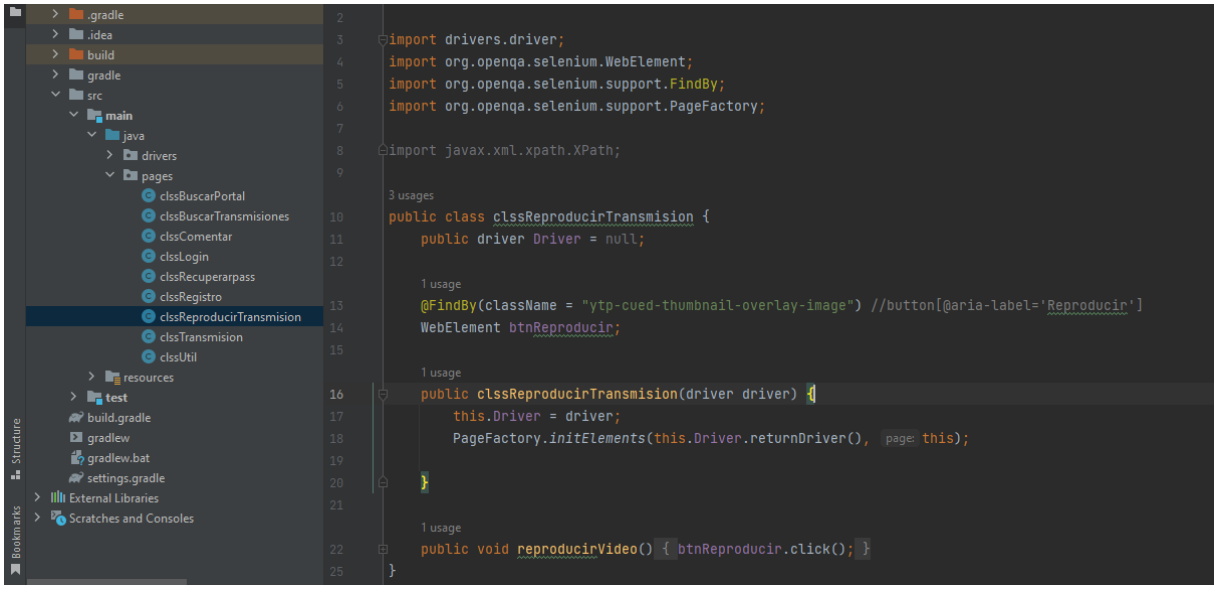
```
@FindBy(  xpath = "//a[text()='Streaming']"  )
WebElement btnStreaming;

3 usages
public clssLogin(driver driver) {
    this.Driver = driver;
    PageFactory.initElements(this.Driver.returnDriver(),  page: this);
}

2 usages
public void clicpopUpModal() { popUpModal.click(); }

1 usage
public void clicParrillaServicios() { btnMenuParrilla.click(); }

1 usage
public void clicBtnStreaming() { btnStreaming.click(); }
2 usages
public void linkiniciarsesion() { lbliniciarsesion.click(); }

2 usages
public void inputcorreolog(String email) {
    txtcorreolog.clear();
    txtcorreolog.sendKeys(email);
}
```

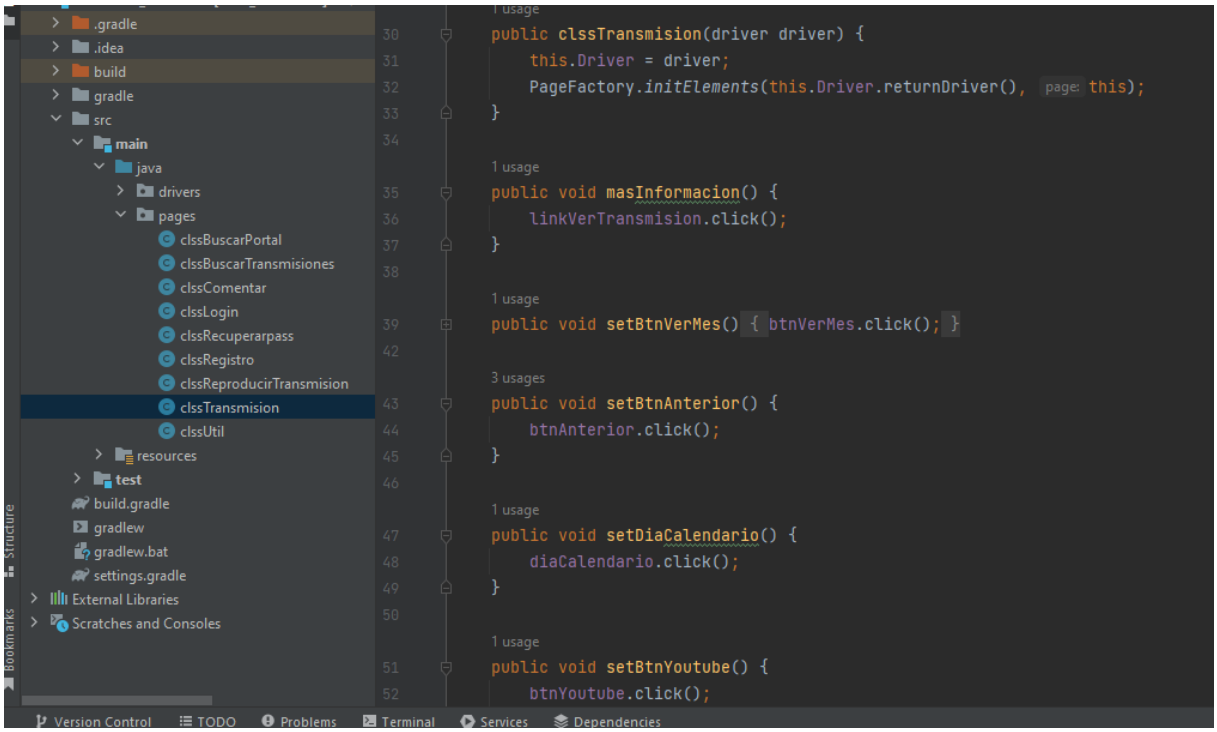


```
public clssRecuperarpass(driver driver) {
    this.Driver = driver;
    PageFactory.initElements(this.Driver.returnDriver(),  page: this);
}

public void linkrecuperarcontrasenia() { linkcambiocontrasenia.click(); }

public void inputdniuser(String dni) {
    txtdni.clear();
    txtdni.sendKeys(dni);
}

public void rbtnotros() { rbtnotros.click(); }

public void inputcodigocaptcha(String codigo) {
    txtcaptcha.clear();
    txtcaptcha.sendKeys(codigo);
}

public void btnsolicitarecuperacion() { btnsolicitarecuperacion.click(); }
public void opcionvaldatos() { opcionvaldatos.click(); }

public void inputubigueonac(String codigo) {
    txtubigueonaci.clear();
    txtubigueonaci.sendKeys(codigo);
}
```

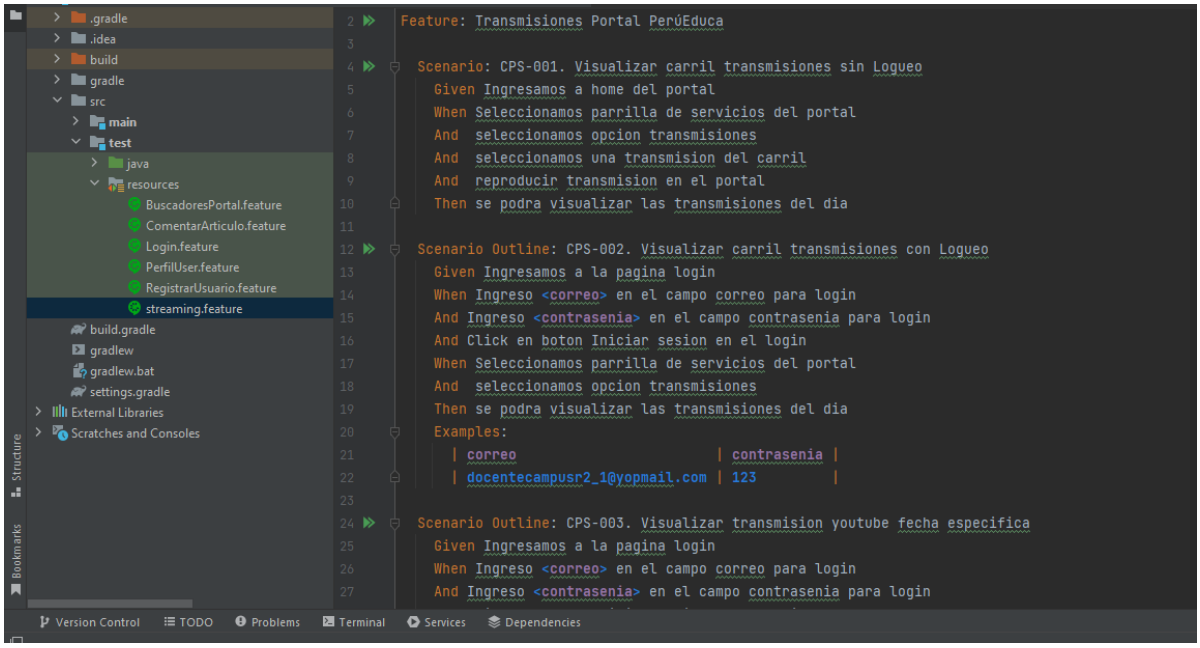


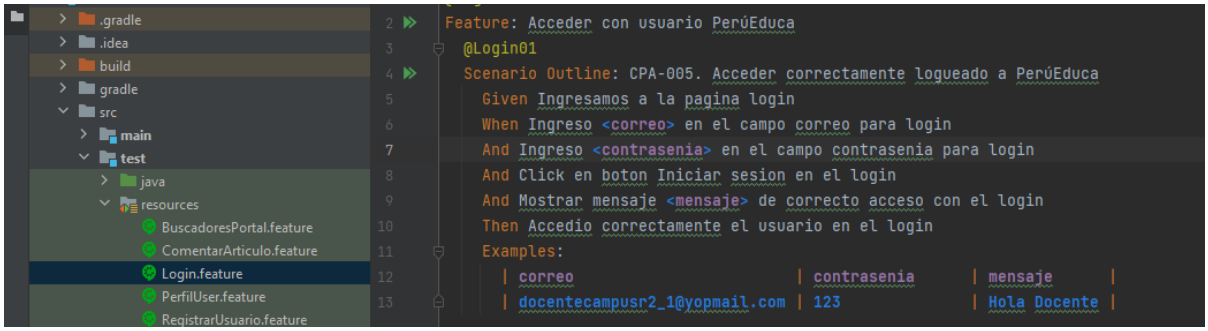
```
public clssRegistro(driver driver) {
    this.Driver = driver;
    PageFactory.initElements(this.Driver.returnDriver(),  page: this);
}

1 usage
public void linkregistrar() { lblRegistrar.click(); }

1 usage
public void inputRegCorreo(String correo) {
    txtemail.clear();
    txtemail.sendKeys(correo);
}

1 usage
public void inputRegPassw(String pasword) {
    txtpassword.clear();
    txtpassword.sendKeys(pasword);
}

public void inputRegPasswRepet(String paswordrept) {
    txtpasswordconf.clear();
    txtpasswordconf.sendKeys(paswordrept);
}

public void ChkRegCapcha() { chckcapchalogin.click(); }
```

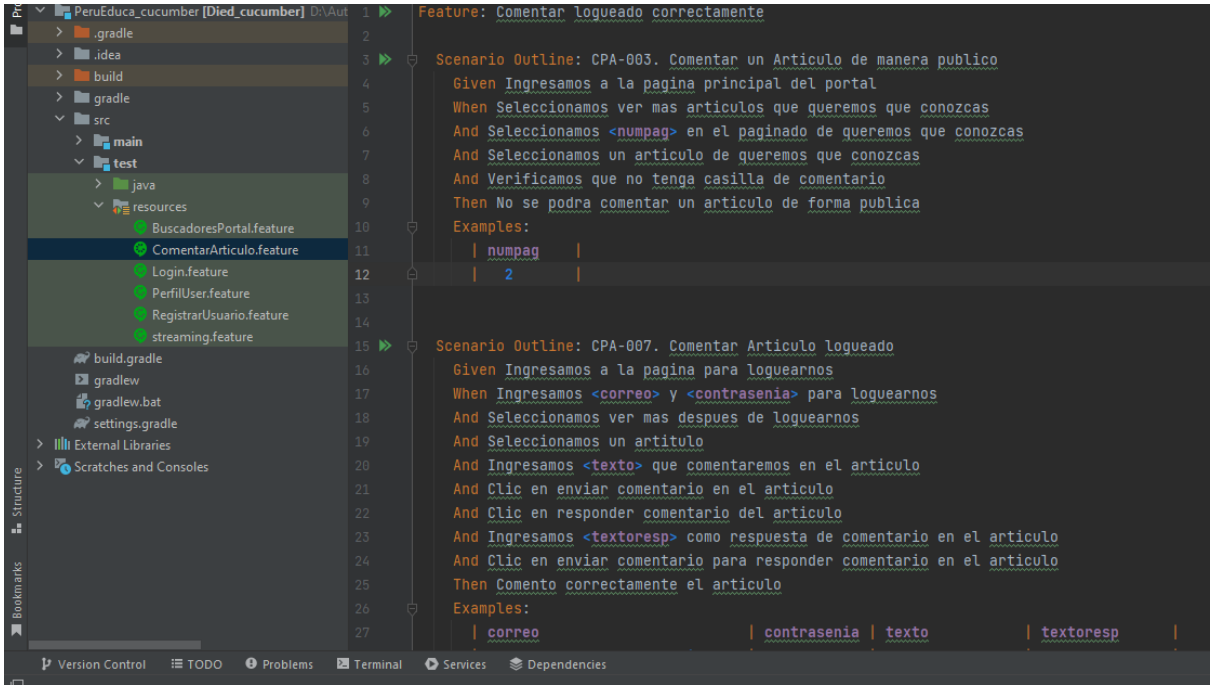

.gradle
.idea
build
gradle
src
main
java
drivers
pages
clssBuscarPortal
clssBuscarTransmisiones
clssComentar
clssLogin
clssRecuperarpass
clssRegistro
clssReproducirTransmision
clssTransmision
clssUtil
resources
test
build.gradle
gradlew
gradlew.bat
settings.gradle
External Libraries
Scratches and Consoles
Structure
Bookmarks
import drivers.driver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;
import javax.xml.xpath.XPath;
3 usages
public class clssReproducirTransmision {
public driver Driver = null;
1 usage
@FindBy(className = "ytp-cued-thumbnail-overlay-image") //button[@aria-label='Reproducir']
WebElement btnReproducir;
1 usage
public clssReproducirTransmision(driver driver) {
this.Driver = driver;
PageFactory.initElements(this.Driver.returnDriver(), page: this);
}
1 usage
public void reproducirVideo() { btnReproducir.click(); }
}

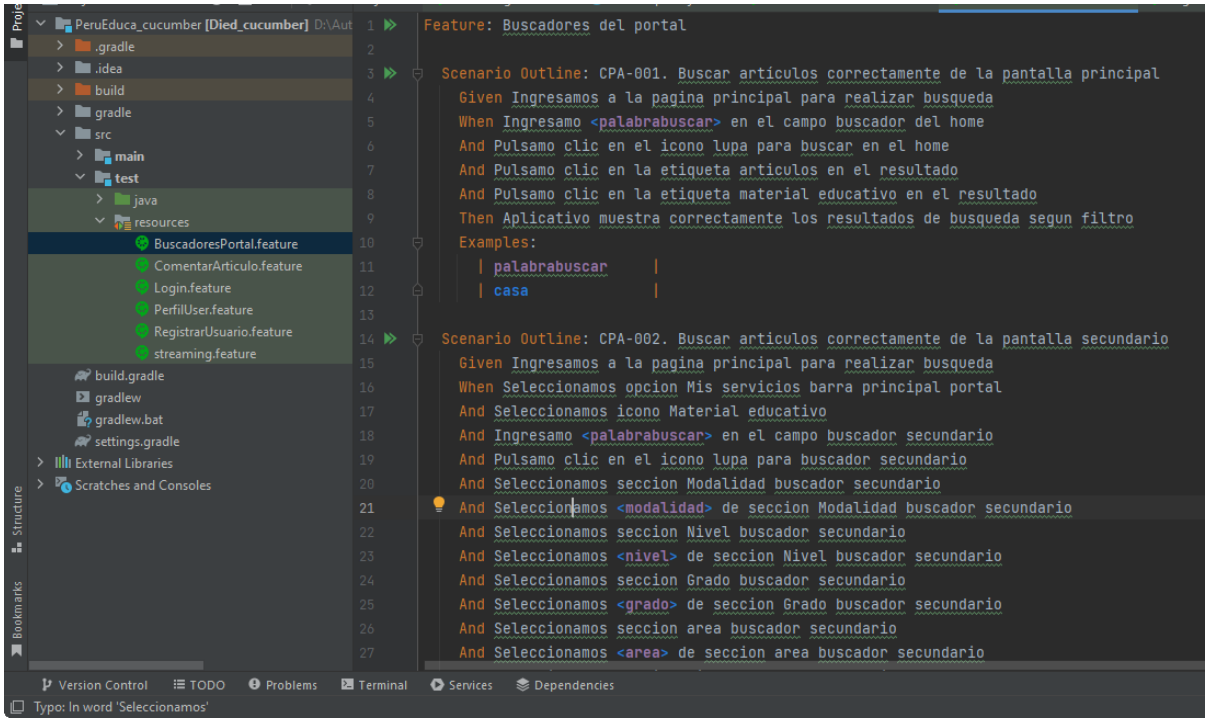

.gradle
.idea
build
gradle
src
main
java
drivers
pages
clssBuscarPortal
clssBuscarTransmisiones
clssComentar
clssLogin
clssRecuperarpass
clssRegistro
clssReproducirTransmision
clssTransmision
clssUtil
resources
test
build.gradle
gradlew
gradlew.bat
settings.gradle
External Libraries
Scratches and Consoles
Structure
Bookmarks
1 usage
public clssTransmision(driver driver) {
this.Driver = driver;
PageFactory.initElements(this.Driver.returnDriver(), page: this);
}
1 usage
public void masInformacion() {
linkVerTransmision.click();
}
1 usage
public void setBtnVerMes() { btnVerMes.click(); }
3 usages
public void setBtnAnterior() {
btnAnterior.click();
}
1 usage
public void setDiaCalendario() {
diaCalendario.click();
}
1 usage
public void setBtnYoutube() {
btnYoutube.click();
Version Control
TODO
Problems
Terminal
Services
Dependencies

Anexo 6 *Lista y código de los Features de automatización*

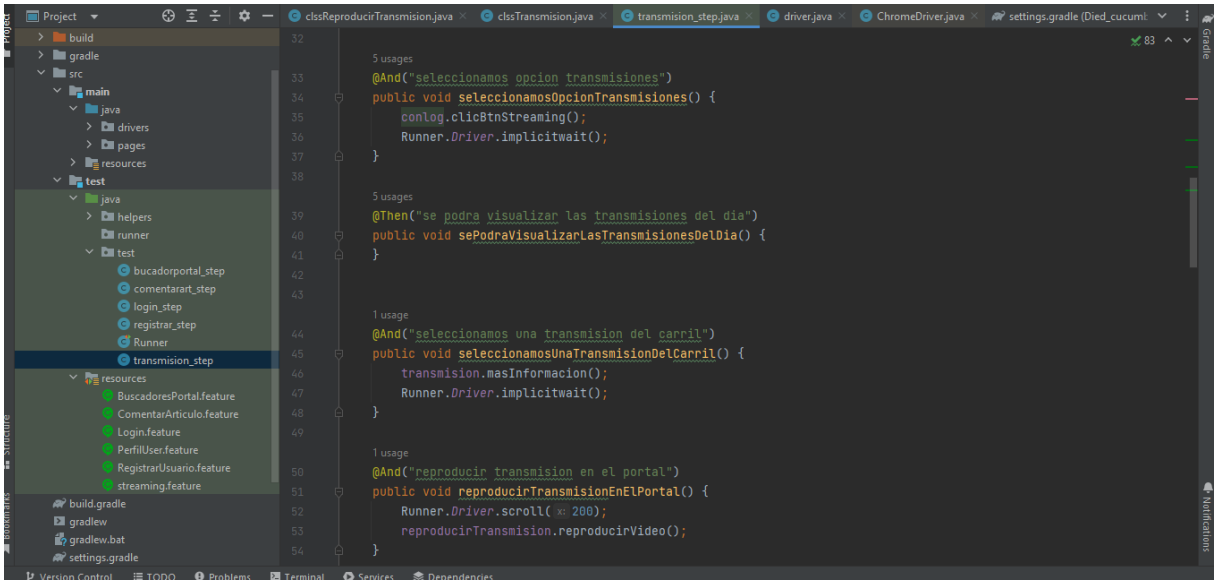


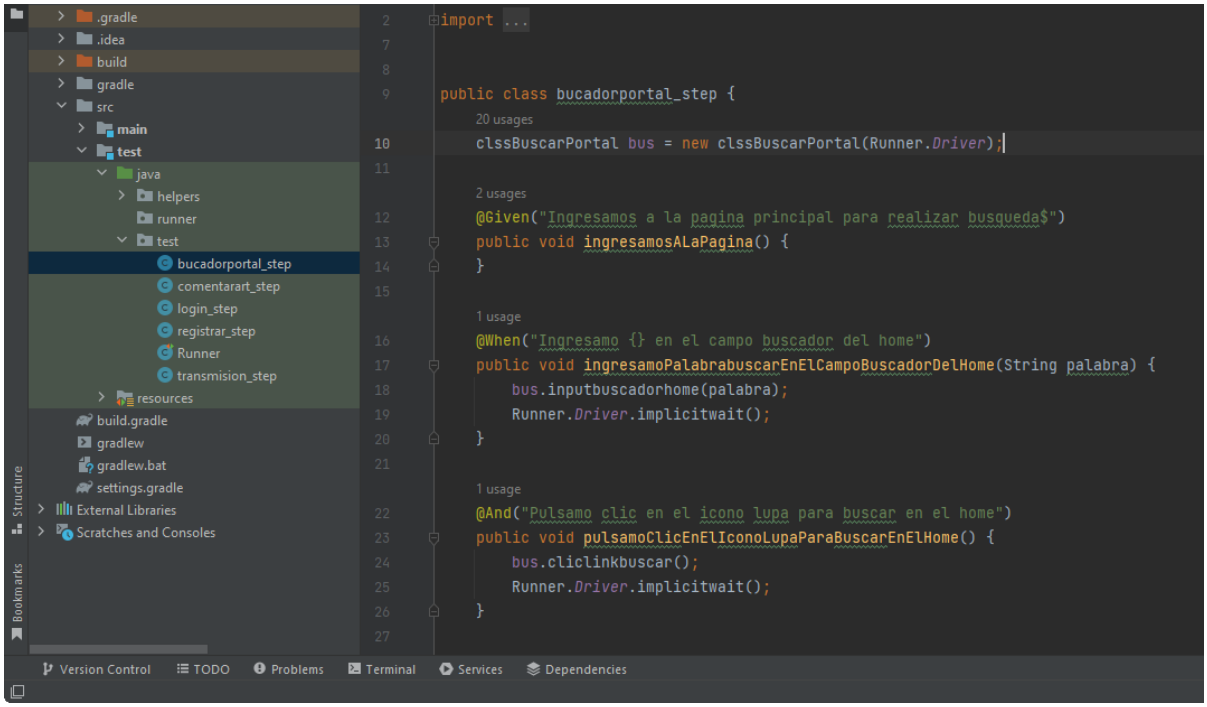


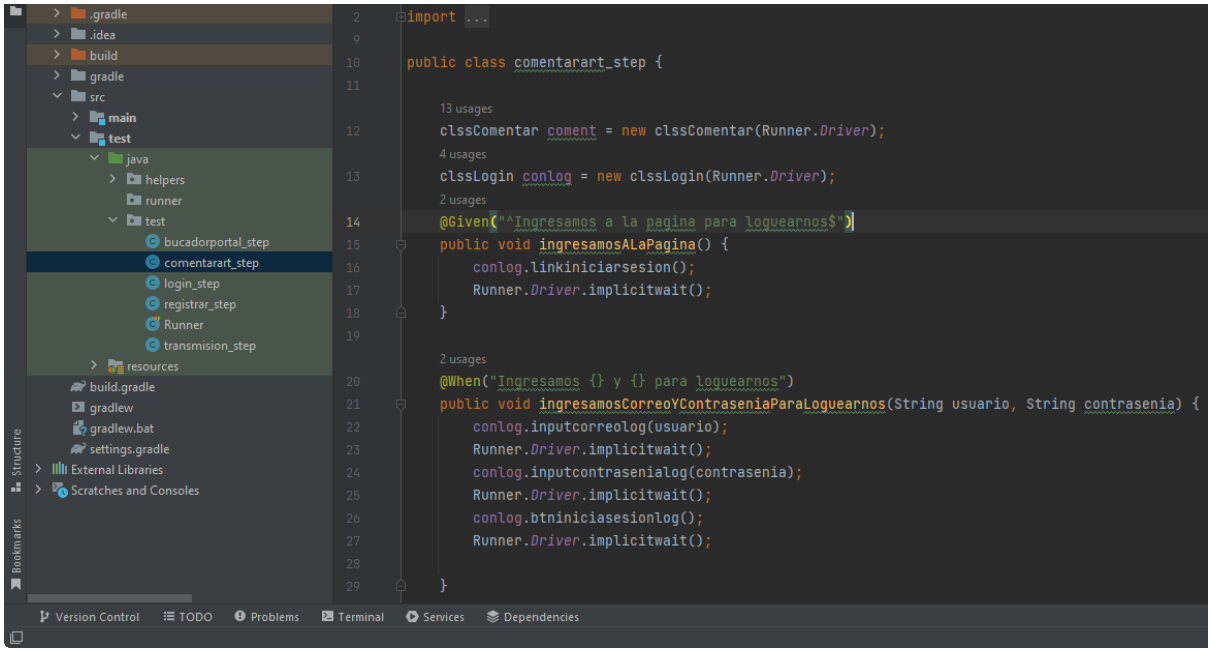

PeruEduca_cucumber [Died_cucumber]
.gradle
.idea
build
gradle
src
main
test
java
resources
BuscadoresPortal.feature
ComentarArticulo.feature
Login.feature
PerfilUser.feature
RegistrarUsuario.feature
streaming.feature
build.gradle
gradlew
gradlew.bat
settings.gradle
External Libraries
Scratches and Consoles
Feature: Comentar logueado correctamente
Scenario Outline: CPA-003. Comentar un Articulo de manera publico
Given Ingresamos a la pagina principal del portal
When Seleccionamos ver mas articulos que queremos que conozcas
And Seleccionamos <numpag> en el paginado de queremos que conozcas
And Seleccionamos un articulo de queremos que conozcas
And Verificamos que no tenga casilla de comentario
Then No se podra comentar un articulo de forma publica
Examples:
| numpag |
| 2 |
Scenario Outline: CPA-007. Comentar Articulo logueado
Given Ingresamos a la pagina para loguearnos
When Ingresamos <correo> y <contrasenia> para loguearnos
And Seleccionamos ver mas despues de loguearnos
And Seleccionamos un artitulo
And Ingresamos <texto> que comentaremos en el articulo
And Clic en enviar comentario en el articulo
And Clic en responder comentario del articulo
And Ingresamos <textoresp> como respuesta de comentario en el articulo
And Clic en enviar comentario para responder comentario en el articulo
Then Comento correctamente el articulo
Examples:
| correo | contrasenia | texto | textoresp |
Version Control
TODO
Problems
Terminal
Services
Dependencies

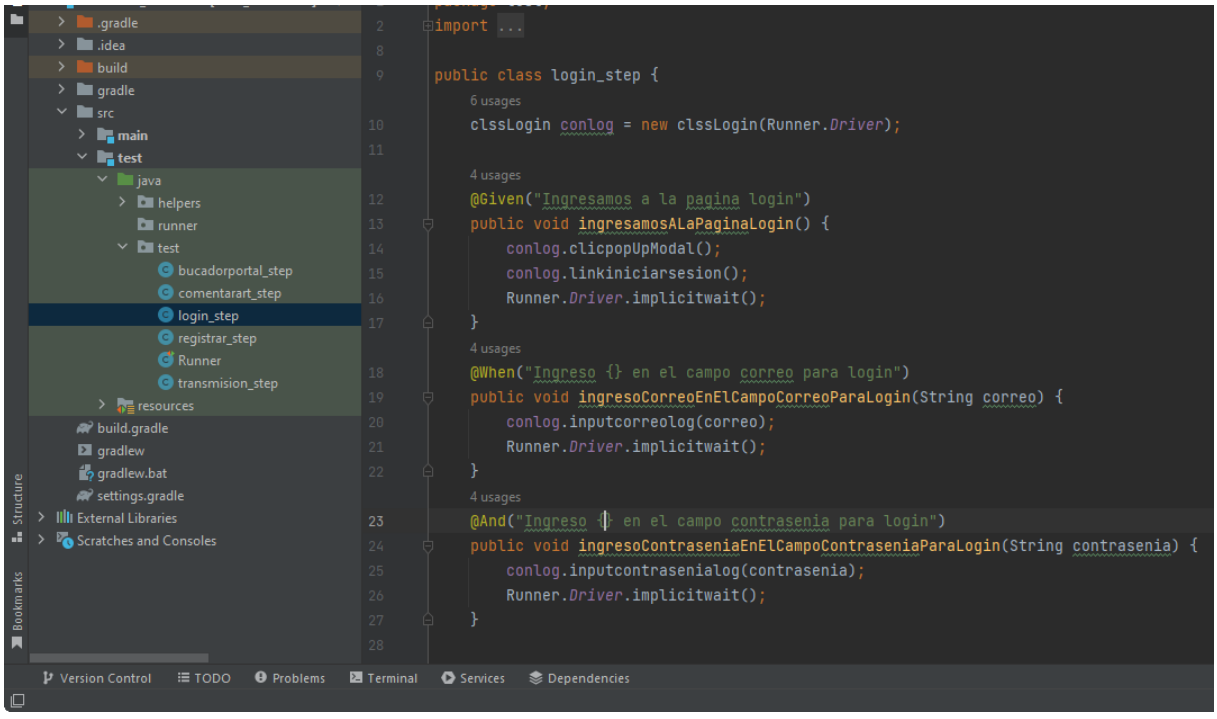

PeruEduca_cucumber [Died_cucumber]
.gradle
.idea
build
gradle
src
main
test
java
resources
BuscadoresPortal.feature
ComentarArticulo.feature
Login.feature
PerfilUser.feature
RegistrarUsuario.feature
streaming.feature
build.gradle
gradlew
gradlew.bat
settings.gradle
External Libraries
Scratches and Consoles
Feature: Buscadores del portal
Scenario Outline: CPA-001. Buscar articulos correctamente de la pantalla principal
Given Ingresamos a la pagina principal para realizar busqueda
When Ingresamo <palabrabuscar> en el campo buscador del home
And Pulsamo clic en el icono lupa para buscar en el home
And Pulsamo clic en la etiqueta articulos en el resultado
And Pulsamo clic en la etiqueta material educativo en el resultado
Then Aplicativo muestra correctamente los resultados de busqueda segun filtro
Examples:
| palabrabuscar |
| casa |
Scenario Outline: CPA-002. Buscar articulos correctamente de la pantalla secundario
Given Ingresamos a la pagina principal para realizar busqueda
When Seleccionamos opcion Mis servicios barra principal portal
And Seleccionamos icono Material educativo
And Ingresamo <palabrabuscar> en el campo buscador secundario
And Pulsamo clic en el icono lupa para buscador secundario
And Seleccionamos seccion Modalidad buscador secundario
And Seleccionamos <modalidad> de seccion Modalidad buscador secundario
And Seleccionamos seccion Nivel buscador secundario
And Seleccionamos <nivel> de seccion Nivel buscador secundario
And Seleccionamos seccion Grado buscador secundario
And Seleccionamos <grado> de seccion Grado buscador secundario
And Seleccionamos seccion area buscador secundario
And Seleccionamos <area> de seccion area buscador secundario
Version Control
TODO
Problems
Terminal
Services
Dependencies
Typo: In word 'Seleccionamos'

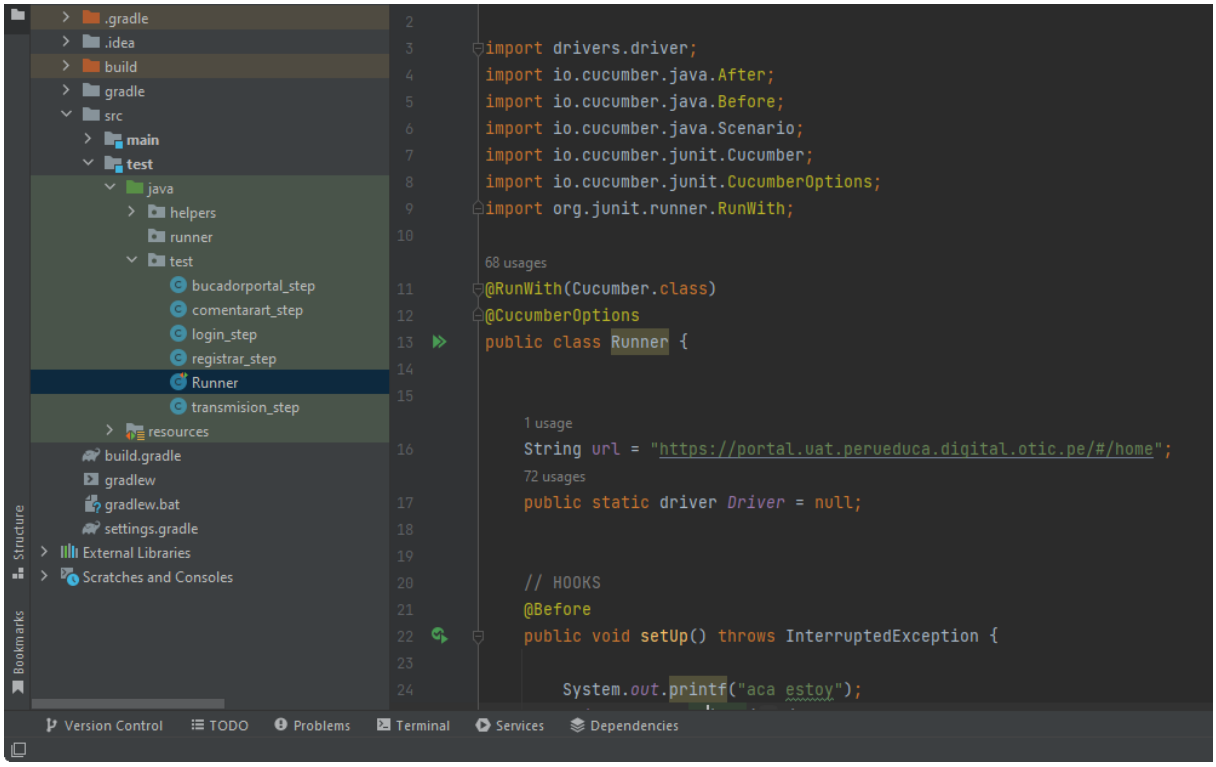
Anexo 7 *Lista y código de los Step Definition de automatización*






.gradle
.idea
build
gradle
src
main
test
java
helpers
runner
test
bucadorportal_step
comentarart_step
login_step
registrar_step
Runner
transmision_step
resources
build.gradle
gradlew
gradlew.bat
settings.gradle
External Libraries
Scratches and Consoles
import ...
public class comentarart_step {
13 usages
clssComentar coment = new clssComentar(Runner.Driver);
4 usages
clssLogin conlog = new clssLogin(Runner.Driver);
2 usages
@Given("^Ingresamos a la pagina para loguearnos$")
public void ingresamosALaPagina() {
conlog.linkiniciarsesion();
Runner.Driver.implicitwait();
}
2 usages
@When("Ingresamos {} y {} para loguearnos")
public void ingresamosCorreoYContraseniaParaLoguearnos(String usuario, String contrasenia) {
conlog.inputcorreolog(usuario);
Runner.Driver.implicitwait();
conlog.inputcontrasenialog(contrasenia);
Runner.Driver.implicitwait();
conlog.btniniciasesionlog();
Runner.Driver.implicitwait();
}
Version Control
TODO
Problems
Terminal
Services
Dependencies


.gradle
.idea
build
gradle
src
main
test
java
helpers
runner
test
bucadorportal_step
comentarart_step
login_step
registrar_step
Runner
transmision_step
resources
build.gradle
gradlew
gradlew.bat
settings.gradle
External Libraries
Scratches and Consoles
import ...
public class login_step {
6 usages
clssLogin conlog = new clssLogin(Runner.Driver);
4 usages
@Given("Ingresamos a la pagina login")
public void ingresamosALaPaginaLogin() {
conlog.clicpopUpModal();
conlog.linkiniciarsesion();
Runner.Driver.implicitwait();
}
4 usages
@When("Ingreso {} en el campo correo para login")
public void ingresoCorreoEnElCampoCorreoParaLogin(String correo) {
conlog.inputcorreolog(correo);
Runner.Driver.implicitwait();
}
4 usages
@And("Ingreso {} en el campo contrasenia para login")
public void ingresoContraseniaEnElCampoContraseniaParaLogin(String contrasenia) {
conlog.inputcontrasenialog(contrasenia);
Runner.Driver.implicitwait();
}
Version Control
TODO
Problems
Terminal
Services
Dependencies



Anexo 8 *Especificación de la técnica e instrumento de recolección de datos*

Descripción de la técnica para recolectar los datos		
Tipo de dato	**Técnica**	**Instrumento**
Cuantitativo	Observación	Ficha de observación para recolectar datos

Descripción del instrumento para recolectar los datos	
Ítem	**Descripción**
Título del instrumento	Ficha de observación para recolectar datos
Autor	Ing. Jhon Keny Quispe Gutierrez
Año	2022
Descripción	Para el estudio se estableció una ficha general escalable, que se adapta en su estructura para recolectar los datos de todos los indicadores establecidos en la investigación.
Nro. de Observaciones	26
Aplicación	Observación Directa

Anexo 9 *Estructura general escalable de la ficha técnica de recolección de datos*

Ficha de Registro <<número>>: Instrumento de medición del indicador tiempo de ejecucion de pruebas <<**Pretest/PosTest**>>

Observador:	<<***Nombres y Apellidos*** >
Proceso Observado:	<<***Nombre del Proceso*** >>

<<PRETEST/POSTEST>>					
Legacy	**Feature**	**Nro. Observación**	***<<Nombre Columna >>***	***<<Nombre Columna >>***	***<<Nombre Columna >>***
Portal PerúEduca v4.0	**Iniciar Sesión**				
	Buscar Recurso				
	Comentarios				
	Analítica				
	Registrar Usuario				
	Streaming				
	Registrar ME				
	Registrar AR				
	Transmisiones				
	Colecciones				
	Actualización de Datos				
		Total			

Anexo 10 *Ficha de Registro 1: Instrumento de medición del indicador tiempo de ejecución de pruebas PRETEST*

Observador:	Quispe Gutierrez Jhon Keny				
Proceso Observado:	Control de Calidad				
PRETEST					
Legacy	**Feature**	**Nro. Observación**	**Nro. Casos de Prueba Ejecutados**	**Tiempo por caso de Prueba(Min.)**	**Total Tiempo(Min.)**
Portal PerúEduca v4.0	**Iniciar Sesión**	1	7	15	105
		2	7	15	105
		3	7	15	105
	Buscar Recurso	4	5	10	50
		5	5	10	50
		6	5	10	50
	Comentarios	7	9	12	108
	Analítica	8	8	8	64
		9	8	8	64
		10	8	8	64
	Registrar Usuario	11	9	19	171
		12	9	19	171
	Streaming	13	9	11	99
		14	9	11	99
		15	9	11	99
	Registrar ME	16	4	20	80
		17	4	20	80
		18	4	20	80
	Registrar AR	19	6	25	150
		20	6	25	150
		21	6	25	150
	Transmisiones	22	12	18	216
	Colecciones	23	9	10	90
	Actualización de Datos	24	5	15	75
		25	5	15	75
		26	5	15	75
		Total	**180**	**390**	**2625**

Anexo 11 *Ficha de Registro 2: Instrumento de medición del indicador tiempo de ejecución de pruebas POSTEST*

Observador:	Quispe Gutierrez Jhon Keny				
Proceso Observado:	Control de Calidad				
POSTEST					
Legacy	**Feature**	**Nro. Observación**	**Nro. Casos de Prueba Ejecutados**	**Tiempo por caso de Prueba(Min.)**	**Total Tiempo(Min.)**
Portal PerúEduca v4.0	**Iniciar Sesión**	1	7	4	28
		2	7	4	28
		3	7	4	28
	Buscar Recurso	4	5	3	15
		5	5	3	15
		6	5	3	15
	Comentarios	7	9	4	36
	Analítica	8	8	3	24
		9	8	3	24
		10	8	3	24
	Registrar Usuario	11	9	6	54
		12	9	6	54
	Streaming	13	9	4	36
		14	9	4	36
		15	9	4	36
	Registrar ME	16	4	6	24
		17	4	6	24
		18	4	6	24
	Registrar AR	19	6	5	30
		20	6	5	30
		21	6	5	30
	Transmisiones	22	12	5	60
	Colecciones	23	9	3	27
	Actualización de Datos	24	5	3	15
		25	5	3	15
		26	5	3	15
		Total	**180**	**108**	**747**

Anexo 12 *Ficha de Registro 3: Instrumento de medición del indicador detección temprana de defectos PRETEST*

Observador:	Quispe Gutierrez Jhon Keny			
Proceso Observado:	Control de Calidad			
PRETEST				
Legacy	**Feature**	**Nro. Observación**	**Nro. Casos de Prueba Ejecutados**	**Cantidad de Bugs Detectados**
Portal PerúEduca v4.0	**Iniciar Sesión**	1	7	1
		2	7	1
		3	7	1
	Buscar Recurso	4	5	0
		5	5	2
		6	5	0
	Comentarios	7	9	1
	Analítica	8	8	1
		9	8	3
		10	8	1
	Registrar Usuario	11	9	2
		12	9	2
	Streaming	13	9	1
		14	9	0
		15	9	1
	Registrar ME	16	4	0
		17	4	1
		18	4	2
	Registrar AR	19	6	0
		20	6	2
		21	6	0
	Transmisiones	22	12	3
	Colecciones	23	9	1
	Actualización de Datos	24	5	0
		25	5	0
		26	5	1
		Total	**180**	**27**

Anexo 13 *Ficha de Registro 4: Instrumento de medición del indicador detección temprana de defectos POSTEST*

Observador:	Quispe Gutierrez Jhon Keny			
Proceso Observado:	Control de Calidad			
POSTEST				
Legacy	**Feature**	**Nro. Observación**	**Nro. Casos de Prueba Ejecutados**	**Cantidad de Bugs Detectados**
Portal PerúEduca v4.0	**Iniciar Sesión**	1	7	1
		2	7	2
		3	7	1
	Buscar Recurso	4	5	2
		5	5	0
		6	5	1
	Comentarios	7	9	3
	Analítica	8	8	1
		9	8	1
		10	8	2
	Registrar Usuario	11	9	0
		12	9	1
	Streaming	13	9	2
		14	9	2
		15	9	0
	Registrar ME	16	4	1
		17	4	1
		18	4	1
	Registrar AR	19	6	2
		20	6	1
		21	6	2
	Transmisiones	22	12	5
	Colecciones	23	9	1
	Actualización de Datos	24	5	1
		25	5	1
		26	5	1
		Total	**180**	**36**

Anexo 14 *Ficha de Registro 5: Instrumento de medición del indicador Cantidad de Pruebas de Regresión PRETEST*

Observador:	Quispe Gutierrez Jhon Keny				
Proceso Observado:	Control de Calidad				
PRETEST					
Legacy	**Feature**	**Nro. Observación**	**Nro. Pruebas de Regresión Planificadas**	**Nro. De Pruebas de Regresión Ejecutadas**	**% de cobertura=(E/P)* 100**
Portal PerúEduca v4.0	**Iniciar Sesión**	1	7	3	43%
		2	7	2	29%
		3	7	4	57%
	Buscar Recurso	4	4	1	25%
		5	4	1	25%
		6	4	1	25%
	Comentarios	7	6	3	50%
	Analítica	8	5	1	20%
		9	5	5	100%
		10	5	2	40%
	Registrar Usuario	11	6	0	0%
		12	6	0	0%
	Streaming	13	4	2	50%
		14	4	0	0%
		15	4	0	0%
	Registrar ME	16	3	1	33%
		17	3	1	33%
		18	3	0	0%
	Registrar AR	19	5	3	60%
		20	5	3	60%
		21	5	1	20%
	Transmisiones	22	7	3	43%
	Colecciones	23	2	2	100%
	Actualización de Datos	24	4	0	0%
		25	4	0	0%
		26	4	2	50%
		Total	**123**	**41**	**33%**

Anexo 15 *Ficha de Registro 6: Instrumento de medición del indicador Cantidad de Pruebas de Regresión POSTEST*

Observador:	Quispe Gutierrez Jhon Keny				
Proceso Observado:	Control de Calidad				
POSTEST					
Legacy	**Feature**	**Nro. Observación**	**Nro. Pruebas de Regresión Planificadas**	**Nro. De Pruebas de Regresión Ejecutadas**	**% de cobertura=(E/P)* 100**
Portal PerúEduca v4.0	**Iniciar Sesión**	1	7	6	86%
		2	7	6	86%
		3	7	6	86%
	Buscar Recurso	4	4	4	100%
		5	4	4	100%
		6	4	4	100%
	Comentarios	7	6	5	83%
	Analítica	8	5	5	100%
		9	5	5	100%
		10	5	5	100%
	Registrar Usuario	11	6	6	100%
		12	6	6	100%
	Streaming	13	4	3	75%
		14	4	4	100%
		15	4	4	100%
	Registrar ME	16	3	3	100%
		17	3	3	100%
		18	3	3	100%
	Registrar AR	19	5	5	100%
		20	5	5	100%
		21	5	5	100%
	Transmisiones	22	7	7	100%
	Colecciones	23	2	2	100%
	Actualización de Datos	24	4	4	100%
		25	4	4	100%
		26	4	4	100%
		Total	**123**	**118**	**96%**

Anexo 16 *Ficha de Registro 7: Instrumento de medición del indicador eficiencia en la detección del defecto PRETEST*

Observador:	Quispe Gutierrez Jhon Keny					
Proceso Observado:	Control de Calidad					
PRETEST						
Legacy	**Feature**	**Nro. Observación**	**Nro. Casos de Prueba Ejecutados**	**Cantidad de defectos en Pruebas del sistema**	**Fallas en Producción**	**Total**
Portal PerúEduca v4.0	**Iniciar Sesión**	1	7	1	0	1
		2	7	1	0	1
		3	7	1	0	1
	Buscar Recurso	4	5	0	1	1
		5	5	2	0	2
		6	5	0	1	1
	Comentarios	7	9	1	0	1
	Analítica	8	8	1	0	1
		9	8	3	0	3
		10	8	1	0	1
	Registrar Usuario	11	9	2	0	2
		12	9	2	0	2
	Streaming	13	9	1	0	1
		14	9	0	1	1
		15	9	1	0	1
	Registrar ME	16	4	0	1	1
		17	4	1	0	1
		18	4	2	0	2
	Registrar AR	19	6	0	1	1
		20	6	2	0	2
		21	6	0	1	1
	Transmisiones	22	12	3	0	3
	Colecciones	23	9	1	0	1
	Actualización de Datos	24	5	0	1	1
		25	5	0	0	0
		26	5	1	0	1
		Total	**180**	**27**	**7**	**34**

Anexo 17 *Ficha de Registro 8: Instrumento de medición del indicador eficiencia en la detección del defecto POSTEST*

Observador:	Quispe Gutierrez Jhon Keny
Proceso Observado:	Control de Calidad

POSTEST						
Legacy	**Feature**	**Nro. Observación**	**Nro. Casos de Prueba Ejecutados**	**Cantidad de defectos en Pruebas del sistema**	**Fallas en Producción**	**Total**
Portal PerúEduca v4.0	**Iniciar Sesión**	1	7	1	0	1
		2	7	2	0	2
		3	7	1	0	1
	Buscar Recurso	4	5	2	0	2
		5	5	0	0	0
		6	5	1	0	1
	Comentarios	7	9	3	0	3
	Analítica	8	8	1	0	1
		9	8	1	1	2
		10	8	2	0	2
	Registrar Usuario	11	9	0	0	0
		12	9	1	0	1
	Streaming	13	9	2	0	2
		14	9	2	0	2
		15	9	0	0	0
	Registrar ME	16	4	1	0	1
		17	4	1	0	1
		18	4	1	0	1
	Registrar AR	19	6	2	0	2
		20	6	1	0	1
		21	6	2	0	2
	Transmisiones	22	12	5	0	5
	Colecciones	23	9	1	0	1
	Actualización de Datos	24	5	1	0	1
		25	5	1	0	1
		26	5	1	0	1
		Total	**180**	**36**	**1**	**37**

Anexo 18 *Ficha de Registro 9: Instrumento de medición del indicador Tasa de diseño de casos de prueba automatizados PRETEST/POSTEST*

Observador:	Quispe Gutierrez Jhon Keny			
Proceso Observado:	Control de Calidad			
PRETEST/POSTEST				
Legacy	**Feature**	**Nro. Observación**	**Nro. Test Scripts iniciales**	**Nuevos Diseños Casos de Prueba**
Portal PerúEduca v4.0	**Iniciar Sesión/Legados de PerúEduca**	1	7	7
		2	7	7
		3	7	7
	Buscar Recurso	4	5	2
		5	5	2
		6	5	2
	Comentarios/legados	7	9	3
	Analítica	8	8	5
		9	8	5
		10	8	5
	Registrar Usuario	11	9	4
		12	9	4
	Streaming	13	9	1
		14	9	1
		15	9	1
	Registrar ME	16	4	2
		17	4	2
		18	4	2
	Registrar AR	19	6	3
		20	6	3
		21	6	3
	Transmisiones	22	12	0
	Colecciones	23	9	4
	Actualización de Datos	24	5	1
		25	5	1
		26	5	1
		Total	**180**	**78**
			60	**26**

Anexo 19 *Listado de casos iniciales propuestos para las pruebas*

RELACIÓN DE CASOS DE PRUEBA - NUEVO PORTAL MVP1 Y REALEASE 2				
Código	**Funcionalidad**	**Nombre del caso de prueba**	**PRIORIDAD**	**DISEÑADO**
CP-001	Iniciar sesión	Validar inicio de sesión con usuario y contraseña correctos	1. Alto	SI
CP-002	Iniciar sesión	Validar inicio de sesión con usuario y contraseña incorrectos	1. Alto	SI
CP-003	Iniciar sesión	Validar inicio de sesión con usuario correcto y contraseña incorrecto	1. Alto	SI
CP-004	Iniciar sesión	Validar inicio de sesión con usuario incorrecto y contraseña correcta	1. Alto	SI
CP-005	Iniciar sesión	Validar inicio de sesión con usuario y contraseña vacíos	1. Alto	SI
CP-006	Iniciar sesión	Validar inicio de sesión sólo con usuario vacío	1. Alto	SI
CP-007	Iniciar sesión	Validar inicio de sesión sólo con contraseña vacío	1. Alto	SI
CP-008	Buscar Recurso	Buscar artículo (AR) desde el Head del Portal	2. Medio	SI
CP-009	Buscar Recurso	Buscar material educativo (ME) desde el Head del Portal	2. Medio	SI
CP-010	Buscar Recurso	Búsqueda avanzada de AR desde el BackOffice	2. Medio	SI
CP-011	Buscar Recurso	Búsqueda avanzada de ME desde el BackOffice	2. Medio	SI
CP-012	Buscar Recurso	Búsqueda por rango de fecha de AR/ME desde el BackOffice	2. Medio	SI
CP-013	Comentarios	Registrar comentario de un AR.	2. Medio	SI
CP-014	Comentarios	Registrar comentario de un ME.	2. Medio	SI
CP-015	Comentarios	Registrar respuesta a un comentario de un AR.	2. Medio	SI
CP-016	Comentarios	Registrar respuesta a un comentario de un ME.	2. Medio	SI
CP-017	Comentarios	Validar BlackList de artículos	2. Medio	SI

RELACIÓN DE CASOS DE PRUEBA - NUEVO PORTAL MVP1 Y REALEASE 2				
Código	**Funcionalidad**	**Nombre del caso de prueba**	**PRIORIDAD**	**DISEÑADO**
CP-018	Comentarios	Validar BlackList de materiales educativos	2. Medio	SI
CP-019	Comentarios	Eliminar comentario de AR	2. Medio	SI
CP-020	Comentarios	Eliminar comentario de ME	2. Medio	SI
CP-021	Comentarios	Validar actualización de comentario.	2. Medio	SI
CP-022	Analítica	Validar la cantidad de valoraciones de ME	1. Alto	SI
CP-023	Analítica	Validar la cantidad de descargas de ME	1. Alto	SI
CP-024	Analítica	Validar la cantidad de compartidos de ME	1. Alto	SI
CP-025	Analítica	Validar la cantidad de comentarios de ME	1. Alto	SI
CP-026	Analítica	Validar la cantidad de valoraciones de AR	1. Alto	SI
CP-027	Analítica	Validar la cantidad de visitas de AR	1. Alto	SI
CP-028	Analítica	Validar la cantidad de compartidos de AR	1. Alto	SI
CP-029	Analítica	Validar la cantidad de comentarios de AR	1. Alto	SI
CP-030	Registrar Usuario	Registrar usuario docente público - DNI	1. Alto	SI
CP-031	Registrar Usuario	Registrar usuario docente público - CE	1. Alto	SI
CP-032	Registrar Usuario	Registrar usuario docente público - PAS	1. Alto	SI
CP-033	Registrar Usuario	Registrar usuario director - DNI	1. Alto	SI
CP-034	Registrar Usuario	Registrar usuario director - CE	1. Alto	SI
CP-035	Registrar Usuario	Registrar usuario director – PAS	1. Alto	SI
CP-036	Registrar Usuario	Registrar usuario estudiante - DNI	1. Alto	SI
CP-037	Registrar Usuario	Registrar usuario estudiante - CE	1. Alto	SI
CP-038	Registrar Usuario	Registrar usuario estudiante – PAS	1. Alto	SI

RELACIÓN DE CASOS DE PRUEBA - NUEVO PORTAL MVP1 Y REALEASE 2				
Código	**Funcionalidad**	**Nombre del caso de prueba**	**PRIORIDAD**	**DISEÑADO**
CP-039	Streaming	Visualizar Carril de Transmisiones - Portal	2. Medio	SI
CP-040	Streaming	Buscar transmisiones desde el calendario - Portal	2. Medio	SI
CP-041	Streaming	Buscar transmisiones desde <<link>> - Portal	2. Medio	SI
CP-042	Streaming	Reproducir Transmisión desde el Portal	2. Medio	SI
CP-043	Streaming	Registrar transmisión - BackOffice	2. Medio	SI
CP-044	Streaming	Editar transmisión - BackOffice	2. Medio	SI
CP-045	Streaming	Publicar transmisión - BackOffice	2. Medio	SI
CP-046	Streaming	Eliminar transmisión – BackOffice	2. Medio	SI
CP-047	Streaming	Inactivar transmisión – BackOffice	2. Medio	SI
CP-048	Registrar ME	Registrar ME - BackOffice	2. Medio	SI
CP-049	Registrar ME	Editar ME - BackOffice	2. Medio	SI
CP-050	Registrar ME	Publicar ME - BackOffice	2. Medio	SI
CP-051	Registrar ME	Inactivar ME - BackOffice	2. Medio	SI
CP-052	Registrar AR	Registrar AR - BackOffice	2. Medio	SI
CP-053	Registrar AR	Editar AR - BackOffice	2. Medio	SI
CP-054	Registrar AR	Inactivar AR - BackOffice	2. Medio	SI
CP-055	Registrar AR	Publicar AR - BackOffice	2. Medio	SI
CP-056	Colecciones	Agregar Colecciones ME - BackOffice	3. Bajo	SI
CP-057	Colecciones	Publicar Colecciones ME - BackOffice	3. Bajo	SI
CP-058	Colecciones	Inactivar Colecciones ME - BackOffice	3. Bajo	SI
CP-059	Actualización de Datos	Actualizar Usuario	3. Bajo	SI
CP-060	Actualización de Datos	Actualizar Email	3. Bajo	SI

Anexo 20 *Diagrama causa efecto del problema de investigación*