

Estrategia de pruebas

1. Aplicación Bajo Pruebas

1.1. Nombre de la Aplicación: Ghost

1.2. Versión: 3.42.5

1.3. Descripción: Ghost es un CMS completamente autónomo y es completamente independiente de cualquier front-end o marco de sitio estático en particular. Es una plataforma de publicación profesional de código abierto construida sobre una pila de tecnología Node.js moderna, diseñada para equipos que necesitan potencia, flexibilidad y rendimiento. Centrada únicamente en la publicación profesional. Impulsa una increíble variedad de sitios web; desde blogueros individuales que recién están comenzando, hasta grandes equipos de escritores y editores en algunas de las organizaciones más grandes del mundo. Financiada por una plataforma Premium como servicio para ejecutarla.

1.4. Funcionalidades Core

- Permite cargar themes mediante css y javascript.
- Ofrece la posibilidad Migrar desde wordpress u otras plataformas a Ghost.
- Permite agregar tablas de contenido y formularios de contacto.
- Scheduler para publicaciones de noticias o páginas.
- Ofrece integración con mapas de sitios como noticias, enlaces a redes sociales.
- Permite realizar búsquedas avanzadas y personalizadas.
- Ofrece herramientas para la creación de contenido multilinguaje.
- Generación de reportes.

1.5. Diagrama de Arquitectura

<https://github.com/abaron10/IncidenciasGhost/wiki/Modelo-de-Arquitectura>

1.6. Diagrama de Contexto

<https://github.com/abaron10/IncidenciasGhost/wiki/Modelo-contexto>

1.7. Modelo de Datos

<https://github.com/abaron10/IncidenciasGhost/wiki/Modelo-de-dominio>

1.8. Modelo GUI

<https://github.com/abaron10/IncidenciasGhost/wiki/Modelo-GUI-del-sistema>.

2. Contexto estrategia de pruebas

2.1. Objetivos

- Realizar los diferentes tipos de pruebas de software en el tiempo establecido a las funcionalidades CORE propuestas con los recursos disponibles.
- Identificar las fallas del sistema asociadas a las funcionalidades CORE propuestas anteriormente con el fin de mejorar la experiencia y funcionamiento del software.
- Priorizar los tipos de pruebas a implementar, donde con ayuda del capital humano se facilite la identificación de problemas presentes en la aplicación durante el tiempo establecido.
- Entregar la propuesta para antes del lunes 31 de mayo las 8 am, con el fin de que el CTO (Chief technical officer) de su aprobación.
- Probar la aplicación en niveles de integración y sistema mediante el uso de técnicas y herramientas usadas en las pruebas de software.

2.2. Duración de la iteración de pruebas

Para las pruebas se cuenta con 4 ingenieros testers senior, que estarán disponibles para trabajar durante 8 semanas, en una dedicación de 8 horas por persona semanalmente.

Las iteraciones se harán los martes en horario de 8:00 am a 6:00 pm de cada semana durante un periodo de 8 semanas. Iniciando el proyecto el 25/05/2020 y terminando el 13/07/2021 con una duración total de 36 días.

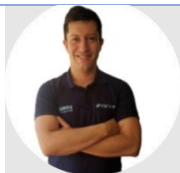
		Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras	Nombres de los recursos
1			➤ Estrategia de pruebas TSDC	35,25 días	mar 25/05/2	mar 13/07/2		
2			➤ Pruebas de reconocimiento Monkeys	0,75 días	mar 25/05/2	mar 25/05/2		
3			Analisis	1 hora	mar 25/05/2	mar 25/05/2		Jhonatan Segura
4			Diseño	1 hora	mar 25/05/2	mar 25/05/2	3	Jhonatan Segura
5			Ejecución	1 hora	mar 25/05/2	mar 25/05/2		Jhonatan Segura
6			Refactoring y ejecución	5 horas	mar 25/05/2	mar 25/05/2	5	Jhonatan Segura
7			➤ Pruebas de reconocimiento Ryppers	1 día	mar 1/06/21	mar 1/06/21		
8			Analisis	1 hora	mar 1/06/21	mar 1/06/21		Andres Baron
9			Diseño	1 hora	mar 1/06/21	mar 1/06/21	8	Andres Baron
10			Ejecución	1 hora	mar 1/06/21	mar 1/06/21	9	Andres Baron
11			Refactoring y ejecución	5 horas	mar 1/06/21	mar 1/06/21	10	Andres Baron
12			➤ Pruebas Manuales	1 día	mar 8/06/21	mar 8/06/21		
13			Analisis	1 hora	mar 8/06/21	mar 8/06/21		Andres Salas
14			Diseño	1 hora	mar 8/06/21	mar 8/06/21	13	Andres Salas
15			Ejecución	1 hora	mar 8/06/21	mar 8/06/21	14	Andres Salas
16			Refactoring y ejecución	5 horas	mar 8/06/21	mar 8/06/21	15	Andres Salas
17			➤ Pruebas de extremo a extremo	1 día	mar 15/06/2	mar 15/06/2		
18			Analisis	1 hora	mar 15/06/2	mar 15/06/2		Johan CV
19			Diseño	1 hora	mar 15/06/2	mar 15/06/2	18	Johan CV
20			Ejecución	1 hora	mar 15/06/2	mar 15/06/2	19	Johan CV
21			Refactoring y ejecución	5 horas	mar 15/06/2	mar 15/06/2	20	Johan CV
22			➤ Pruebas de extremo a extremo	1 día	mar 22/06/2	mar 22/06/2		
23			Analisis	1 hora	mar 22/06/2	mar 22/06/2		Jhonatan Segura
24			Diseño	1 hora	mar 22/06/2	mar 22/06/2	23	Jhonatan Segura
25			Ejecución	1 hora	mar 22/06/2	mar 22/06/2	24	Jhonatan Segura
26			Refactoring y ejecución	5 horas	mar 22/06/2	mar 22/06/2	25	Jhonatan Segura
27			➤ Pruebas de regresion visual	1 día	mar 29/06/2	mar 29/06/2		
28			Analisis	1 hora	mar 29/06/2	mar 29/06/2		Andres Baron
29			Diseño	1 hora	mar 29/06/2	mar 29/06/2	28	Andres Baron
30			Ejecución	1 hora	mar 29/06/2	mar 29/06/2	29	Andres Baron
31			Refactoring y ejecución	5 horas	mar 29/06/2	mar 29/06/2	30	Andres Baron
32			➤ Pruebas con validación de datos	1 día	mar 6/07/21	mar 6/07/21		
33			Analisis	1 hora	mar 6/07/21	mar 6/07/21		Johan CV
34			Diseño	1 hora	mar 6/07/21	mar 6/07/21	33	Johan CV
35			Ejecución	1 hora	mar 6/07/21	mar 6/07/21	34	Johan CV
36			Refactoring y ejecución	5 horas	mar 6/07/21	mar 6/07/21	35	Johan CV
37			➤ Pruebas de extremo a extremo	35,25 días	mar 25/05/2	mar 13/07/2		
38			Analisis	1 hora	mar 13/07/2	mar 13/07/2		Andres Salas
39			Diseño	1 hora	mar 13/07/2	mar 13/07/2	38	Andres Salas
40			Ejecución	1 hora	mar 25/05/2	mar 25/05/2		Andres Salas
41			Refactoring y ejecución	5 horas	mar 25/05/2	mar 25/05/2	40	Andres Salas

Figura 1. Plan de trabajo para la ejecución de la iteración de pruebas.

2.3 Presupuesto de pruebas

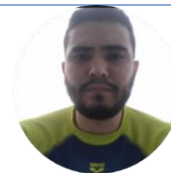
2.3.1 Recursos Humanos

A continuación, se presenta el perfil técnico de los colaboradores tentativos para este proyecto.



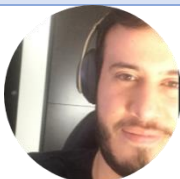
Jhonatan Segura:

Analista de calidad con 10 años de experiencia en automatización de pruebas usando plataformas como Cypress, Kraken, Celenium, con habilidades para formular planes de pruebas



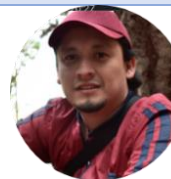
Johan Carvajal:

Lider QA con más de 5 años de experiencia como tester, experiencia en diseño y definición de sets y casos de pruebas funcionales, plan de pruebas, diseño de estrategia de calidad. Manejo Cypress, Kraken, Cucumber.



Fernando Barón:

Analista de calidad con 9 años de experiencia en automatización de pruebas de aplicaciones web manejo de plataformas como Cypress, Cucumber, Selenium, manejo de patrones screenplay y pageObjectModel



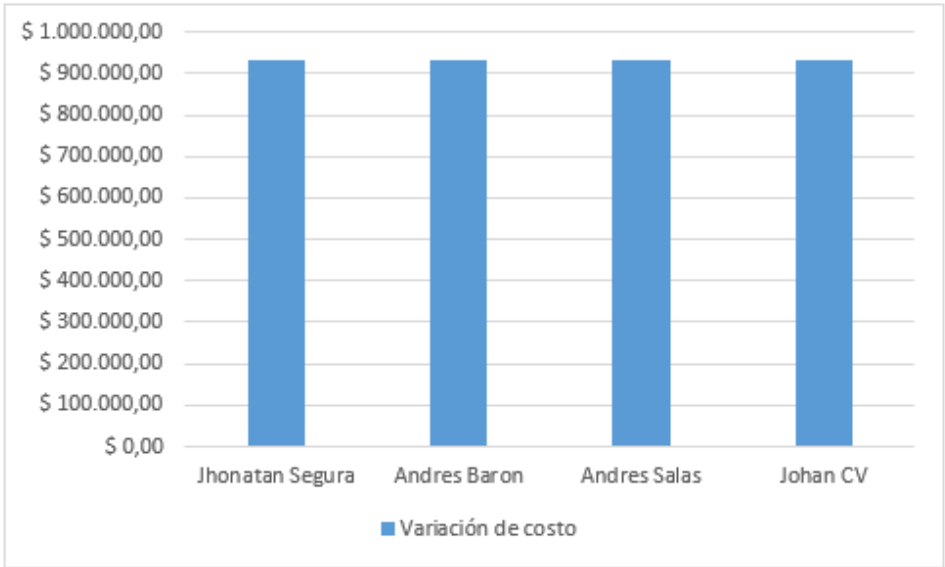
Andrés Salas:

Analista de pruebas con 8 años de experiencia en construcción de pruebas funcionales de software, usando plataformas como Cucumber, Selenium, manejo de lenguaje Gerkin con capacidad de elaboración de casos de pruebas, análisis y comprensión.

A continuación, se muestra la propuesta salarial según los valores de mercados establecidos en Colombia.

VARIACIÓN DE COSTO DE RECURSOS

Variación de costo de todos los recursos de trabajo.



Nombre	Costo	Costo de línea base	Variación de costo
Jhonatan Segura	\$ 930.000,00	\$ 0,00	\$ 930.000,00
Andres Baron	\$ 930.000,00	\$ 0,00	\$ 930.000,00
Andres Salas	\$ 930.000,00	\$ 0,00	\$ 930.000,00
Johan CV	\$ 930.000,00	\$ 0,00	\$ 930.000,00





Figura 2. Ingenieros asignados para el proyecto

Nombre	% completado	Costo	Costo de línea base	Variación de costo
Estrategia de pruebas TSDC	0%	\$ 3.720.000,00	\$ 0,00	\$ 3.720.000,00

Tabla 1. Distribución de costos por recurso humano para pruebas.

2.3.2 Recursos Computacionales

Cada ingeniero cuenta con su computador propio, al analizar las especificaciones de cada uno se evidencia que son maquinas contemporáneas aptas para el desarrollo de software por lo que no se esperan retrasos o inconvenientes a la hora de realizarlas pruebas.

Jhonatan Segura	MacBook Pro-2019 	Procesador: Intel Core i5 Memoria RAM: 16GB de 2133MHz Turbo Boost 3.3GHz SSD basado en PCIe de 256GB7 4 puertos Thunderbolt 3
Johan Carvajal	MacBook Pro-2019 	Procesador: Intel Core i5 Memoria RAM: 16GB de 2133MHz Turbo Boost 3.3GHz SSD basado en PCIe de 256GB7 4 puertos Thunderbolt 3
Andrés Salas	Asus vivobook x512k 	Procesador: Intel Core I5-8365U Memoria RAM : 8GB Turbo Boost 3.9GHz SSD 256 GB
Andrés Barón	Lenovo IdeaPad 3 AMD Athlon 	Procesador: AMD ATHLON SILVER 3050U 2.3G 2C MB Memoria RAM : 8GB Turbo boost: 2.3GHZ SSD 256GB

2.3.3 Recursos Económicos para la contratación de servicios/personal

Para esta fase no se contempla la tercerización de un servicio, por lo que en la etapa de revisión final se estudiara esta posibilidad dependiendo del rendimiento y avance de los ingenieros en cuestión.

2.4 TNT (Técnicas, Niveles y Tipos) de pruebas:

Pruebas de Integración

Para la estrategia de pruebas, se le dará prioridad a los niveles de integración y de sistema. El sistema de Ghost ya cuenta con módulos integrados funcionales por lo que se tomó la decisión de proponer pruebas de integración con el fin de comprobar que los componentes tengan un comportamiento estable según lo esperado a la hora de integrarlos en un todo. A continuación se explica brevemente las herramientas propuestas para el nivel de integración:

1. Cypress:

Es una herramienta que permite las pruebas de extremo a extremo para aplicaciones web. Compatible con frameworks de desarrollo web modernos. <https://www.cypress.io/how-it-works>

2. Kraken:

Kraken es una herramienta de código abierto para realizar pruebas automáticas del tipo E2E con aplicaciones móviles para Android y también con aplicaciones web. Esta soporta escenarios donde es requerida la intercomunicación entre usuarios o dispositivos. El proyecto fue desarrollado haciendo uso de Cucumber y Calabash, que son librerías manejadas por Ruby para realizar pruebas por medio de una notación que asemeja el lenguaje natural y permiten crear escenarios a través de la especificación de ejemplos. Su extensión para aplicaciones web se maneja por medio de los drivers Geckodriver/Chromedriver y Selenium WebDriver.

Pruebas de Sistema

El nivel de sistema según lo explicado anteriormente se puede probar debido a que ya se cuenta con una integración preliminar de los componentes, por lo que pruebas de tipo Positivas / Negativas ayudaran a encontrar posibles fallos en la aplicación. Como herramientas para la generación de datos aleatorios se proponen las siguientes:

Mockaroo:

Esta API maneja un endpoint en el cual usuario selecciona diferentes tipos de atributos para generar datos aleatorios e inyectarlos en pruebas para la aplicación. Estos datos extraídos pueden ser usados en técnicas como pool de datos pseudoaleatorios y comprobar el correcto funcionamiento de la aplicación.

Faker:

Es una librería que permite generar un pool de datos en tiempo de ejecución en diferentes formatos, como: database, datatype, company, date, fake, finanzas internet, texto, música, teléfono, música. El data pool es un conjunto de datos usado como repositorio de las entradas a ser utilizadas en pruebas. Desde el punto de vista del propósito, los data pools pueden ser genéricos o específicos para una funcionalidad en particular.

ResembleJS & BackstopJS

Que es Resemble Js:

Es una herramienta para el análisis y comparación de imágenes con Javascript, tanto para Html5 como para NodeJs.

Que es Backstop Js:

BackstopJS es una herramienta de código abierto para probar aplicaciones web, que se centra en cómo se ve su aplicación desde la perspectiva del usuario final. Automatiza las pruebas de regresión visual de su interfaz de usuario web receptiva comparando capturas de pantalla a lo largo del tiempo.

Fuente: [Backstop Js](#).

Pruebas exploratorias

Rippers & Monkeys

Los monkeys son pruebas que inyectan datos aleatorios en una interfaz web, son de bajo costo y ayudan a encontrar fallos en el sistema que los testers no encontrarían fácilmente. Los rippers a diferencia de los monkeys, son pruebas de exploración sistemática de la aplicación, permitiendo seguir una traza de los errores con el fin de poderlos reproducir.

Pruebas de exploración Manuales

Pruebas exploratorias manuales basadas en la experiencia del usuario (Tester Senior) con el fin de familiarizarse con la aplicación y encontrar fallos que son mas notorios a simple vista.

Niveles

En la tabla 2. Subrayado en amarillo se ve la selección de los niveles de pruebas escogidas para la iteración para la planeación propuesta y el objetivo que cumple esta implementación.

Nivel	Tipo	Técnica	Objetivo
Unidad	Funcionales	Caja blanca	Cubrir un mayor espacio de estados y entradas
Integración	Funcionales/No funcionales	Caja gris	Comprobar que los componentes se comportan de forma estable al integrarlos
Sistema	Positivas/Negativas	Caja negra	Probar el sistema como un todo
Aceptación	Positivas/Negativas	Caja Negra	Probar el sistema en un ambiente de producción

Tabla 2. Implementación de tácticas, niveles y tipos.

2.5. Distribución de esfuerzo

La distribución de esfuerzo está centrada en la realización de pruebas exploratorias las cuales van a permitir detectar errores de manera temprana. La dedicación será del 23%. Luego de las dos primeras iteraciones se ejecutarán pruebas manuales que permitirán encontrar errores que un QA senior con su experiencia logra identificar a diferencia de un bot. Estas pruebas tendrán una dedicación del 13%. Luego de esto se correrán 2 iteraciones de pruebas de extremo a extremo que incluyen un pequeño ciclo de refactoring de pruebas las cuales se harán más robustas con cada iteración. Estas pruebas tienen una dedicación del 26%. Aplicaremos luego pruebas de validación de datos y pruebas de regresión visual. Estas tendrán una dedicación total del 13% cada una. Finalmente correremos un ciclo de pruebas de extremo a extremo para asegurar la versión final. A continuación, se muestra de manera gráfica la distribución del esfuerzo.

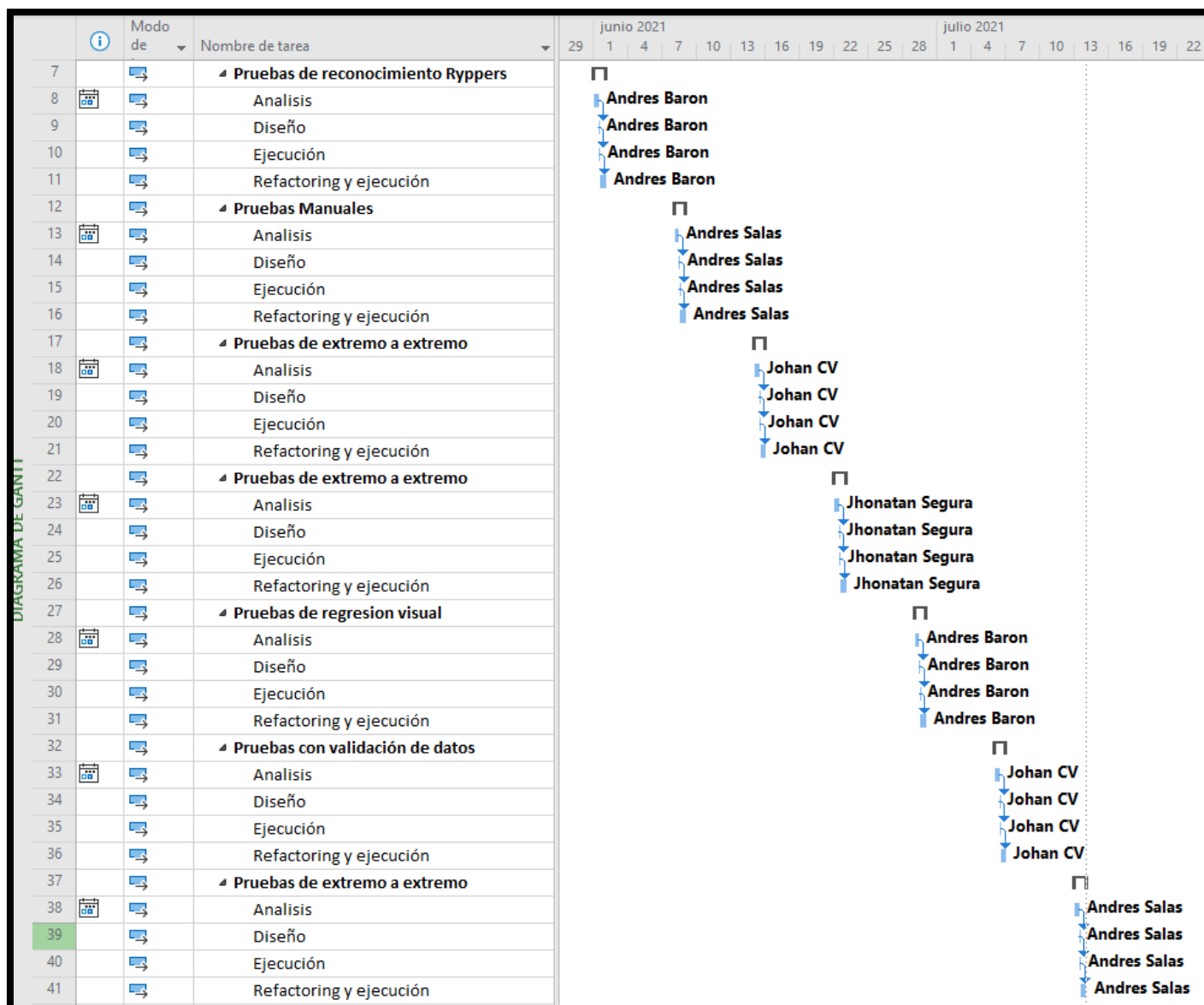


Tabla 3. Distribución de esfuerzos.

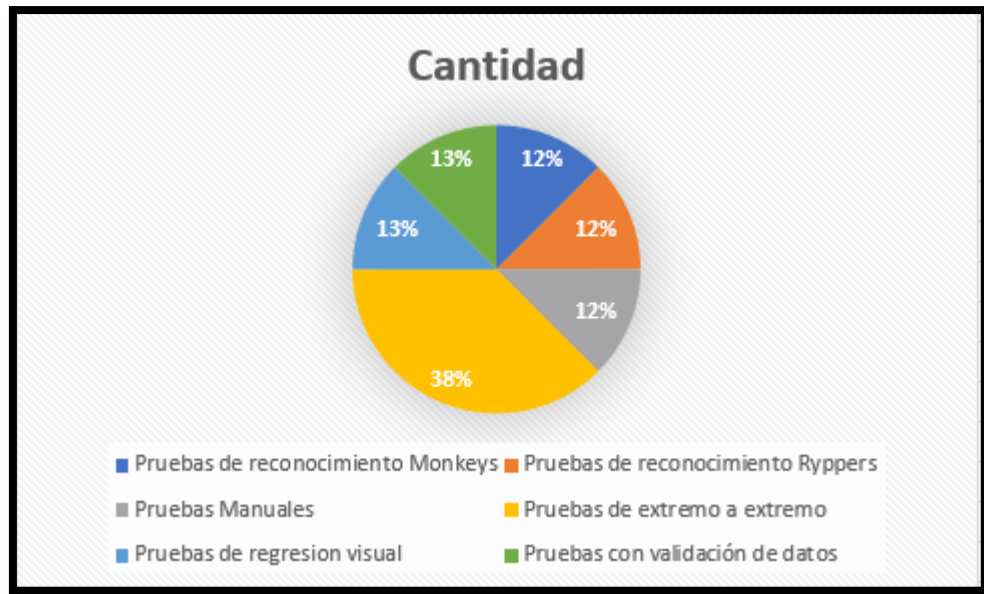


Figura 3. distribución porcentual de esfuerzo.

2.6. Link al video:

<https://youtu.be/E-Euxp5hcNU>