Estrategia de Pruebas Final

1. Aplicación Bajo Pruebas

1.1. Nombre Aplicación: Ghost
1.2. Versión: 3.41.1 y 4.44.0 (pruebas VRT)
1.3. Descripción: Ghost es una plataforma full stack (ofrece tanto una base de datos, un servidor y una aplicación para el cliente) de publicación basada en una pila de tecnologías pertenecientes a Node.js (Javascript), diseñada para equipos que necesitan agilización en crear apps tipo blog.
1.4. Funcionalidades Core:

- Posts: crear (con o sin cronometro), editar, eliminar, listar, visualizar.
- Pages: crear (con o sin cronometro), editar, listar, visualizar.
- Tags: crear, editar, eliminar, listar.
- Navegación (urls): crear, editar, eliminar, listar
- Miembros: crear, editar, eliminar, listar.
- Sitio web: Visualizar.

1.5. Diagrama de Arquitectura: Arquitectura proporcionada por la pagina oficial de Ghost.
1.6. Diagrama de Contexto: Modelo de Contexto
1.7. Modelo de Datos: Modelo de datos realizado en genmymodel.
1.8. Modelo de GUI: Modelo de GUI

2. Contexto de la estrategia de pruebas

2.1. Objetivos:

1. Comprender el funcionamiento general de la aplicación bajo pruebas mediante el uso de pruebas exploratorias manuales, enfocadas en descubrir las funcionalidades de esta.
2. Encontrar errores, crashes, excepciones y defectos en la aplicación bajo pruebas mediante el uso de pruebas de reconocimiento.
3. Evaluar el correcto funcionamiento de múltiples funcionalidades, o escenarios, de la aplicación bajo pruebas mediante el uso de pruebas de extremo a extremo.
4. Analizar las diferencias visuales presentes entre la versión a usar de la aplicación bajo pruebas y una versión más reciente, mediante el uso de pruebas VRT.
5. Evaluar el comportamiento de la aplicación bajo pruebas al presentarse datos inválidos mediante el uso de pruebas de extremo a extremo con diferentes estrategias de generación de datos.

2.2. Duración de la iteración de pruebas:

La estrategia de pruebas comprende un periodo de ocho semanas, y se llevará a cabo desde el 28 de noviembre hasta el 20 de diciembre. Se dedicarán ocho horas semanales por ingeniero.

2.3. Presupuesto de pruebas:

2.3.1. Recursos Humanos

El equipo de pruebas se encuentra compuesto por dos ingenieros automatizadores (testers) seniors con aproximadamente año y medio de experiencia profesional cada uno. Estos ingenieros cuentan con experiencia en herramientas para el desarrollo de pruebas como Kraken, Cypress y RIPuppet. Así mismo, cuentan con experiencia con herramientas de generación de datos como Faker JS y Mockaroo, y experiencia en herramientas de comparación y análisis visual como Resemble JS.

Cada uno de los ingenieros cuenta con ocho horas por semana, por un periodo de ocho semanas, para diseñar, implementar y ejecutar los diferentes escenarios de pruebas, los cuales incluyen pruebas exploratorias, pruebas de reconocimiento, pruebas VRT, pruebas E2E (de extremo a extremo) y pruebas de validación de datos.

2.3.2. Recursos Computacionales

El equipo de pruebas cuenta con los siguientes equipos computacionales:

Sistema Operativo	Windows 10 - Versión 21H2
Memoria RAM	32 GB
Almacenamiento	SSD 1 TB
Procesador	Intel Core i7-9750H - 2.6 GHz
GPU	NVIDIA GeForce GTX 1650 - 4 GB

Sistema Operativo	Windows 10 - Versión 21H2
Memoria RAM	32 GB
Almacenamiento	SSD 1 TB
Procesador	AMD Ryzen 9-3900X – 3.8 GHz
GPU	NVIDIA GeForce RTX 3080 – 10 GB

Sistema Operativo	Pop!_OS 22.04 LTS x86_64
Memoria RAM	8GB
Almacenamiento	SSD 500 GB
Procesador	Intel i5-9300HF (8) @ 2.400GHz
GPU	NVIDIA GeForce GTX 1050 3 GB Max

Sistema Operativo	Pop!_OS 22.04 LTS x86_64
Memoria RAM	8GB
Almacenamiento	SSD 500 GB
Procesador	Intel i7-4790 (8) @ 4.000GHz
GPU	Intel HD Graphics

Se cuenta con total disponibilidad de estos recursos ya que son propios del equipo de pruebas, por lo que pueden ser usados por el tiempo que sea necesario sin restricciones externas a la duración de la estrategia.

2.3.3. Recursos Económicos para la contratación de servicios/personal:

Dado que no se cuenta con presupuesto por parte de la compañía para contratar servicios externos, se ha tomado la decisión de no contratar servicios ni personal externo. De igual manera, el equipo no considera que el apoyo de terceros sea necesario para el correcto desarrollo de la estrategia, pues se cuenta con los conocimientos y recursos necesarios para implementar la estrategia de pruebas exitosamente.

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

Técnicas: Inicialmente, las pruebas exploratorias serán ejecutadas de forma manual, para obtener conocimiento de la aplicación bajo pruebas. Para las demás pruebas (reconocimiento, VRT y extremo a extremo) serán diseñadas por nosotros, pero ejecutadas de forma automatizada, pues ya se tiene conocimiento de cómo funciona el sistema.

Nivel: Las pruebas exploratorias, de reconocimiento, VRT y de extremo a extremo validan el sistema completo por lo que las ubicamos en el nivel de sistema. Así mismo, las pruebas exploratorias y de extremo a extremo se desarrollan desde la perspectiva del usuario, por lo que también cuentan como de aceptación.

Al no contar con acceso al código de la aplicación, no nos es posible definir pruebas unitarias. De igual manera, al no conocer los componentes que integran el sistema, no podemos definir pruebas de integración.

Tipos: Como no tenemos acceso al código realizaremos exclusivamente pruebas de caja negra. También nos enfocaremos únicamente en pruebas funcionales, dado que el alcance de la estrategia no nos permite destinar recursos ni tiempo para pruebas no funcionales.

Nivel	Tipo	Técnica	Objetivo
Sistema	Funcional	Manuales	**1**
Sistema	Caja Negra	Manuales	**1**
Aceptación	Funcional	Manuales	**1**
Aceptación	Caja Negra	Manuales	**1**
Sistema	Funcional	Automatizadas	**2**
Sistema	Caja Negra	Automatizadas	**2**
Sistema	Funcional	Automatizadas	**3**
Sistema	Caja Negra	Automatizadas	**3**
Aceptación	Funcional	Automatizadas	**3**
Aceptación	Caja Negra	Automatizadas	**3**
Sistema	Funcional	Automatizadas	**4**
Sistema	Caja Negra	Automatizadas	**4**
Sistema	Funcional	Automatizadas	**5**
Sistema	Caja Negra	Automatizadas	**5**
Aceptación	Funcional	Automatizadas	**5**
Aceptación	Caja Negra	Automatizadas	**5**

2.5. Distribución de Esfuerzo

Se cuenta para esta estrategia con dos ingenieros automatizadores (testers) con disponibilidad de tiempo de 8 horas semanales cada uno, por un periodo de tiempo de ocho semanas. Los testers harán uso de los equipos de cómputo con el que el equipo de pruebas cuenta para la realización de las tareas de cada semana. Teniendo en cuenta esto, la distribución de trabajo para cada semana será la siguiente:

Semana 1:

Para esta semana se dedicarán todas las horas disponibles de nuestros testers a realizar pruebas exploratorias manuales, con el fin de obtener el mayor conocimiento posible del funcionamiento general de la aplicación bajo pruebas. Los resultados de estas pruebas exploratorias se registrarán en el inventario de pruebas, para tener una guía en el desarrollo de futuras pruebas automatizadas.

Semana 2:

Durante esta semana nuestros testers continuarán realizando pruebas exploratorias manuales, y registrando los resultados en el inventario de pruebas. Sin embargo, ya que en este punto se tiene cierto conocimiento del funcionamiento del sistema, los testers utilizarán sus equipos de cómputo que se encuentren sin utilizar para ejecutar pruebas de reconocimiento. Esto, con el fin de encontrar errores o defectos en las funcionalidades exploradas en la semana pasada. Estas pruebas de reconocimiento serán pruebas de exploración sistemática utilizando la herramienta RIPuppet.

Semana 3:

En la tercera semana, los testers empezarán a diseñar y construir la suite de pruebas de extremo a extremo (E2E) para la aplicación bajo pruebas, con base en el conocimiento adquirido del funcionamiento del sistema y los errores o defectos presentes. Para la construcción de esta suite, se ha decidido emplear la metodología BDT (*Behaviour Driven Testing*) con el fin de facilitar la comunicación entre el equipo de pruebas y el equipo de negocio. Por lo tanto, también se utilizará el patrón *Given-When-Then* al especificar las pruebas E2E, puesto que permite especificar pruebas en un lenguaje sencillo de comprender para personas del lado del negocio. Por último, se ha escogido la herramienta Kraken para ejecutar estas pruebas, ya que es una herramienta sencilla de utilizar que soporta la metodología y patrón de pruebas seleccionados.

A la par que se diseñan y construyen las pruebas E2E, los equipos de cómputo libres continuarán ejecutando la mayor cantidad de pruebas de reconocimiento que el tiempo permita. Los hallazgos de estas pruebas serán tenidos en cuenta para la construcción de las pruebas E2E.

Semana 4:

Para esta semana nuestros testers ejecutarán las pruebas construidas, validarán los resultados de estas y ajustarán la suite de pruebas con base en estos resultados. De igual manera, crearán, ejecutarán y validarán nuevos escenarios que aparezcan durante el trabajo de la semana. En este punto se dejarán de realizar pruebas de reconocimiento, para enfocar el trabajo en la suite de pruebas E2E. Se espera al finalizar esta semana que la suite de pruebas E2E cubra las funcionalidades más importantes de la aplicación bajo pruebas.

Semana 5:

En este punto se cuenta con la suite de pruebas E2E finalizada, la cual cubre las funcionalidades y escenarios más importante de la aplicación bajo pruebas. Por lo tanto, nuestro equipo ha tomado la decisión de comparar la versión de la aplicación que se está usando con una versión más reciente. Esto con el fin de analizar el impacto que tendría sobre la suite de pruebas el migrar a una versión más reciente. Para esto, nuestros testers tomarán los escenarios de prueba de las funcionalidades más importantes, y los adaptarán para que puedan ser ejecutados sobre esta nueva versión (4.44.0). Posteriormente, se construirán pruebas VRT utilizando la herramienta Resemble JS, la cual permite hacer una comparación visual sobre cada paso de los escenarios de prueba de la nueva versión y la versión que se está usando. El análisis de la comparación permitirá evaluar si es viable migrar a una nueva versión, o el impacto sobre la suite de pruebas es demasiado grande.

Semana 6:

Durante las semanas restantes nuestro equipo de testers se enfocarán en crear escenarios de validación de datos de la aplicación bajo pruebas. Esto significa que se busca evaluar como la aplicación responde ante diferentes datos de entrada (válidos, inválidos, de límite). Para esto, durante esta semana, se explorarán herramientas que permitan generar estos datos siguiendo tres estrategias de generación: pool de datos a-priori, pool de datos (pseudo) aleatorio dinámico y escenarios aleatorios. Las herramientas a usar son, respectivamente: Mockaroo, la cual permite generar un gran pool de datos siguiendo unas reglas especificadas por los testers; Kraken-Faker, que permite generar datos pseudoaleatorios en tiempo de ejecución; y Faker js, el cual permite generar datos aleatorios en tiempo de ejecución.

Semana 7:

Durante esta semana, los testers adaptarán la suite de pruebas E2E para que prueben la validación de datos de la aplicación bajo pruebas, siguiendo las tres estrategias de generación de datos. Por lo tanto, se tendrá tres escenarios por cada funcionalidad a probar, una por cada estrategia. Para el caso de pool de datos a-priori, los testers generarán el pool de datos y adaptarán los escenarios para que utilicen este pool. Para las demás estrategias, se ajustarán los escenarios para que hagan uso de las funciones que estas herramientas proveen para la generación de datos. Se espera al final de la semana tener una suite de pruebas E2E para la validación de datos del sistema.

Semana 8:

En la última semana de esta estrategia de pruebas se espera tener los resultados de todas las pruebas realizadas en las semanas pasadas. Esto incluye las pruebas exploratorias manuales, pruebas de reconocimiento, la suite de pruebas E2E, las pruebas VRT y la suite de pruebas E2E para validación de datos. Con estos resultados, el equipo de testers deberá hacer un análisis y reportar los hallazgos al CTO de la compañía, para que estos tomen decisiones informadas. No se planea ejecutar demasiado en esta semana para, en caso de que el plan de trabajo de alguna semana no pueda cumplirse completamente, tener un cierto margen de tiempo en el que se pueda finalizar el trabajo de la estrategia de pruebas.