***Entrega Final  
Pruebas de software***

Contenido

[1. Aplicación Bajo Pruebas 2](#_Toc104728780)

[**1.1.** Nombre Aplicación 2](#_Toc104728781)

[**1.2.** Versiones 2](#_Toc104728782)

[**1.3.** Descripción 2](#_Toc104728783)

[**1.4.** Funcionalidades Core 2](#_Toc104728784)

[**1.5.** Diagrama de Arquitectura 5](#_Toc104728785)

[**1.6.** **Diagrama de Contexto** 6](#_Toc104728786)

[**1.7.** Modelo de Datos 6](#_Toc104728787)

[**1.8.** Modelo de GUI 6](#_Toc104728788)

[2. Contexto de la estrategia de pruebas 7](#_Toc104728789)

[**2.1.** Objetivos 7](#_Toc104728790)

[2.1.1. Objetivo general 7](#_Toc104728791)

[2.1.2. Objetivos específicos 7](#_Toc104728792)

[2.1.3. Descripción 7](#_Toc104728793)

[**2.2.** Duración de la iteración de pruebas 8](#_Toc104728794)

[**2.3.** Presupuesto de pruebas 8](#_Toc104728795)

[2.3.1. Recursos Humanos 8](#_Toc104728796)

[2.3.2. Recursos Computacionales 8](#_Toc104728797)

[2.3.3. Recursos Económicos para la contratación de servicios/personal 9](#_Toc104728798)

[**2.4.** TNT (Técnicas, Niveles y Tipos) de pruebas 9](#_Toc104728799)

[**2.5.** Distribución de Esfuerzo 10](#_Toc104728800)

[3. Anexos 13](#_Toc104728801)

[**3.1.** Anexo A – Enlace a repositorio en GITHUB 13](#_Toc104728802)

[**3.2.** Anexo B – Enlace a video 13](#_Toc104728803)

***Estrategia de Pruebas***

# Aplicación Bajo Pruebas

## Nombre Aplicación

Ghost v4.47.0

## Versiones

Ghost-CLI version: 1.19.2

Ghost version: 4.47.0

npm version: 8.5.0

node version: 16.14.2

## Descripción

Ghost es un sistema manejador de contenidos, también conocido como CMS, cuya funcionalidad consiste en centralizar, por medio de una plataforma, la creación y gestión de contenido para sitios web. La funcionalidad de Ghost es bastante cercana a la de otras herramientas como WordPress o Joomla, que son un poco más conocidas y utilizadas para la creación de blogs.

## Funcionalidades Core

|  |  |  |
| --- | --- | --- |
| 1 | Administración del sitio | **Posts:** Los tipos de datos principales para Ghost son los posts, los cuales permiten publicar el contenido |
| **Multi-idioma:** Ghost permite publicar en cualquier idioma o multi-idiomas |
| **Menú de navegación fácil:** el software tiene una interfaz de usuario simple. |
| **Inyección de código:** se puede agregar un código personalizado al encabezado y pie de página del sitio. |
| **Selección de zona horaria:** la zona horaria se puede seleccionar para aprovechar al máximo la programación y la experiencia de la audiencia. |
| **Marca personalizada:** Ghost no coloca el logotipo de su marca en ninguna parte de su sitio web. |
| **Páginas:** Para contenido estático, diferente a los posts, las páginas permiten la creación de cosas como contactos y descripciones. Ghost permite segmentar las publicaciones debajo de las páginas para que no parezcan una fuente de publicaciones. |
| **Contraseña del sitio:** los usuarios pueden ocultar su sitio web de la vista pública cuando está en construcción. |
| **Cuentas sociales conectadas:** los metadatos de los sitios web de los usuarios se pueden vincular directamente a las cuentas sociales. |
| **Redireccionamientos personalizados:** administre los redireccionamientos 301 y 302 personalizados para detectar direcciones URL antiguas y asignarlas a nuevas ubicaciones. |
| 2 | Editor | Buena Experiencia de escritura |
| Edición a pantalla completa |
| **Soporte Markdown**: permite combinar las funciones de los editores más populares actualmente |
| **Enlaces inteligentes:** Permite resaltar una URL con simplemente pegar su texto |
| **Snippets:** permite crear contenido para ser reutilizado después |
| Tarjetas de imagen |
| Tarjetas de galería |
| **Tarjetas embebidas:** permite añadir tarjetas predeterminadas como Youtube, Twitter, ... |
| **Tarjetas Unsplash:** permite ingresar a la galería de imágenes de fotógrafos profesionales |
| Tarjetas HTML personalizadas |
| Tarjetas Markdown personalizadas |
| 3 | Flujo de trabajo de publicación | URL personalizadas |
| **Etiquetas Taxonomía:** para organizar el contenido de madra dinámica |
| Etiquetas internas |
| Publicar vistas previas |
| Publicaciones programadas |
| Extractos personalizados |
| Publicaciones destacadas |
| Líneas de varios autores |
| Tarjetas de Twitter personalizadas |
| **Contenido gráfico abierto personalizado:** permite ingresar un título personalizado, descripción e imagen de publicación para contenido de redes sociales |
| Soporte de podcasts |
| Archivos de etiquetas y autores |
| Plantillas de publicaciones personalizadas |
| RSS Feeds |
| 4 | Buscar optimización del motor | No se necesitan complementos |
| Marcado y atributos ordenados |
| Site maps automáticos de Google |
| Datos estructurados automáticos |
| **Robots.txt:** Ghost crea automáticamente un archivo robots.txt correctamente formado que se vincula a su mapa del sitio y excluye las carpetas del sistema. |
| **Meta-página personalizada:** Los títulos de página semánticos se crean automáticamente, pero también se pueden personalizar manualmente por página. |
| Limpiar enlaces permanentes |
| Estructuras de URL personalizadas |
| Etiquetas canónicas |
| **Google AMP incorporado:** Habilite las páginas móviles aceleradas en su sitio con un clic para obtener un potencial de clasificación superior en los resultados de búsqueda móvil |
| Pings RPC automáticos |
| **Rendimiento superior:** Pruebas independientes han encontrado que es hasta 20 veces más rápido que WordPress. |
| Servicio de confianza |
| 5 | Membresías y suscripciones | **Registro de miembros:** permite pedir a sus visitantes que ingresen su dirección de correo electrónico para acceder al contenido protegido. |
| Suscripciones recurrentes |
| Miembros de importación/exportación |
| Webhooks de miembros |
| Boletines por correo electrónico |
| Estadísticas de miembros |
| Gestión de miembros |
| 6 | Integraciones | **Aplicaciones y servicios:** Ghost es compatible con casi todas las aplicaciones y servicios de terceros que existen. |
| **Zapier:** Ghost se integra fácilmente con más de 5000 aplicaciones y servicios para activar flujos de trabajo y automatizaciones potentes y sofisticados. |
| **Slack:** envía notificaciones a Slack cada vez que se publique una nueva publicación. |
| **Primer Promotor:** Cree su propio programa de recomendación personalizado con FirstPromoter y ofrezca incentivos a las personas que quieran correr la voz sobre su trabajo. |
| **Integraciones personalizadas:** Al tener control total sobre el front-end, junto con la API JSON de Ghost y una serie de webhooks útiles, crear integraciones personalizadas del lado del cliente es sencillo. |
| 7 | Gestión de equipos | Usuarios del personal |
| Roles y permisos |
| Perfiles personalizados |
| Invitaciones de usuario |
| Suspender usuarios |
| 8 | Api de desarrollador | API RESTful JSON |
| Documentos detallados del desarrollador |
| **CMS headless:** Ghost funciona como un CMS totalmente autónomo donde sea necesario, proporcionando contenido como servicio a cualquier aplicación o infraestructura con una selección de potentes SDK. |
| **Cliente de administración fantasma:** Desde el primer momento, Ghost viene con un cliente de administración Ember.js con todas las funciones; sin embargo, puede desarrollar fácilmente clientes de administración personalizados para que Ghost envíe sus propios datos a la API. |
| **Front-end agnóstico:** Si bien Ghost se envía con una capa de tema Handlebars.js nativa de manera predeterminada, es compatible con cualquier interfaz de usuario, incluidos Gatsby, Jekyll, Hugo o cualquier otro generador de sitios estáticos o aplicación móvil. |
| Adaptadores de almacenamiento personalizados |
| Elección de la base de datos |
| **Webhooks:** Ghost funciona como un CMS totalmente autónomo donde sea necesario, proporcionando contenido como servicio a cualquier aplicación o infraestructura con una selección de potentes SDK. |
| Potente CLI |
| Análisis de compatibilidad de temas |
| Versionamiento |
| 9 | Soporte | Soporte prioritario |
| Soporte comunitario |
| 10 | Seguridad | Pruebas de penetración periódicas |
| SSL automático |
| Permisos estandarizados |
| Validación de datos y serialización |
| Fichas codificadas en todas partes |
| Hashing de contraseña |
| **Prevención SQLi:** Ghost utiliza un generador de consultas ORM + y no genera ninguna de sus propias consultas SQL sin procesar. Ghost no tiene interpolación de variables directamente a cadenas SQL. |
| **Prevención XSS:** Ghost usa cadenas seguras/de escape que se usan en todas partes, incluso y especialmente en todos los asistentes de manillar personalizados que se usan en los temas de Ghost. |
| Protección de fuerza bruta |
| Gestión de dependencias |

Para mayor detalle diríjase a https://ghost.org/feature-index/

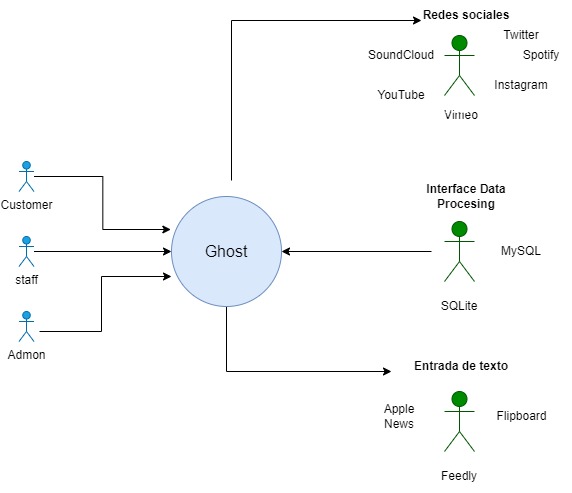
## Diagrama de Arquitectura

Diagrama

Descripción generada automáticamente

Fuente: https://ghost.org/docs/architecture/

## **Diagrama de Contexto**



## Modelo de Datos

El modelo de datos se puede encontrar en la carpeta 00-Documentacion del repositorio, con el nombre de ModeloDatos.png.

## Modelo de GUI

El modelo de GUI se puede encontrar en la carpeta 00-Documentacion del repositorio, con el nombre de ModeloGUI.pdf con los grafos e imágenes.

# Contexto de la estrategia de pruebas

## Objetivos

### Objetivo general

Reducir la probabilidad de que los usuarios encuentren defectos en el sistema encontrando errores críticos y/o comportamientos inesperados de la aplicación Ghost con el fin de retroalimentar al cliente.

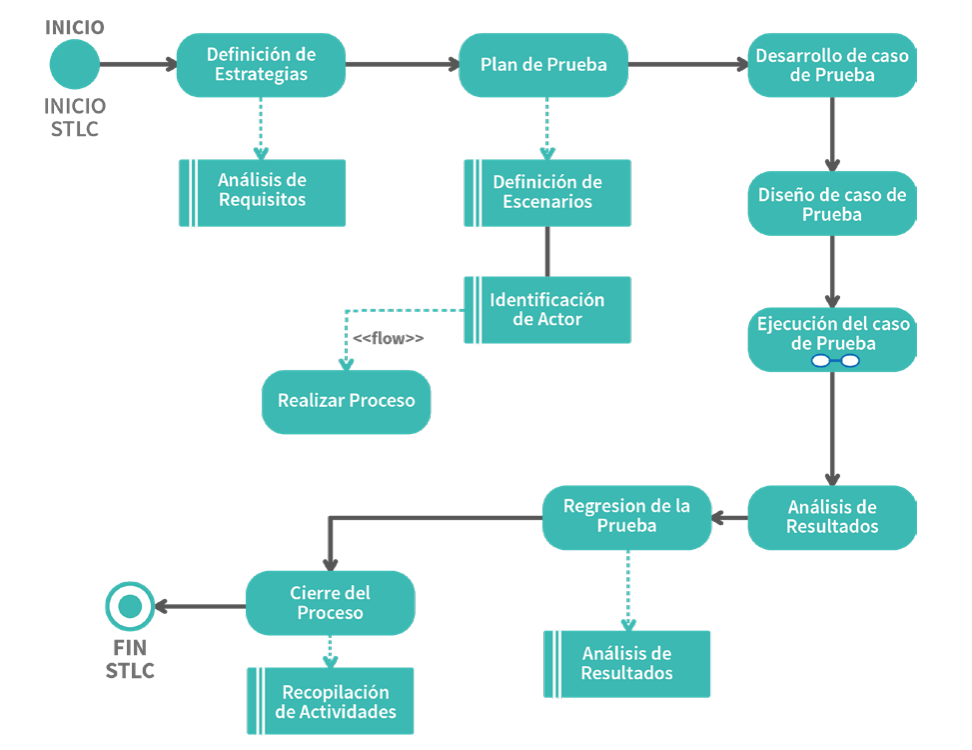
### Objetivos específicos

1. Generar conocimiento sobre la herramienta Ghost a nivel interno del equipo.
2. Identificar escenarios críticos con las pruebas exploratorias para enfocar los esfuerzos de automatización sobre dichos escenarios.
3. Verificar la correcta ejecución de las principales funcionalidades mediante el uso de pruebas de extremo a extremo.
4. Identificar cambios en la interfaz gráfica que puedan ser desfavorables para la ejecución normal de la aplicación.
5. Evaluar el comportamiento del sistema con datos inválidos y en condiciones de frontera, ya sea en formularios o ingreso de texto.
6. Automatizar mediante el uso de APIs, las herramientas que se usarán a lo largo de esta estrategia, para abarcar la mayor cantidad de escenarios posibles y así ser más eficientes a la hora de encontrar errores.

### Descripción

El enfoque de esta estrategia de pruebas será funcional, dada las capacidades que se mencionarán en los próximos puntos. Comenzando con pruebas exploratorias manuales que nos darán un indicio de cómo funciona la aplicación.

Las pruebas exploratorias y el listado de funcionalidades ayudarán a definir los escenarios críticos susceptibles a fallos, y dichos escenarios serán automatizados con las herramientas definidas más adelante, como por ejemplo, pruebas de reconocimiento (Monkey Testing y Ripper). Un resumen de la estrategia se contextualiza en el siguiente diagrama.



Proceso de pruebas. De Balderas Torres, Brenda G. (2018). Estrategias para las pruebas de software que permiten mejorar la calidad en el desarrollo de proyectos de software [Licenciado en Ingeniería en Tecnologías de la Información, Benemérita Universidad Autónoma de Puebla.

## Duración de la iteración de pruebas

Duración del proyecto: 8 Semanas (40 días hábiles), repartidas de la siguiente manera:

|  |  |  |
| --- | --- | --- |
| **Hito** | **Esfuerzo requerido en semanas** | **Recursos** |
| Fase de pruebas manuales | 4 | 4 Testers Senior |
| Fase de pruebas de reconocimiento | 6 | 4 Testers Senior |
| Fase de extremo a extremo o E2E | 2 | 4 Testers Senior |
| Fase de pruebas de regresión visual VRT | 2 | 4 Testers Senior |
| Fase de pruebas de validación de datos | 4 | 4 Testers Senior |

## Presupuesto de pruebas

### Recursos Humanos

Rol: Tester Senior

Disponibilidad: 8 horas semanales/persona

Herramientas utilizadas: Conocimientos en Cypress, RIP, Backstop

### Recursos Computacionales

Para la presente estrategia se cuenta con las siguientes máquinas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Sistema Operativo** | **Procesador** | **Memoria RAM** | **Disco duro (Disponible)** |
| **Máquina 1** | Windows 10 Home | Intel Core i5 | 24 GB | 50 GB |
| **Máquina 2** | Windows 10 Enterprise | Intel Core i7 | 16 GB | 50 GB |
| **Máquina 3** | Macbook Pro M1 | Apple M1 | 8 GB | 50 GB |
| **Máquina 4** | Macbook Pro M1 | Apple M1 | 12 GB | 80 GB |

Software instalado en cada máquina:

* Docker Desktop con container de Ghost 4.47.0
* SQL Lite
* Git
* Cypress 9.5.3
* Node JS 16.14.2
* Npm 8.5.0
* Faker
* Visual Studio Code
* Backstop

### Recursos Económicos para la contratación de servicios/personal

La contratación del Talento Humano será directa y no tercerizada, por tanto, no aplica. No contrataremos servicios externos adicionales.

## TNT (Técnicas, Niveles y Tipos) de pruebas

|  |  |  |  |
| --- | --- | --- | --- |
| **Nivel** | **Tipo** | **Técnica** | **Objetivo** |
| Pruebas de sistema | Pruebas funcionales  Pruebas de caja negra  Pruebas positivas | Pruebas manuales exploratorias | Primer objetivo de los específicos. |
| Pruebas de sistema | Pruebas funcionales  Pruebas de caja negra  Pruebas positivas | Pruebas automatizadas de reconocimiento de Monkey Testing y Rippers con Cypress | Segundo y sexto objetivo de los específicos. |
| Pruebas de sistema | Pruebas funcionales.  Pruebas de caja negra.  Pruebas positivas | Pruebas automatizadas E2E con API de Cypress | Tercer y sexto objetivo de los específicos. |
| Pruebas de integración | Pruebas funcionales  Pruebas de caja negra  Pruebas positivas | Pruebas automatizadas de regresión visual con Backstop | Cuarto y sexto objetivo de los específicos. |
| Pruebas de integración | Pruebas funcionales.  Pruebas de caja negra.  Pruebas positivas | Pruebas automatizadas de validación de datos con Cypress y Faker | Quinto y sexto objetivo de los específicos. |

**Ejecución de las pruebas de reconocimiento**

Se ejecutarán pruebas de reconocimiento desde el rol de administrador para identificar posibles errores en la aplicación mediante el uso de Monkey Testing desarrollado con la herramienta Cypress.

**Ejecución de las pruebas de extremo a extremo (E2E)**

Se ejecutarán pruebas de extremo a extremo a nivel de integración, a través del API de automatización Cypress.

**Ejecución de las pruebas regresión visual (VRT)**

Se ejecutarán pruebas de regresión visual a nivel de sistema, a través del API de automatización Cypress y la librería Backstop.

**Ejecución de las pruebas validación de datos**

Se ejecutarán pruebas de validación de datos a nivel de integración, a través del API de automatización Cypress y el uso de la libraría Faker.

**Consideraciones para las pruebas.**

* No se ejecutarán pruebas Unitarias, ya que no somos responsables del desarrollo del producto.
* No se ejecutarán pruebas de aceptación debido a la restricción del presupuesto y a que no tenemos interacción directa con usuarios reales del sistema, por lo que no se tiene contacto con escenarios reales.
* Se selecciona la API Cypress por sus ventajas al esperar que los componentes se pinten en pantalla y por el gran nivel de experiencia que el equipo de pruebas tiene en ella.

## Distribución de Esfuerzo

A continuación se detalla el esquema de ejecución de esta estrategia, los tiempos estimados y el uso de los recursos para cada entrega

Interfaz de usuario gráfica, Aplicación, Tabla, Escala de tiempo

Descripción generada automáticamente

* **Sprint 1:**

Duración: Dos semanas

En esta primera iteración se realizarán pruebas de exploración manuales, pruebas de reconocimiento automatizadas y pruebas de validación de datos para algunas de las funcionalidades principales, las cuales permitirán al equipo de pruebas familiarizarse con el sistema, identificar sus funcionalidades y el espacio de entradas y estados. A partir de este ejercicio, el equipo podrá definir y documentar los casos y escenarios de prueba que serán objeto de trabajo a lo largo de las iteraciones contempladas.

En esta etapa del proceso, intervendrán los cuatro tester senior durante las ocho horas semanales cada uno. Durante las horas laborales estarán desarrollando los diferentes escenarios de pruebas.

Después de algún tiempo realizando las pruebas manuales se ha priorizado las funcionalidades de Login, Crear Post y Crear Page, en las cuales se realizarán las pruebas de validación de datos.

* **Sprint 2:**

Duración: Dos semanas

Se reportarán los avances del Sprint uno con sus respectivos hallazgos.

En este punto, ya tendremos un mejor conocimiento de la aplicación y la mayoría de las funcionalidades. Por lo tanto definiremos que funcionalidades adicionales se van a probar en los sprint dos, tres y cuatro. En este sprint se continuará con las pruebas de validación de datos a funcionalidades que correspondan a esta semana. Adicionalmente en horarios no hábiles, una de las máquinas estará ejecutando pruebas de reconocimiento de generación aleatoria, además de las pruebas de validación de datos.

Con el fin de darle fuerza a las pruebas de reconocimiento se incorporará una nueva herramienta que se espera sea RIPuppet, donde se desarrollarán y ejecutarán pruebas de exploración sistemática para extraer información del estado actual de la interfaz gráfica, incrementando la posibilidad de encontrar errores. Es importante resaltar que se le desea dar fuerza a estas pruebas por su bajo costo y su gran capacidad para ejecutar un gran número de eventos, permitiéndonos testear un gran espacio en la aplicación.

Para garantizar la consistencia de los escenarios de prueba e identificar pruebas Flaky, se ejecutarán las pruebas de validación de datos un mínimo de cinco veces. Esto se realizará con la automatización de las pruebas y se ejecutará en horarios no hábiles por las máquinas disponibles.

* **Sprint 3:**

Duración: Dos semanas

Se reportarán los avances del Sprint dos con sus respectivos hallazgos.

En esta etapa del proyecto se espera una nueva versión de la aplicación por lo tanto todos los esfuerzos se centrarán se centrarán en realizar pruebas de regresión visual, para encontrar posibles defectos o cambios no esperados en la aplicación. Estas pruebas iniciarán por comprobar las funcionalidades principales y se espera que cubran un 80% de la aplicación. Con esto se espera cubrir el conjunto de impacto de los cambios en la última versión.

Para la ejecución de estas pruebas se espera automatizar la toma de pantallazos mediante el uso de Cypress y se utilizará la herramienta BackStop para la comparación de imágenes.

El cliente nos ha manifestado que no se espera un cambio drástico en la apariencia de la aplicación, por lo cual se manejará un umbral de desajuste bajo, el cual se ira ajustando de acuerdo con la ejecución de las pruebas para obtener información valiosa cuando la aplicación resalte una discordancia.

* **Sprint 4:**

Duración: Dos semanas

Se reportarán los avances del Sprint tres con sus respectivos hallazgos.

En la última etapa del proceso, realizaremos las pruebas de extremo a extremo con el objetivo de probar el correcto funcionamiento de funcionalidades completas. Dado que es una de las etapas más costosas del proceso de pruebas, solamente se ejecutarán en esta etapa y se espera un cubrimiento del 40% de las funcionalidades de la aplicación, priorizadas según los hallazgos del Sprint 2.

Al igual que en el sprint 2, para garantizar la consistencia de los escenarios de prueba e identificar pruebas Flaky, se ejecutarán las pruebas E2E un mínimo de cinco veces, procurando hacer uso de horarios no hábiles.

También se ejecutarán pruebas de reconocimiento tanto de exploración sistemática como de generación aleatoria, buscando en cada momento encontrar la mayor cantidad de errores posibles.

Finalizando este proceso, se generará un reporte final con los hallazgos de las pruebas y se entregará un repositorio con los problemas detectados para hacer las correcciones finales a la aplicación.

Se espera una retroalimentación con el equipo del desarrollo a favor del mejoramiento continuo.

Nota:

La distribución de esfuerzos para esta estrategia se basa en el patrón de Pirámide de Automatización. En el cual se busca optimizar los esfuerzos dedicados a cada actividad de acuerdo con los resultados que para este caso serían fallas en el sistema. Si contabilizamos los tiempos dedicados a cada tipo de prueba tenemos:

Pruebas exploratorias manuales: 4 semanas

Pruebas E2E: 2 semanas

Pruebas VRT: 2 semanas

Pruebas de validación de datos: 4 semanas

Pruebas de reconocimiento: 6 semanas

De esta manera la distribución de esfuerzos puede verse de la siguiente manera:

Diagrama

Descripción generada automáticamente

# Anexos

## Anexo A – Enlace a repositorio en GITHUB

A continuación, se comparte el enlace al repositorio de código fuente e insumos solicitados en la presente estrategia de pruebas:

<https://github.com/AAlbaB/ProyectoFinal_Pruebas>

## Anexo B – Enlace a video

A continuación, se comparte el enlace al video en el cual se expone la estrategia de pruebas: