Estrategia de Pruebas

1. Aplicación Bajo Pruebas

1.1. Nombre Aplicación:

Ghost Admin

1.2. Versión:

4.42.0

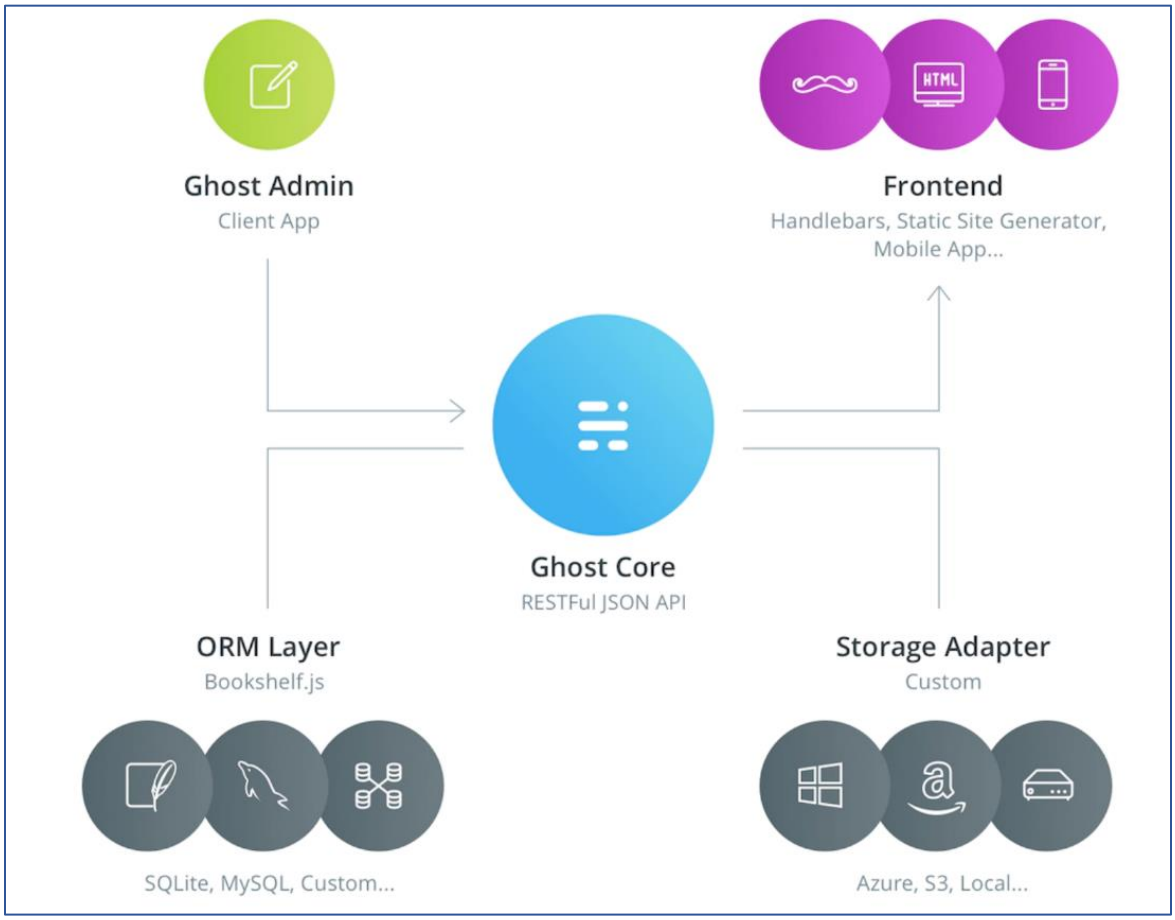
1.3. Descripción:

Ghost es un sistema gestor de contenidos (CMS) gratuito y open source. Está escrito en JavaScript y diseñado para simplificar el proceso de publicación en línea de blogs.

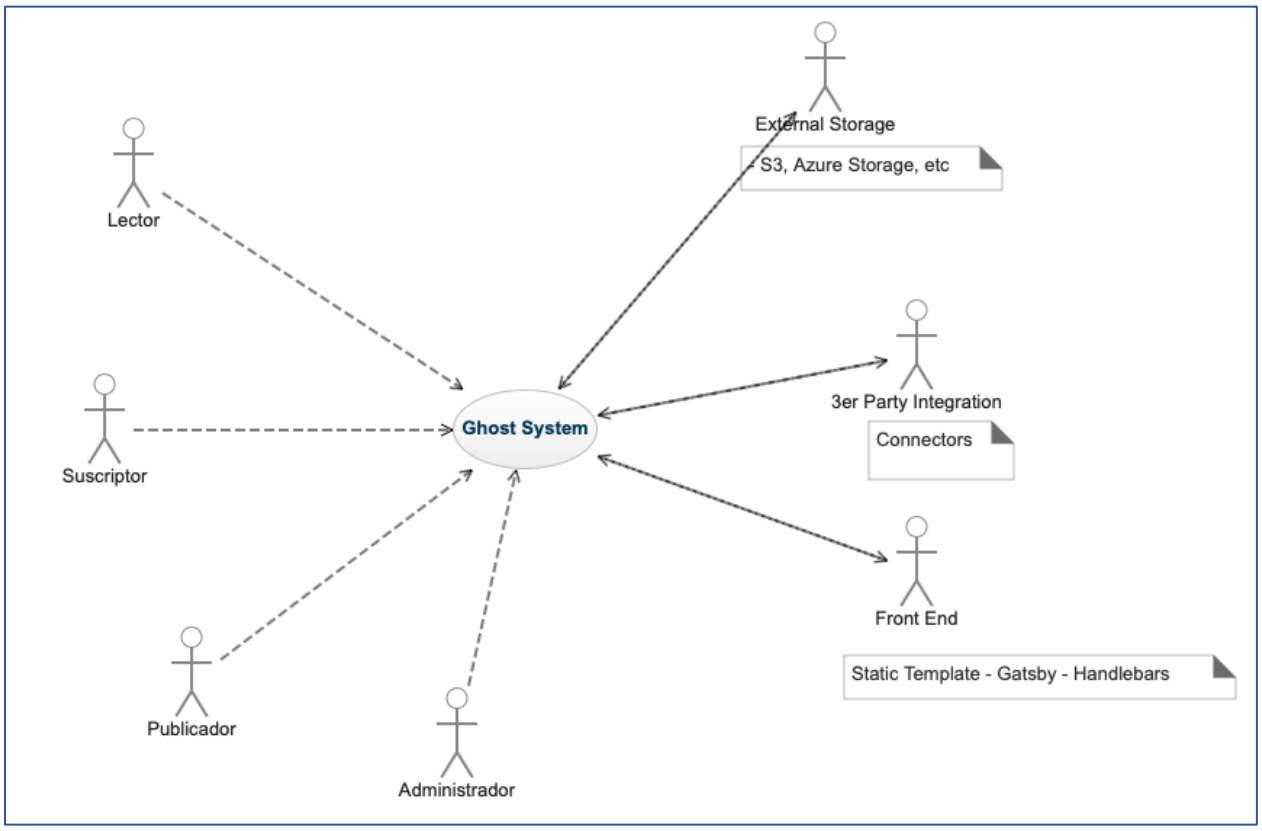
1.4. Funcionalidades Core:

- Crear Tag
- Editar Tag
- Eliminar Tag
- Listar Tags
- Agregar miembro
- Editar miembro
- Listar miembros
- Eliminar miembro
- Crear Post
- Listar Post
- Editar Post
- Eliminar Post
- Previsualizar Post
- Buscar en Ghost
- Modificar perfil
- Agregar página
- Editar página
- Eliminar página

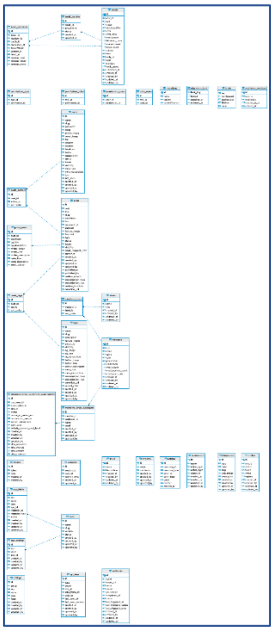
1.5. Diagrama de Arquitectura:



1.6. Diagrama de Contexto:



1.7. Modelo de Datos:

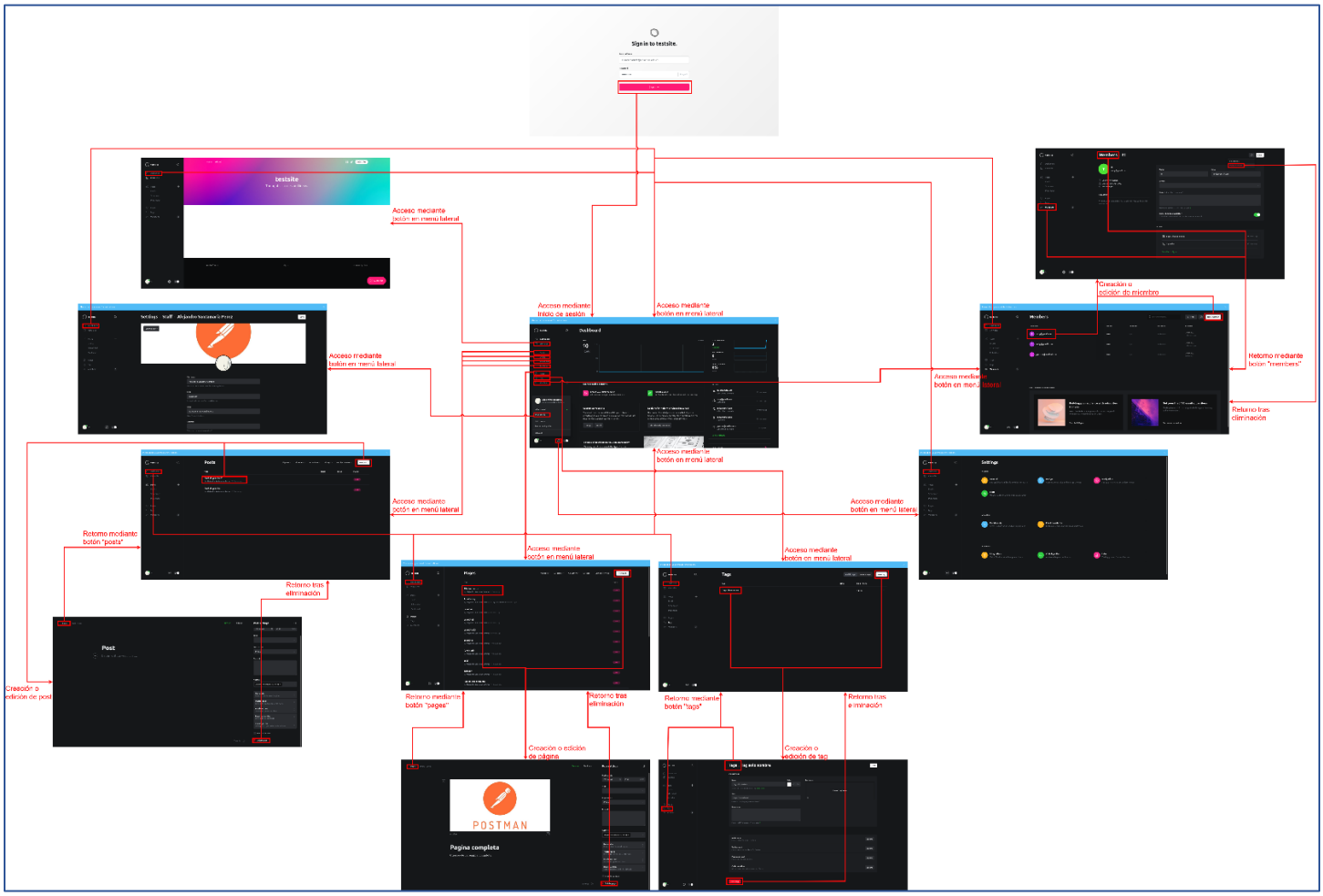


Nota:

Dado el tamaño de la imagen, se expone a continuación un enlace mediante el cual se puede acceder al modelo de datos.

Modelo de datos

1.8. Modelo de GUI:



Nota:

Se simplificaron algunas relaciones que se repiten entre vistas, principalmente las de los botones ubicados en el menú lateral; esto con el fin de evitar saturar el modelo.

Dado el tamaño de la imagen, puede resultar complejo para el lector detallarla; por tanto, se almacenan las imágenes en una carpeta alojada en OneDrive, donde se encuentra el diagrama tanto en formato png como svg. Dicha carpeta puede ser accedida mediante este link por cualquier usuario con cuenta de la Universidad de los Andes; debido a restricciones administrativas, este acceso no puede ser público para usuarios no identificados.

2. Contexto de la estrategia de pruebas

2.1. Objetivos:

- **Mejorar pruebas unitarias:** Revisar las pruebas unitarias existentes, y aumentar el porcentaje de cobertura, principalmente (aunque no exclusivamente) mediante la creación de casos borde y casos de excepción.
- **Mejorar la comprensión de Ghost:** Mediante pruebas de exploración manuales, mejorar el entendimiento de las funcionalidades de Ghost, y realizar una primera búsqueda de issues.
- **Crear pruebas e2e:** Crear pruebas de extremo a extremo usando diferentes escenarios con diferentes técnicas de generación de datos, como pool de datos a priori, y generación aleatoria.
- **Implementar VRT:** Realizar la implementación de pruebas de regresión visual, que podrán ser ejecutadas cada vez que se genere una nueva versión de la aplicación bajo pruebas.
- **Ejecutar pruebas de reconocimiento:** Ejecutar pruebas mediante Monkeys y Rippers sobre la aplicación, para identificar series de eventos que no han sido probados en las pruebas e2e, y generan excepciones o crashes.

2.2. Duración de la iteración de pruebas:

En este documento se planteará una estrategia de pruebas con duración total de 8 semanas.

2.3. Presupuesto de pruebas:

2.3.1. Recursos Humanos

- Cuatro testers senior (8h semanales/persona).

2.3.2. Recursos Computacionales

Para la ejecución de esta estrategia de pruebas, se cuenta con los computadores personales de cada uno de los recursos humanos disponibles. Estos recursos computacionales estarán disponibles en un principio durante el mismo tiempo que los testers (8h semanales). Adicionalmente, el tiempo que cada tester disponga a su discreción para ejecutar pruebas que no requieran ningún tipo de intervención humana.

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

Para esta estrategia no se cuenta con recursos económicos adicionales a los encargados de costear el costo del personal y los recursos computacionales descritos previamente.

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

Nivel	Tipo	Técnica	Objetivo
Unidad	Funcional/Caja blanca/Positiva	Automatizado / Unit testing	Mejorar pruebas unitarias
Unidad	Funcional/Caja blanca/Negativa	Automatizado / Unit testing	Mejorar pruebas unitarias
Sistema	Funcional/Caja negra/Positiva	Manual / Pruebas exploratorias	Mejorar la comprensión de Ghost
Sistema	Funcional/Caja negra/Negativa	Manual / Pruebas exploratorias	Mejorar la comprensión de Ghost
Sistema	Funcional/Caja negra/Positiva	Automatizado / e2e testing	Crear pruebas e2e
Sistema	Implementar VRT	Automatizado / e2e testing	Crear pruebas e2e
Sistema	Funcional/Caja negra/Positiva	Automatizado / VRT	Implementar VRT
Sistema	Funcional/Caja negra/Negativa	Automatizado / VRT	Implementar VRT
Sistema	Funcional/Caja negra/Negativa	Automatizado / Monkey testing	Ejecutar pruebas de reconocimiento

2.5. Distribución de Esfuerzo

2.5.1. Semana 1

Durante la primera semana, se propone realizar dos tareas:

- En primer lugar, realizar pruebas de exploración manuales sobre la aplicación, con el objetivo de encontrar los primeros issues, además de asegurar que los 4 ingenieros comprendan de mejor manera el funcionamiento de Ghost.
- Como segunda tarea, se propone realizar lo ajustes necesarios a la herramienta Kraken para que las pruebas de extremo a extremo (a crear en próximas semanas) sean capaces de tomar pantallazos en cada paso. Esta tarea incluye crear algunos escenarios de prueba, necesarios para garantizar el correcto funcionamiento de los ajustes realizados.

La decisión de priorizar la adaptación de Kraken se debe a que al revisar el repositorio de Ghost, se evidencia que semanalmente se generan entre una y tres versiones, por tanto, se busca habilitar cuanto antes la posibilidad de realizar pruebas de regresión visual al momento de generar una nueva versión.

Adicionalmente, se reserva una hora en el tiempo de cada ingeniero para el reporte de issues encontrados.

La siguiente tabla muestra gráficamente la distribución de esfuerzo propuesta para esta primera semana:

	Hora 1	Hora 2	Hora 3	Hora 4	Hora 5	Hora 6	Hora 7	Hora 8
Ingeniero 1								
Ingeniero 2								
Ingeniero 3								
Ingeniero 4								

	Pruebas de exploración manual
	Adaptación de Kraken
	Reporte de Issues

2.5.2. Semana 2

Para la segunda semana de la estrategia de pruebas, se continúa con la priorización de habilitar cuanto antes la realización de pruebas VRT; para esto, se propone dedicar toda la semana a la realización de pruebas de extremo a extremo para los módulos de post y tags, que son los módulos independientes más grandes de la aplicación. Los datos a usar en estas pruebas serán generados aleatoriamente en tiempo de ejecución.

Adicionalmente, se reserva una hora en el tiempo de cada ingeniero para el reporte de issues encontrados.

La siguiente tabla muestra gráficamente la distribución de esfuerzo propuesta para esta segunda semana:

	Hora 1	Hora 2	Hora 3	Hora 4	Hora 5	Hora 6	Hora 7	Hora 8
Ingeniero 1								
Ingeniero 2								
Ingeniero 3								
Ingeniero 4								

	Creación de pruebas de extremo a extremo
	Reporte de Issues

2.5.3. Semana 3

Durante la semana 3, se propone continuar con el trabajo de crear pruebas de extremo a extremo iniciado en la semana 2. Para esta semana, se crearán las pruebas faltantes del resto de módulos de la aplicación.

Adicionalmente, se reserva una hora en el tiempo de cada ingeniero para el reporte de issues encontrados.

La siguiente tabla muestra gráficamente la distribución de esfuerzo propuesta para esta tercera semana:

	Hora 1	Hora 2	Hora 3	Hora 4	Hora 5	Hora 6	Hora 7	Hora 8
Ingeniero 1								
Ingeniero 2								
Ingeniero 3								
Ingeniero 4								

	Creación de pruebas de extremo a extremo
	Reporte de Issues

2.5.4. Semana 4

Tras la creación de las pruebas de extremo a extremo para toda la aplicación, se puede iniciar la realización de pruebas de regresión visual, por tanto, durante la semana 4, y a partir de ésta, se reservan 3 horas del tiempo de uno de los ingenieros para la realización de estas pruebas cada vez que se cree una nueva versión de Ghost (como se mencionó previamente, la frecuencia suele ser de entre una y tres versiones nuevas por semana).

Sumado a esto, el objetivo principal de esta semana será la creación de test unitarios para mejorar el porcentaje de cobertura del código base de la aplicación. Sería ideal alcanzar un 80% o más de cobertura.

Adicionalmente, se reserva una hora en el tiempo de cada ingeniero para el reporte de issues encontrados.

La siguiente tabla muestra gráficamente la distribución de esfuerzo propuesta para esta cuarta semana:

	Hora 1	Hora 2	Hora 3	Hora 4	Hora 5	Hora 6	Hora 7	Hora 8
Ingeniero 1								
Ingeniero 2								
Ingeniero 3								
Ingeniero 4								

	Creación de pruebas unitarias
	Ejecución de pruebas VRT
	Reporte de Issues

2.5.5. Semana 5

Durante la semana 5, se propone la implementación de pruebas de reconocimiento con monkeys y Rippers; tras la implementación, la ejecución de estas pruebas se ejecutará en horas en que los recursos computacionales no se encuentren en uso; mientras que las horas de los ingenieros serán utilizadas en la tarea de continuar la creación de test unitarios iniciada en la semana 4.

Adicionalmente, se reserva una hora en el tiempo de cada ingeniero para el reporte de issues encontrados, y tres horas de tiempo de un ingeniero para la realización de VRT sobre las versiones que se generen en la semana.

La siguiente tabla muestra gráficamente la distribución de esfuerzo propuesta para esta quinta semana:

	Hora 1	Hora 2	Hora 3	Hora 4	Hora 5	Hora 6	Hora 7	Hora 8
Ingeniero 1								
Ingeniero 2								
Ingeniero 3								
Ingeniero 4								

	Creación de pruebas unitarias
	Implementación de Monkeys y Rippers
	Ejecución de pruebas VRT
	Reporte de Issues

2.5.6. Semana 6

Tras la ejecución de pruebas mediante Monkeys y Rippers en la semana 5, la semana 6 será dedicada a complementar los escenarios de pruebas de extremo a extremo existentes, teniendo en cuenta los casos explorados por las pruebas de reconocimiento, principalmente aquellos casos en los cuales se encontró algún issue (en caso de ser encontrado alguno).

Adicionalmente, se reserva una hora en el tiempo de cada ingeniero para el reporte de issues encontrados, y tres horas de tiempo de un ingeniero para la realización de VRT sobre las versiones que se generen en la semana.

La siguiente tabla muestra gráficamente la distribución de esfuerzo propuesta para esta sexta semana:

	Hora 1	Hora 2	Hora 3	Hora 4	Hora 5	Hora 6	Hora 7	Hora 8
Ingeniero 1								
Ingeniero 2								
Ingeniero 3								
Ingeniero 4								

	Complementación de escenarios e2e
	Ejecución de pruebas VRT
	Reporte de Issues

2.5.7. Semana 7

Teniendo ya una base sólida de pruebas de extremo a extremo, se propone utilizar las semanas 7 y 8 para complementarlas; hasta esta semana, el funcionamiento de estas pruebas se basaba en la generación de datos aleatorios, entonces, se implementará la posibilidad de ejecutarlas con pool de datos a priori. Durante la semana 7 se hará esta implementación para los escenarios relacionados a tags y posts.

Adicionalmente, se reserva una hora en el tiempo de cada ingeniero para el reporte de issues encontrados, y tres horas de tiempo de un ingeniero para la realización de VRT sobre las versiones que se generen en la semana.

La siguiente tabla muestra gráficamente la distribución de esfuerzo propuesta para esta séptima semana:

	Hora 1	Hora 2	Hora 3	Hora 4	Hora 5	Hora 6	Hora 7	Hora 8
Ingeniero 1								
Ingeniero 2								
Ingeniero 3								
Ingeniero 4								

	Implementación pool de datos a priori
	Ejecución de pruebas VRT
	Reporte de Issues

2.5.8. Semana 8

Durante esta última semana, se finalizará el proceso iniciado en la semana 7, y se hará la implementación de pool de datos a priori para las pruebas de extremo a extremo que aún hagan falta.

Adicionalmente, se reserva una hora en el tiempo de cada ingeniero para el reporte de issues encontrados, y tres horas de tiempo de un ingeniero para la realización de VRT sobre las versiones que se generen en la semana.

La siguiente tabla muestra gráficamente la distribución de esfuerzo propuesta para esta séptima semana:

	Hora 1	Hora 2	Hora 3	Hora 4	Hora 5	Hora 6	Hora 7	Hora 8
Ingeniero 1								
Ingeniero 2								
Ingeniero 3								
Ingeniero 4								

	Implementación pool de datos a priori
	Ejecución de pruebas VRT
	Reporte de Issues