Equipo 24

Andrés David Duque Cadena
Juan Andrés López Motta

Estrategia de Pruebas #1

Aplicación Bajo Pruebas

Nombre de la aplicación: Ghost

Versión: 5.82.11 **Versión de node: 20.12.2**

Descripción

Ghost es una plataforma de publicación y blogs, diseñada para ser simple pero potente. A diferencia de otras plataformas como WordPress, Ghost se enfoca exclusivamente en las necesidades de los bloggers y escritores, ofreciendo una experiencia de usuario limpia y sin distracciones. Esto permite que los creadores de contenido se centren en escribir y publicar sin tener que lidiar con configuraciones complicadas o plugins innecesarios.

La aplicación Ghost se destaca por su editor minimalista, que utiliza una interfaz de "Markdown", permitiendo una escritura y edición eficiente y sin complicaciones. Además, ofrece características robustas para SEO (optimización para motores de búsqueda) y opciones de monetización integradas, como la capacidad de iniciar suscripciones pagadas directamente a través de la plataforma.

Funcionalidades core

- **API de Contenido y API de Administración**:
    - **API de Contenido**: Permite acceso de solo lectura para entregar contenido publicado al mundo. Está diseñada para ser completamente cachable
    - **API de Administración**: Permite la creación y actualización de contenido con acceso de lectura y escritura, ofreciendo autenticación basada en roles
    - Ambas APIs permiten una integración eficiente con aplicaciones externas y flujos de trabajo editorial personalizados
- **Almacenamiento y Datos**:
    - Ghost utiliza Bookshelf.js ORM por defecto, soportando SQLite3 para desarrollo y MySQL para producción
    - Permite el uso de adaptadores de almacenamiento personalizados para extender la gestión de archivos
- **CLI de Ghost**:
    - Un conjunto de utilidades CLI para orquestar componentes y mantener el sistema actualizado
- **SDK de JavaScript**:
    - Incluye un cliente de API y un SDK para facilitar la autenticación y el acceso a los datos, integrándose fácilmente con desarrollos en JavaScript
- **Webhooks**:
    - Permite notificar a servicios externos sobre cambios o actualizaciones en el contenido, integrando Ghost con otras herramientas de automatización y distribución de contenido

- **Control de Versiones**:

    - Ghost mantiene una estabilidad en sus APIs con cambios mínimos entre versiones mayores, permitiendo dependencias seguras en producción
- **Ghost-Admin**:
    - Una interfaz de administración avanzada y separada del API core de Ghost, optimizada para editores y desarrolladores, con capacidades completas para gestionar el flujo de trabajo editorial
- **Front-End**:
    - Ghost actúa como un CMS completamente agnóstico del front-end, compatible con cualquier marco de trabajo de front-end o generador de sitios estáticos
    - Incluye una capa de tema Handlebars.js servida por un servidor Express.js, pero también permite desarrollos front-end personalizados, incluso para aplicaciones móviles o nativas no basadas en la web

Diagrama de la arquitectura

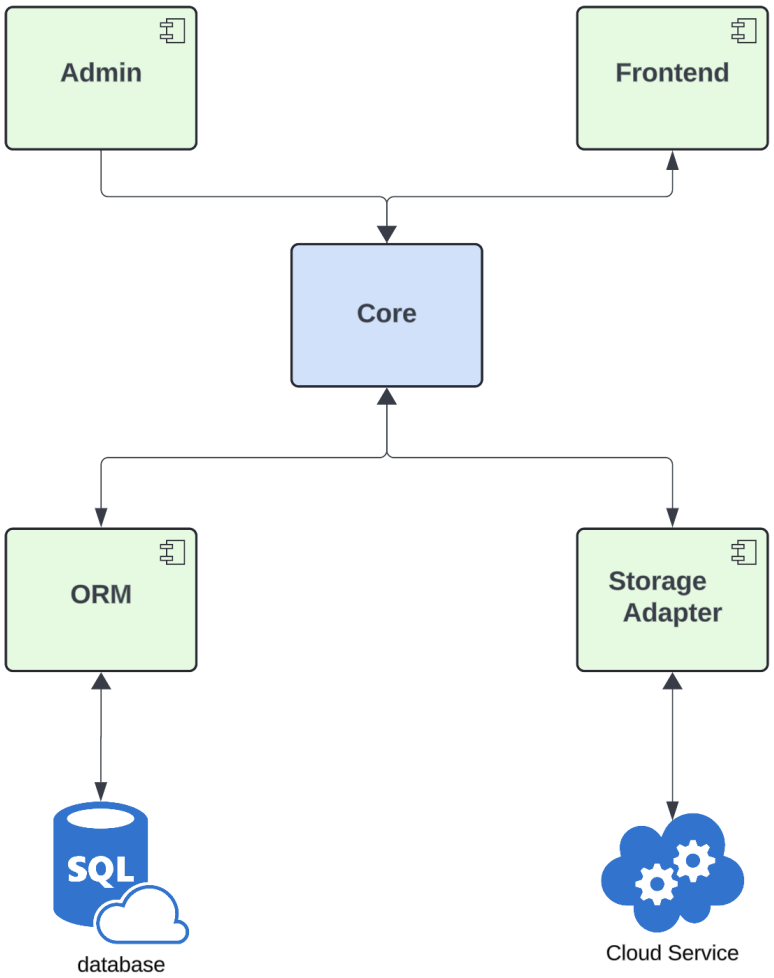
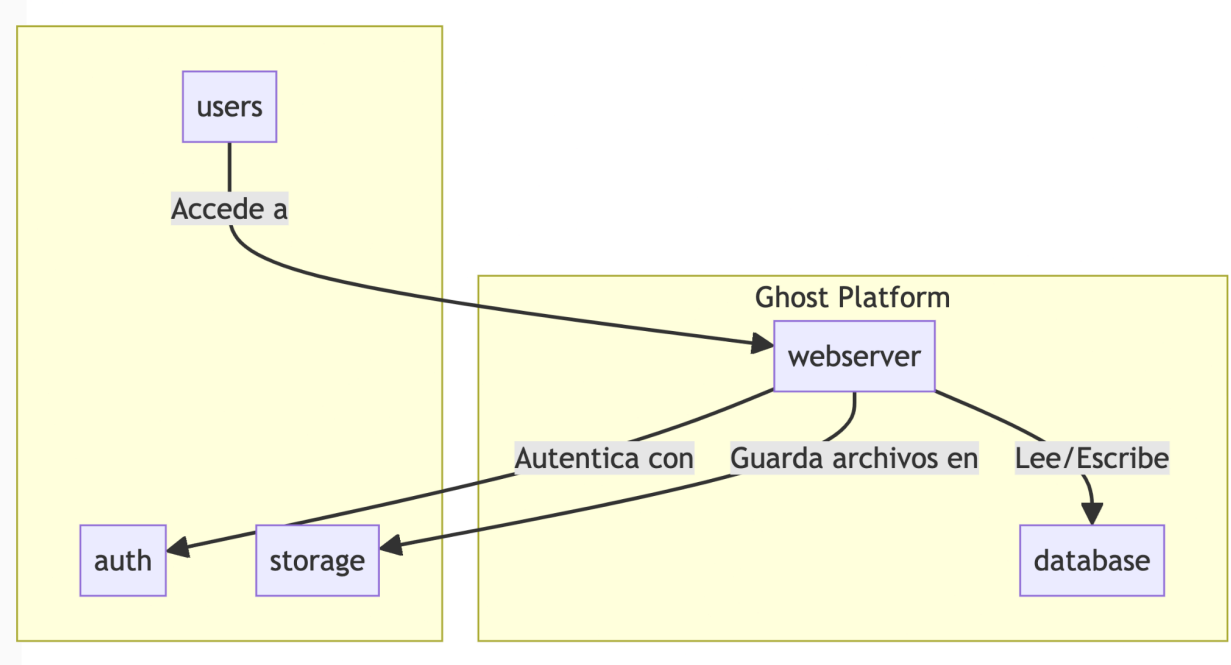


Diagrama de contexto

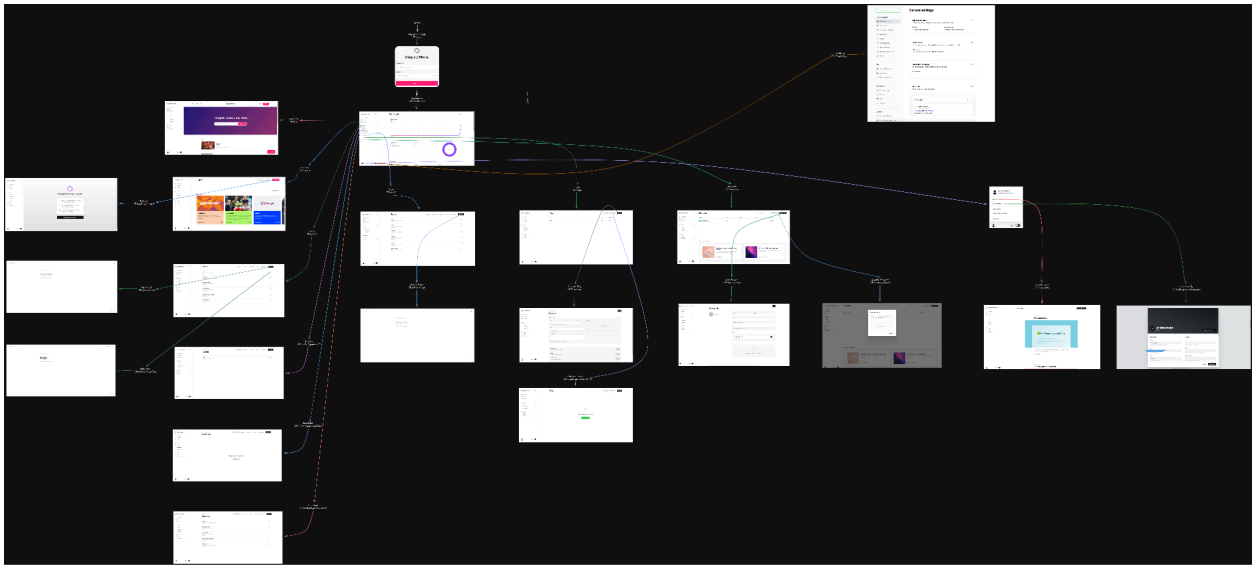


Modelo de datos

El link donde podrás ver la imagen en tamaño completo y con buena resolución es:

https://drive.google.com/file/d/1M3_x0Wts6a-X7GCEsDNbxwJxzDIhDppf/view?usp=sharing

Modelo GUI



El link donde podrás ver la imagen en tamaño completo y con buena resolución es:
https://drive.google.com/drive/u/0/folders/1o6MlDn693TyT0_UXyudS2xrlGBWJx1FM

Contexto de la Estrategia de Pruebas

Objetivos

- Detectar la mayor cantidad de bugs y defectos posibles en el software para asegurar su calidad y funcionamiento correcto.
- Garantizar que todas las áreas críticas del software estén cubiertas por las pruebas, incluyendo pruebas manuales, de reconocimiento, de extremo a extremo (E2E), de regresión visual (VRT) y de validación de datos.
- Validar que el software sea estable y funcione correctamente en diferentes escenarios y bajo distintas condiciones de uso.
- Utilizar eficientemente el tiempo y los recursos disponibles, maximizando la productividad del equipo de testers y aprovechando los recursos computacionales.
- Implementar pruebas automatizadas para aumentar la eficiencia del proceso de pruebas y reducir el tiempo necesario para la validación del software.
- Asegurar que todas las integraciones entre diferentes módulos y componentes del software funcionen correctamente.
- Entregar un producto de alta calidad que cumpla con los requisitos y expectativas del cliente.
- Evaluar las herramientas y técnicas utilizadas en el proceso de pruebas para determinar sus pros y contras, así como su efectividad en la identificación de bugs y en la mejora de la calidad del software.

Duración de la iteración de pruebas

- Desde el 01/04/2024 hasta el 27/05/2024

- 8 semanas (256 horas de trabajo humano)

Presupuesto de pruebas

Recursos humanos

Para desarrollar un presupuesto de pruebas que se ajuste al contexto del proyecto, consideremos los recursos humanos disponibles (cuatro testers junior, cada uno con 8 horas semanales durante 8 semanas) y los recursos computacionales que podamos necesitar. Dado que no se cuenta con recursos monetarios para contratar servicios externos, el enfoque estará en maximizar el uso de herramientas y plataformas de prueba internas o gratuitas. Aquí está un desglose del presupuesto de pruebas:

Recursos Humanos

- 4 testers junior
- 8 horas por semana cada uno
- Duración total: 8 semanas

Total de horas disponibles

- 4 testers × 8 horas/semana × 8 semanas=256 horas

Técnicas, niveles y tipos de prueba

Técnicas de Pruebas

1. Pruebas Manuales
    - Exploratorias
    - Basadas en casos de uso
2. Pruebas Automatizadas
    - Reconocimiento
    - End-to-End (E2E)
    - VRT
    - Validación

Niveles de Pruebas

1. Sistema
2. Aceptación

Tipos de Pruebas

1. Pruebas Funcionales
    - Validan que cada función del software opere conforme a los requisitos especificados.
    - Incluyen pruebas de unidad, integración y sistema.
2. Pruebas No Funcionales
    - Evalúan aspectos como rendimiento, usabilidad, seguridad y compatibilidad.
    - Ejemplos: pruebas de carga, pruebas de estrés, pruebas de seguridad.
3. Pruebas de Regresión
    - Aseguran que nuevas modificaciones no introduzcan errores en las funcionalidades existentes.

- Uso de suites de pruebas automatizadas para revalidar funcionalidades.

4. Pruebas de Reconocimiento Visual
    - Validación de la interfaz de usuario para detectar diferencias visuales involuntarias.
5. Pruebas Exploratorias
    - Realizadas manualmente por los testers para descubrir errores que no están cubiertos por casos de prueba predefinidos.
    - Basadas en el conocimiento y experiencia del tester.
6. Pruebas de Validación de Datos
    - Verificación de la integridad y exactitud de los datos en la aplicación.
    - Incluyen pruebas de base de datos y validación de datos a lo largo de diferentes estados y transacciones.

Distribución de esfuerzo

1. **Diseño de Estrategia de Pruebas** (20 horas)
    - Desarrollo de la estrategia de pruebas y planificación detallada.
2. **Preparación de Casos de Prueba** (40 horas)
    - Creación de casos de prueba manuales y automatizados.
    - Definición de escenarios de prueba de reconocimiento, E2E, VRT y validación de datos.
3. **Configuración de Herramientas y Entorno de Pruebas** (20 horas)
    - Instalación y configuración de herramientas de automatización (e.g., Cypress).
    - Configuración del entorno para pruebas de regresión visual.
4. **Ejecución de Pruebas Manuales** (52 horas)

- Ejecución de pruebas exploratorias y documentar los resultados.

5. **Automatización de Pruebas** (44 horas)
    - Desarrollo y ejecución de scripts de pruebas automatizadas.
6. **Pruebas de Reconocimiento y E2E** (38 horas)
    - Ejecución de pruebas de reconocimiento y de extremo a extremo.
7. **Pruebas de Regresión Visual** (30 horas)
    - Configuración y ejecución de pruebas de regresión visual.
8. **Validación de Datos** (22 horas)
    - Ejecución de escenarios de validación de datos.

Actividad	Horas Asignadas
Diseño de Estrategia de Pruebas	20
Preparación de Casos de Prueba	40
Configuración de Herramientas y Entorno	20
Ejecución de Pruebas Manuales	52
Automatización de Pruebas	44
Pruebas de Reconocimiento y E2E	38
Pruebas de Regresión Visual	30
Validación de Datos	22
Total	256

Semana 1 y 2

- diseño de estrategias de pruebas
- preparación de casos de pruebas

Semana 3 y 4

- configuración de herramientas de entorno
- Ejecución de pruebas manuales

Semana 5

- Automatización de pruebas

Semana 6

- Pruebas de reconocimiento E2E

Semana 7

- Pruebas de regresión visual

Semana 8

- Validación de datos

Justificación estrategia de pruebas

La estrategia de pruebas definida para el proyecto de automatización se justifica plenamente por varios factores clave. Primero, se asegura una cobertura completa y exhaustiva al incluir pruebas manuales, automatizadas, de reconocimiento visual, de extremo a extremo (E2E) y de validación de datos. Esta variedad de pruebas garantiza una verificación integral del sistema, permitiendo identificar y corregir defectos en diferentes niveles de la aplicación, desde unidades individuales hasta la integración completa del sistema.

Además, la estrategia maximiza el uso de recursos humanos y tecnológicos disponibles. Con un equipo de cuatro testers junior dedicando 8 horas semanales durante 8 semanas, se ha definido la distribución de tareas entre pruebas manuales y automatizadas para optimizar el tiempo y la experiencia del equipo. Esto incrementa la eficiencia y permite que cada tester se enfoque en áreas específicas según su fortaleza.

La reducción de riesgos y el aseguramiento de la calidad son otros de los factores cruciales. Las pruebas de regresión visual y de validación de datos están diseñadas para detectar cambios no deseados y asegurar la integridad de los datos en el sistema, reduciendo riesgos y garantizando que en las modificaciones en el software no introduzcan nuevos errores.

La estrategia también destaca por su adaptabilidad y flexibilidad. Al utilizar herramientas y plataformas gratuitas o internas, se evita la dependencia de servicios externos, lo cual es idóneo dado el presupuesto limitado. Esto permite ajustar la estrategia según las necesidades específicas del proyecto y los recursos disponibles.

Otro aspecto importante es la mejora continua y el feedback. El enfoque en la ejecución de pruebas exploratorias y el reporte de incidencias asegura que se identifiquen no sólo los defectos esperados, sino también problemas inesperados, proporcionando un valioso feedback continuo para el equipo de desarrollo.

La preparación para futuras implementaciones es otro factor de justificación. Evaluar la efectividad de las herramientas y técnicas utilizadas en esta estrategia no solo beneficia el proyecto actual, sino que también proporciona una base sólida para la posible apertura de una división de pruebas automatizadas en TSDC. Los resultados y el análisis de esta estrategia servirán como un estudio de caso para futuras implementaciones y decisiones estratégicas en la compañía.

La documentación y trazabilidad están aseguradas mediante un inventario detallado de pruebas manuales y la documentación completa en el repositorio, facilitando el seguimiento, la revisión y la auditoría del proceso de pruebas. Esto garantiza la transparencia y la rendición de cuentas en todas las fases del proyecto.

Finalmente, la estrategia está alineada con los objetivos de negocio de TSDC, asegurando que el software entregado cumpla con los requisitos del cliente y los estándares de calidad esperados. La presentación de los resultados y el análisis al CTO permite una evaluación clara y directa del impacto y los beneficios de la estrategia implementada. En resumen, esta estrategia de pruebas está justificada por su enfoque integral, optimización de recursos, reducción de riesgos, flexibilidad, mejora continua, preparación para el futuro, documentación detallada y alineación con los objetivos de negocio. Esto asegura no solo la calidad del producto final, sino también el desarrollo y fortalecimiento de las capacidades internas de pruebas en TSDC.

Link del video: https://youtu.be/tL-If2-Wnl4