Contenido

[Estrategia de pruebas 1](#_Toc67252555)

[1. Aplicación bajo prueba 1](#_Toc67252556)

[1.1 Nombre Aplicación: 1](#_Toc67252557)

[1.2 Versión: 1](#_Toc67252558)

[1.3 Descripción: 1](#_Toc67252559)

[1.4 Funcionalidades: 1](#_Toc67252560)

[1.5 Diagrama de arquitectura: 3](#_Toc67252561)

[1.6 Diagrama de contexto: 4](#_Toc67252562)

[1.7 Modelo de Datos: 5](#_Toc67252563)

[1.8 Modelo de GUI: 5](#_Toc67252564)

[2. Contexto de la estrategia de pruebas 6](#_Toc67252565)

[2.1 Objetivos 6](#_Toc67252566)

[2.2 Presupuesto de pruebas 7](#_Toc67252567)

[2.2.1 Distribución de esfuerzo 7](#_Toc67252568)

[2.2.2 Recursos humanos 10](#_Toc67252569)

[2.2.3 Recursos computacionales 11](#_Toc67252570)

[2.3 Recursos económicos para contratación de servicios 11](#_Toc67252571)

[2.4 TNT (Técnicas, Niveles y Tipos) de pruebas 11](#_Toc67252572)

[Anexos 13](#_Toc67252573)

[Anexo A - Enlace a repositorio Github 13](#_Toc67252574)

[Anexo B - Enlace al video 13](#_Toc67252575)

# Estrategia de pruebas

# Aplicación bajo prueba

## Nombre Aplicación:

Ghost

## Versión:

3.4

## Descripción:

Ghost es una plataforma de blogs de código abierto y gratuita escrita en JavaScript y distribuida bajo la licencia MIT, diseñada para simplificar el proceso de publicación en línea para bloggers individuales, así como publicaciones en línea.

## Funcionalidades:

|  |  |  |
| --- | --- | --- |
| **Identificador funcionalidad** | **Funcionalidad** | **Descripción** |
| PA-S01-FN001 | Autenticar usuario administrador | Interfaz de usuario que permite al usuario administrador acceder a la consola de administración de la aplicación. |
| PA-S01-FN002 | Recuperar credenciales de administrador | Mecanismo que permite al usuario administrador recuperar el acceso a la aplicación en caso de haber olvidado las credenciales de autenticación. |
| PA-S01-FN003 | Crear tag | Adición de palabras claves que podrán ser asociadas a los contenidos públicos para que estos sean indexados por los motóres de busqueda públicos de internet. |
| PA-S01-FN004 | Editar tag | Dado un registro existente de Tag, modificar alguno de sus atributos y persistir los cambios. |
| PA-S01-FN005 | Configurar enlaces a cuentas de redes sociales | Conformación de enlaces hacia las cuentas de redes sociales asociadas al sitio web. |
| PA-S01-FN006 | Edigtar post | Permite actualizar la información básica de un post previamente registrado en el sistema. |
| PA-S01-FN007 | Gestionar secciones de navegación en interfaz de usuario sobre el sitio público (tags) | Las secciones de navegación representan la estructura de navegación del sitio web público. Sobre este es posible adicionar y/o suprimir y/o modificar las secciones existentes. |
| PA-S01-FN008 | Listar posts existentes | Desde el sitio administrador, se lista la totalidad de post existentes dentro del sitio, indicando para cada registro información como título, fecha de publicación, autor, etc. |
| PA-S01-FN009 | Actualizar password de administrador | Funcionalidad que permite actualizar el password del usuario administrador del sitio |
| PA-S01-FN010 | Invitar colaborador | Funcionalidad que permite convocar a nuevos colaboradores para que generen contenidos y administren la aplicación. |
| PA-S01-FN011 | Revocar invitación a colaborador | Funcionalidad que permite revocar la invitación previamente emitida a un colaborador. |
| PA-S01-FN012 | Establecer titulo y descripción del sitio | Establecer el título y la desccripción del CMS. |
| PA-S01-FN013 | Cambiar zona horaria | Permite establecer la zona horaria bajo la cual se encuentran los clientes mas usuales de la aplicación. |

## Diagrama de arquitectura:

Diagrama

Descripción generada automáticamente

Ghost está estructurado como una aplicación web moderna y desacoplada con una arquitectura basada en servicios. A continuación, se describen los componentes más relevantes:

* Una API JSON de núcleo robusto
* Una aplicación para la administración de la plataforma
* Una capa de tema de front-end simple y potente
* Una base de datos donde se persiste la información
* Una capa de integración hacia algunos de los servicios cloud proporcionados por los proveedores mas imporates del mercado.

Estos componentes funcionan juntos para que cada sitio de Ghost funcione sin problemas, pero debido a que están desacoplados, hay mucho espacio para la personalización.

## Diagrama de contexto:

Diagrama

Descripción generada automáticamente

Hay cinco roles de usuario de personal diferentes dentro de Ghost:

* **Colaboradores**: pueden iniciar sesión y escribir publicaciones, pero no pueden publicar
* **Autores**: pueden crear y publicar nuevas publicaciones y etiquetas.
* **Editores**: pueden invitar, administrar y editar autores y colaboradores.
* **Administradores**: tienen permisos completos para editar todos los datos y configuraciones
* **Propietario**: un administrador que no se puede eliminar y que tiene acceso a los detalles de facturación
* **Usuario final**: Visualiza los post´s y el contenido publicado en ghost.

## Modelo de Datos:

Diagrama, Esquemático

Descripción generada automáticamente

## Modelo de GUI:

Diagrama

Descripción generada automáticamente

# Contexto de la estrategia de pruebas

## Objetivos

* Abordar el espacio de estados y de entradas de [las funcionalidades seleccionadas](#_Funcionalidades:), aplicando sobre ellas técnicas, niveles y tipos de pruebas, de acuerdo con el presupuesto estipulado y recursos disponibles.
* A través de la ejecución de pruebas, identificar la mayor cantidad de bugs dentro del sistema bajo pruebas.
* Maximizar el número de pruebas a ejecutar dentro del plazo estipulado para la ejecución del presente plan.
* Asegurar el incremento sostenible de la calidad del producto a través de la ejecución de las iteraciones del plan de pruebas.

## Presupuesto de pruebas

### Distribución de esfuerzo

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

La presente estrategia contempla cuatro iteraciones de dos semanas cada una, se busca estructurar una estrategia TNT que siga la distribución de esfuerzos bajo el modelo Pirámide. A continuación, se detallan las actividades a ejecutar en cada sprint.

#### Sprint 1

En esta etapa se realizarán pruebas de exploración, 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 prueba que serán objeto de trabajo a lo largo de las iteraciones contempladas.

Posteriormente, el equipo de pruebas deberá enfocarse en la implementación de pruebas unitarias en cada una de las funcionalidades seleccionadas en el plan de pruebas.

#### Sprint 2

A este punto, contamos con un set de pruebas unitarias robusto y un listado de casos y escenarios de prueba. Con el objetivo de consolidar las pruebas unitarias existentes a este punto, desarrollaremos un monkey que ejecute pruebas aleatorias de exploración y permita al equipo de pruebas identificar aquellos casos que no fueron contemplados en las pruebas inicialmente implementadas. Las pruebas serán ejecutadas en horario no hábil por periodos de hasta 12 horas, al inicio del día hábil inmediatamente siguiente, el equipo de pruebas tomará los resultados arrojados por el monkey, los estudiará en detalle y generará un reporte de hallazgos.

A partir del reporte generado, el equipo de pruebas procederá con el ajuste correspondiente en las pruebas unitarias. Finalmente, se construirá un reporte de gestión para la gerencia de la organización, en el cual se reporten las actividades ejecutadas a la fecha, así como el número de hallazgos encontrados y solventados.

#### Sprint 3

Interfaz de usuario gráfica

Descripción generada automáticamente

En este punto del plan de pruebas, se definen dos áreas de trabajo: pruebas aleatorias automatizadas y pruebas automatizadas de regresión, cada integrante del equipo de pruebas se apersonará de uno de los dos frentes.

Para materializar la estrategia de pruebas aleatorias, el equipo de pruebas en conjunto tomará como insumo los hallazgos encontrados y resueltos en el sprint 2, a continuación, reevaluará el conjunto de casos y escenarios de prueba planteados buscando incluir los escenarios que no hayan sido contemplados inicialmente. A partir de este insumo y del conocimiento adquirido en la ejecución de las pruebas exploratorias, el primer ingeniero de pruebas procederá con la implementación de pruebas automáticas e2e utilizando como herramienta el API Cypress.

Las pruebas en Cypress deben considerar el 100% de los casos y escenarios incluidos en el plan de pruebas, una vez hayan sido implementados, se procede con la ejecución de estos, tan pronto finalice este proceso, el ingeniero de pruebas de remitirse a la lectura e interpretación de los resultados obtenidos enfocándose en la identificación de bugs que pudiera arrojar el sistema, este ejercicio debe generar como salida, un reporte donde se puedan consolidar los hallazgos encontrados; a partir del ejercicio anterior, el ingeniero procederá a ajustar las pruebas unitarias correspondientes.

Con el objetivo de validar la versión de Cypress inicialmente entrega al equipo (versión X), la versión generada a partir de la solución a los bugs reportados en la primera iteración (versión X +1) y de contar con una base para asegurará las futuras versiones del producto que pudieran generar, el segundo ingeniero procederá con la construcción de pruebas automáticas de regresión utilizando las API’s Cypress y Resembler.JS. Se deben automatizar el 100% de los escenarios de prueba definidos al principio del presente sprint. Una vez se finalice la implementación de estas pruebas, se procederá con la ejecución de los mismos, el análisis de los resultados obtenidos, la construcción del reporte de hallazgos y el ajuste respectivo en las pruebas unitarias de acuerdo a los hallazgos encontrados

#### Sprint 4

A partir de los hallazgos encontrados y solucionados en los sprint 1, 2 y 3, se espera que al inicio del sprint 4 se cuente con un producto estable, con un alto nivel de calidad y un bajo porcentaje de bugs. En consecuencia, en el presente sprint los esfuerzos se orientarán hacia la ejecución de pruebas de sistema y de aceptación.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Para el presente sprint, el trabajo será distribuido equitativamente entre los integrantes del equipo de pruebas. La primera semana se espera recolectar un conjunto de datos a priori, los cuales permitirán ejecutar las pruebas de sistema. Posteriormente, el equipo desarrollará casos de pruebas automatizadas en los cuales se utilizará la data recolectada; a continuación, se procederá con la ejecución de las pruebas desarrolladas e inmediatamente después, se ejecutará la suit de pruebas de regresión implementadas en el sprint anterior. De ambos procesos se evaluarán los resultados obtenidos y se genera el reporte del ciclo de pruebas, en caso de encontrar errores adicionales, se procederá con el ajuste respectivo a las pruebas unitarias. Finalmente, se ejecutará por segunda ocasión la suite de pruebas de sistema y de regresión, esperando que los errores encontrados hayan sido superados.

Finalmente, en la última semana se utilizará una sábana de datos semi automatizados utilizando la misma suite de pruebas de sistema desarrollada en la semana inmediatamente anterior, en este caso se dará un enfoque de pruebas de aceptación teniendo en cuenta que los datos recolectados se asimilarán a los datos reales que procesará la aplicación. Luego se procederá a ejecutar las pruebas mencionadas y - de nuevo – las pruebas de regresión. Se generará un reporte de hallazgos y en caso de encontrar errores, se procederá a ajustar las pruebas unitarias, y se iterará una vez más sobre la ejecución de las pruebas señaladas en este sprint.

Por último, se generará el reporte de gestión final, señalando la totalidad de errores encontrados y solventados, las funcionalidades abarcadas, las técnicas, niveles y tipos utilizados en la estrategia de pruebas y el estado del producto final.

### Recursos humanos

Se requieren dos ingenieros de pruebas con perfil semi senior con al menos dos años de experiencia en tareas de desarrollo de sw y dos meses en automatización de pruebas. Debe contar con conocimientos específicos en HTML, CSS, Javascript, JQuery, Junit, Node.JS, git, git hub, postman. Deseable experiencia y conocimiento en Cypress, RIP y Resemble JS.

### Recursos computacionales

Para la presente estrategia se requieren dos máquinas con las siguientes características:

|  |  |
| --- | --- |
| **Sistema Operativo:** | Linux Ubuntu / Windows 10 |
| **Procesador** | Core i7 |
| **Memoria RAM** | 16 Gigas |
| **Disco duro** | 50 Gigas disponibles |
| **Software instalado** | * Ghost 3.4 * SQL Lite * Git bash * Cypress * Node.Js * Npm |

Las presentes maquinas son requeridas en horario laboral: de 7:00 am a 6:00 pm y en jornada extendida cuando sea requerido.

## Recursos económicos para contratación de servicios

|  |  |  |  |
| --- | --- | --- | --- |
| **Recurso** | **Cantidad** | **Valor unitario estimado/mes** | **Valor total estimado/mes** |
| Ingeniero de pruebas | 2 | USD $1500 | USD $3000 |
| Computador personal | 2 | No Aplica | $0 |
| **Total:** |  | **USD $1500** | **USD $3000** |

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

Para asegurar la calidad del producto, se propone el uso de las siguientes técnicas:

* Para asegurar las funcionalidades críticas de la aplicación, se propone hacer uso de API’s de automatización a nivel de pruebas unitarias, para ello se utilizará el framework Mocha.
* Para complementar el universo de entradas y estados abordados inicialmente en los casos de prueba y escenarios de prueba definidos, se propone implementar pruebas funcionales aleatorias de integración utilizando la técnica de pruebas aleatorias con Monkey Test; para hará uso de la herramienta Cypres.
* En la implementación de pruebas e2e, se hará uso de pruebas funcionales a nivel de integración a través del API de automatización Cypress.
* En la implementación de pruebas de regresión, se hará uso de pruebas funcionales a nivel de sistema a través del API de automatización Cypress y la librería Resembles JS.
* Finalmente, para ejecutar las pruebas de aceptación de tipo funcional con en estrategia de datos a priori y datos pseudo automáticos, se propone el uso del API Cypress.

# Anexos

## Anexo A - Enlace a repositorio Github

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

<https://github.com/fabiandavid01/tallersemana8>

## Anexo B - Enlace al video

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

<https://uniandes-my.sharepoint.com/:v:/g/personal/a_luquel_uniandes_edu_co/EZ46a6yXANxMu5pEU8SwUFkBoTz_7gjg5ufikOq2p9Kn5g?e=JeaOfk>