Estrategia de Pruebas

1. Aplicación Bajo Pruebas
   1. **Nombre Aplicación:** Ghost
   2. **Versión:** 3.45.0
   3. **Descripción**:

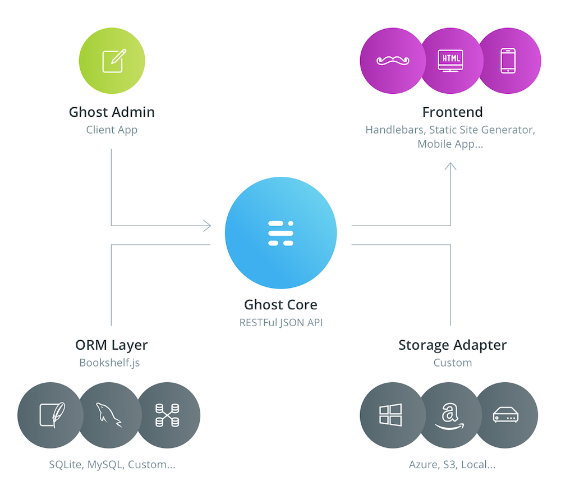
Ghost es una aplicación Desktop de código abierto la cual cumple la función de blog/CMS para cualquier persona o empresa que lo necesite. Fue lanzada en Abril del 2016, con 3 versiones hasta el momento. Esta aplicación está escrita en Javascript y muestra una arquitectura de software similar a otros CMS. Además de dar la posibilidad de publicar posts, la aplicación permite gestionar los usuarios que pueden acceder al CMS y la personalización de los estilos publicados en la página.

Teniendo en cuenta la gran cantidad de características que nos presenta la aplicación, podría decirse que hace parte del top 5 de CMS presentes en el mercado. En efecto, en su página web se presentan casos de comparación del producto frente a alternativas como Medium, WordPress, Substack, entre otros. Ghost resalta por ser un proyecto moderno, rápido y liviano, construido en tecnologías de punta (a diferencia de por ejemplo WordPress) y continua siendo un proyecto de código abierto (lo cual es una ventaja frente a Medium y Substack).

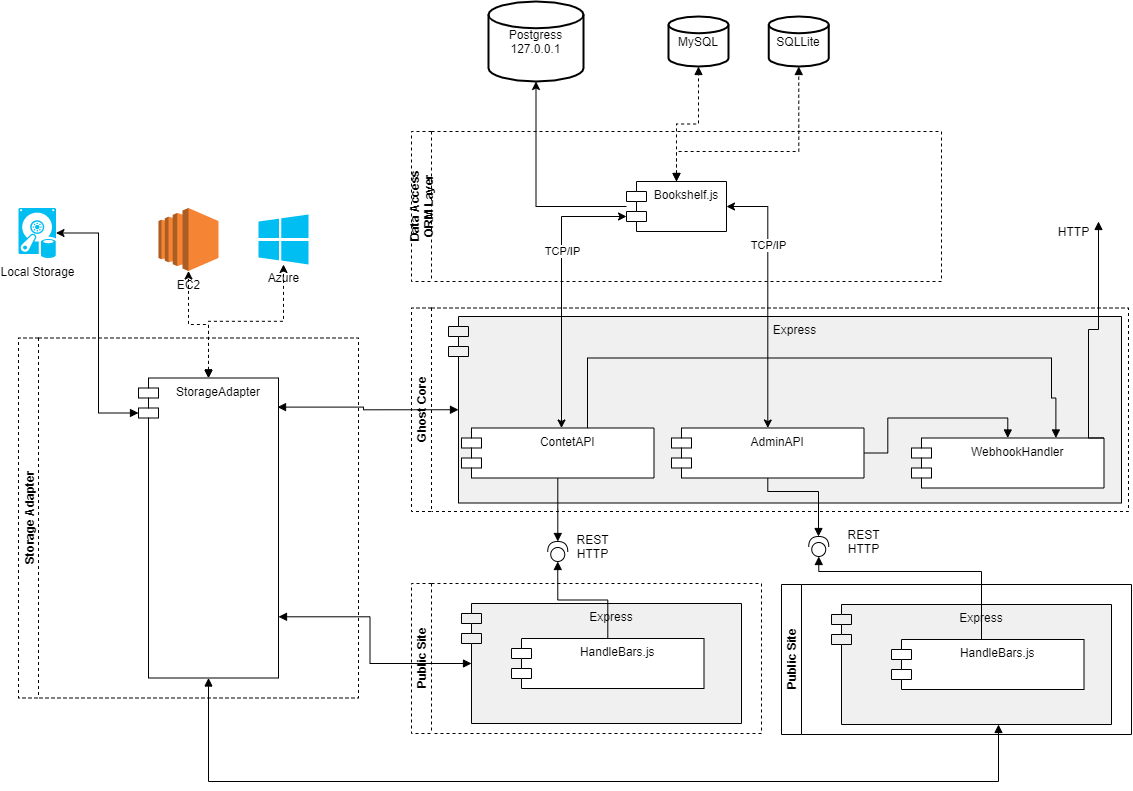
Como mencionamos anteriormente, una de las principales características de Ghost es la de estar escrito en JavaScript y no utilizar otros lenguajes del lado del servidor tradicionalmente asociados a los sistemas de gestión de contenidos como PHP o Java. Sin embargo, se ha desarrollado mediante Node.js, el cual es un entorno de programación diseñado para crear productos del lado del servidor con Javascript basado en el motor de V8 de Google Chrome.

(Tomado de [www.rubenalcaraz.es....](http://www.rubenalcaraz.es/pinakes/software-libre/ghost-una-plataforma-de-codigo-abierto-para-blogueros/#:~:text=Ghost%20es%20un%20sistema%20de,crowdfunding%20en%20la%20plataforma%20Kickstarter.) )

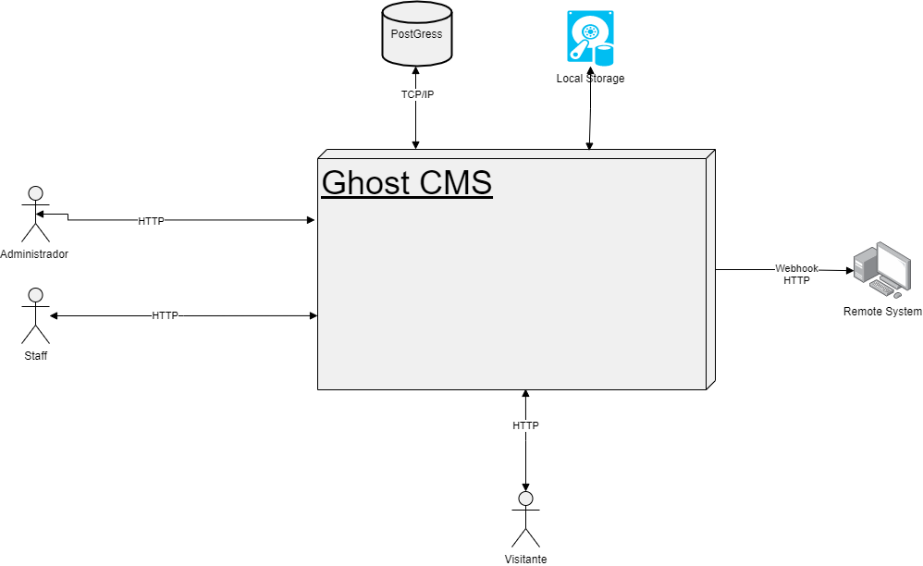
* 1. **Funcionalidades Core**:
* Posts
  + Listar posts publicados
  + Listar posts en borrador
  + Listar posts agendados
  + Crear post
  + Editar un post
  + Asignar una imagen a un post
  + Asignar una URL a un post
  + Gestionar hora de publicación de un post
  + Asignar tags a un post
  + Asignar un resumen a un post
  + Gestionar autores de un post
  + Gestionar metadata de un post para motores de búsqueda
  + Gestionar metadata de un post para Twitter
  + Gestionar metadata de un post para Facebook
  + Gestion de scripts y estilos para un post
  + Publicar un post
  + Despublicar un post
* Pages
  + Listar paginas
  + Crear paginas
  + Editar paginas
  + Asignar una imagen a una pagina
  + Asignar una url a una pagina
  + Establecer la fecha y hora de publicación de un pagina
  + Establecer un tag para una pagina
  + Establecer un resumen para una pagina
  + Establecer los autores de una pagina
  + Gestionar metadata de una página para motores de búsqueda
  + Gestionar metadata de una página para Twitter
  + Gestionar metadata de una página para Facebook
  + Gestion de scripts y estilos para una página
* Tags
  + Listar tags
  + Crear tag
  + Editar tag
  + Gestionar metadata de un tag para motores de búsqueda
  + Gestionar metadata de un tag para Twitter
  + Gestionar metadata de un tag para Facebook
  + Gestion de scripts y estilos para un tag
* Staff
  + Listar usuarios del sistema
  + Crear usuario
  + Editar usuario
* Settings
  + General
    - Establecer título y descripción del sitio
    - Establecer la zona horaria de la plataforma
    - Establecer el lenguaje de publicación
    - Asociar un icono al sitio
    - Establecer el logotipo del sitio
    - Establecer una cubierta para las publicaciones
    - Gestionar metadata para motores de búsqueda
    - Gestionar metadata para Twitter
    - Gestionar metadata para Facebook
  + Diseño
    - Gestionar el menú principal de navegación
    - Gestionar el menú secundario de navegación
    - Seleccionar el tema que usara el sitio
    - Subir temas al sitio
  + Gestion de scripts y estilos
  + Gestion de integraciones con aplicaciones de terceros.
  + Labs
    - Gestion de membresías
    - Importar contenido
    - Exportar contenido
    - Borrar todo el contenido
    - Establecer modo opaco.
  1. **Diagrama de Arquitectura:**



Tomado de <https://ghost.org/docs/architecture/>



* 1. **Diagrama de Contexto:**

****

* 1. **Modelo de Datos:**

<https://raw.githubusercontent.com/raulgsalguero82/PruebasAutomaticas-Ghost/master/ModeloDeDominio.png>

* 1. **Modelo de GUI:**

<https://github.com/raulgsalguero82/PruebasAutomaticas-Ghost/blob/master/ModeloDeGUI.png?raw=true>

1. **Contexto de la estrategia de pruebas**
   1. **Objetivos:**

El objetivo general que se busca es lograr generar un esquema de pruebas que permita la evolución y mejora del cms de forma segura y sostenible.

Con este fin hemos definido cuatro fases con objetivos y actividades específicas:

1. Semana 1, 2 y 3: En estas semanas buscamos adquirir un conocimiento y visión generales del sistema mediante el uso de pruebas manuales y de reconocimiento, con el fin de poder entender mejor su funcionamiento y sentar las bases de los procesos que serán susceptibles de ser automatizados. Así mismo se espera construir el inventario de pruebas manuelas que serán base para la documentación del proyecto. En esta etapa tiene sentido realizar las pruebas de reconocimiento debido a que estas nos permiten encontrar desde un primer momento las excepciones y casos raros que presente la aplicación. Adicionalmente, estas pruebas de tipo Ripper nos permiten avanzar en el objetivo de documentar plenamente la estrategia de pruebas.
2. Semanas 4,5 y 6: Durante estas tres semanas buscamos elaborar una batería de pruebas automatizadas de aquellas funcionalidades que hemos detectado que son susceptibles de serlo. Más precisamente, se harán pruebas sobre aquellas funcionalidades que son consideradas esenciales para el funcionamiento mínimo de la aplicación.

Para esta semana, además de usar como insumo las pruebas exploratorias de las 3 primeras semanas, usaremos el conocimiento adquirido del modelo de datos y contexto de la aplicación para de esta forma usarlo en la generación de data de prueba que nos permita explorar mejor los escenarios a estudia. De esta manera, se pueden testear múltiples escenarios de prueba que incluyan la interacción por parte del usuario en los formularios, lo cual nos ayudará a entender integralmente como se comporta la aplicación cuando procesa datos no esperados en los formularios.

1. Semana 7: Durante esta semana nos proponemos el poner en marcha pruebas de regresión usando la data arrojada por las pruebas automatizadas en el ciclo anterior (semanas 4,5 y 6) con el fin de sentar las bases para que este proceso pueda repetirse de forma automatizada en el futuro. Así mismo, este proceso nos permite hacer una comparación entre dos versiones distintas de la aplicación, anclando al tiempo los escenarios de prueba. Dentro de otras cosas, esto permite llevar una trazabilidad sobre los esfuerzos realizados por el equipo de desarrollo para corregir los errores encontrados y encontrar diferencias en diseño y usabilidad entre diferentes versiones de la aplicación.
2. Semana 8: Durante esta semana detallaremos los pasos y requerimientos para que las automatizaciones realizadas durante las fases 2 y 3 (semanas 4 a 7) se puedan repetir sistemáticamente en el futuro, así como la elaboración de análisis y retrospectivas que permitan mejorar el proceso de pruebas
   1. **Presupuesto de pruebas:**

Se parte del presupuesto que para esta estrategia se cuenta con dos testers senior, durante 8 semanas, en una dedicación de 8 horas por persona semanalmente.

* + 1. **Distribución de Esfuerzo:**

Para cumplir con los objetivos propuestos, proponemos la siguiente distribución de tareas y entregables:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Semana | Tarea | Responsable | Entregable | Duración (Horas) |
| 1 | Instalación y configuración de ghost | Raul | \* URL de ejecución y datos de acceso | 1 |
| 1 | Exploración manual de la parte publica | Raul | \* Issues observables | 2 |
| 1 | Exploración manual de la parte privada | Miguel | \* Issues observables | 3 |
| 1 | Elaboración de diagrama de UI | Miguel y Raul | \* Diagrama UI | 8 |
| 1 | Elaboración de reporte de avances | Miguel y Raul | \* Video reporte semana1 | 2 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| 2 | Configuración y puesta en marcha de un ripper sobre la parte privada | Raul | \* Reporte de Issues encontrados | 4 |
| 2 | Configuración y puesta en marcha de un monkey sobre la parte publica | Miguel | \* Reporte de Issues encontrados | 4 |
| 2 | Elaboración del modelo de datos de la aplicación | Miguel y Raul | \* Diagrama de Datos | 4 |
| 2 | Elaboración de reporte de avances | Miguel y Raul | \* Videoreporte semana2 | 4 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| 3 | Elaboración de diagrama de arquitectura | Miguel y Raul | \* Diagrama de arquitectura | 6 |
| 3 | Elaboración de diagrama de contexto | Miguel y Raul | \* Diagrama de contexto | 6 |
| 3 | Elaboración de reporte de avances | Miguel y Raul | \* Videoreporte semana3 | 4 |
|  |  |  |  |  |
| 4 | Automatización de pruebas (Test-001 a Test-005) | Raul | \* Reporte de Issues encontrados | 7 |
| 4 | Automatización de pruebas (Test-006 a Test-010) | Miguel | \* Reporte de Issues encontrados | 7 |
| 4 | Elaboración de reporte de avances | Miguel y Raul | \* Videoreporte semana4 | 2 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| 5 | Automatización de pruebas (Test-0011 a Test-015) | Raul | \* Reporte de Issues encontrados | 7 |
| 5 | Automatización de pruebas (Test-016 a Test-020) | Miguel | \* Reporte de Issues encontrados | 7 |
| 5 | Elaboración de reporte de avances | Miguel y Raul | \* Videoreporte semana5 | 2 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| 5 | Automatización de pruebas (Test-0011 a Test-015) | Raul | \* Reporte de Issues encontrados | 7 |
| 5 | Automatización de pruebas (Test-016 a Test-020) | Miguel | \* Reporte de Issues encontrados | 7 |
| 5 | Elaboración de reporte de avances | Miguel y Raul | \* Videoreporte semana5 | 2 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| 6 | Automatización de pruebas (Test-0021 a Test-025) | Raul | \* Reporte de Issues encontrados | 7 |
| 6 | Automatización de pruebas (Test-026 a Test-030) | Miguel | \* Reporte de Issues encontrados | 7 |
| 6 | Elaboración de reporte de avances | Miguel y Raul | \* Videoreporte semana6 | 2 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| 7 | Instalación y configuración de instancia ghost a probar (Versión 3.46.0) | Miguel | \* URL y datos de acceso | 1 |
| 7 | Limpieza y preparación de la instancia base de ghost (Version 3.45.0) | Raul | \* URL y datos de acceso | 1 |
| 7 | Ejecución de batería de pruebas automatizadas (Test-001 a Test-030) sobre la versión base | Raul | \* Capturas paso a paso de cada uno de los escenarios probados | 5 |
| 7 | Ejecución de batería de pruebas automatizadas (Test-001 a Test-030) sobre la versión a comparar | Miguel | \* Capturas paso a paso de cada uno de los escenarios probados | 5 |
| 7 | Elaboración del reporte de diferencia encontradas entre la versión base y la nueva versión | Miguel y Raul | \* Reporte de diferencias encontradas | 4 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| 8 | Compilación de herramientas necesarias para la ejecución de las pruebas | Miguel y Raul | \* Gits con instrucciones para la ejecución | 10 |
| 8 | Elaboración del reporte final del proceso de pruebas | Miguel y Raul | \* Reporte final semana 8 | 6 |

* + 1. **Recursos Humanos:**

Nuestros dos ingenieros automatizadores contaran con los siguientes skills:

* Conocimiento de JS
* Conocimientos de DOM
* Conocimientos avanzados en HTML
* Conocimientos avanzados en CSS
* Manejo avanzado de Cypress.io
* Manejo avanzado de ResemblerJS
* Manejo avanzado de NodeJS
* Configuración y despliegue de instancias Linux
  + 1. **Recursos Computacionales:**

Para esto, se emplearán dos instancias gratuitas EC2 Linux las cuales ofrecen 750h de procesamiento, una desde la cuenta de cada ingeniero automatizador

Para más datos acerca de estas instancias por favor dirigirse al link de AWS

https://aws.amazon.com/es/free/?all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc

Así, en cada una de estas instancias se propone la siguiente configuración:

1. Instancia Web:
   1. Ghost 3.45.0 (Puerto 2368)
   2. Ghost 3.46.0 (Puerto 2369)
2. Instancia Pruebas:
   1. CypressIO
   2. NodeJS
      1. **Recursos Económicos para la contratación de servicios:**

En este apartado nos planteamos hacerlo todo con recursos propios o empleando software de licencia libre o herramientas de pruebas gratuitas.

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

Tal y como comentábamos en la sección de objetivos, cada una de estas secciones tiene una definición de TNT diferente dado que los objetivos individuales son diferentes, a saber:

**Fase 1: Exploración y conocimiento del sistema (Semanas 1 a 3)**

**Técnicas:**

* Pruebas manuales: Con el fin de recabar información acerca del comportamiento general del sistema y poder generar scripts que repliquen dichos comportamientos, se procederá a realizar una tanda de pruebas manuales por parte de los ingenieros de automatización.
* Uso de herramientas de reconocimiento automatizadas: Con el fin de poder tener una visión más amplia del mapa de navegación del sitio y de los posibles errores de ejecución “ocultos”, plantemos el uso de reapers y monkeys que mejoren nuestra visión y cobertura de la aplicación.

**Niveles:**

* Nivel de aceptación: Se abordará este nivel desde la perspectiva de las expectativas que pueda tener de la aplicación un usuario externo al equipo de desarrollo

**Tipos:**

* **No Funcionales:** Planteamos el monitorizar el uso de recursos del sistema ante las pruebas realizadas.
* **Funcionales:** Con el fin de definir scripts automatizables, cuando esto sea posible, planteamos realizar pruebas que verifiquen las funcionalidades esperadas a partir de la funcionalidad que se descubra del sistema.

Con este fin, se plantea ejecutar pruebas positivas que corroboren que dichas funcionalidades operan como se esperaría y pruebas negativas que pongan a prueba las validaciones y barreras que debería ofrecer el sistema ante posibles fallos humanos.

**Fase 2: Automatización de pruebas (Semanas 4 a 6)**

**Técnicas:**

* Proponemos la implementación de pruebas e2e basadas en scripts ejecutables desde cypress que nos permitan validar que las funcionalidades esperadas se cumplan y actúen como se espera.

Así mismo, con el fin de complementar estas pruebas e2e y dado el conocimiento del modelo de datos y arquitectura ganados en la fase anterior, usaremos las siguientes técnicas de generación de datos:

\* Pools de datos a priori: Con el fin de validar secciones con restricciones conocidas a partir de la exploración manual, usaremos pools de datos con escenarios limites conocidos o con información requerida para acceder a otros escenarios (Ej.: usuarios y passwords válidos, fechas conocidas, configuraciones de entradas invalidas, etc)

\* Pools de datos pseudo aleatorios: Con el fin de poder dar mayor cobertura a nuestros escenarios y lograr validación de nuestras pruebas mediante la ejecución sistemática de los mismos, usaremos datos con los tipos esperados (o no esperados) en cuanto a su formato.

\* Datos aleatorios: Finalmente, y con el fin de descubrir posibles escenarios limites, para aquellos campos y tests que lo ameriten, plantearemos pruebas con datos totalmente aleatorios provistos por librerías diseñadas para ello.

**Niveles:**

* Nivel de aceptación: Se abordará este nivel desde la perspectiva de las expectativas que pueda tener de la aplicación un usuario externo al equipo de desarrollo
* Nivel de Sistema: Se abordará este nivel desde la perspectiva de un usuario con conocimiento avanzado de la estructura del sistema y su arquitectura.

Así, buscaremos, por ejemplo, que una posible inyección de sql, fuerza bruta etc, no afecten la ejecución del sistema y su integridad.

**Tipos:**

* **No Funcionales:** Planteamos el monitorizar el uso de recursos del sistema ante las pruebas realizadas.
* **Funcionales:** Con el fin de definir scripts automatizables, cuando esto sea posible, planteamos realizar pruebas que verifiquen las funcionalidades esperadas a partir de la funcionalidad que se descubra del sistema.

Con este fin, se plantea ejecutar pruebas positivas que corroboren que dichas funcionalidades operan como se esperaría y pruebas negativas que pongan a prueba las validaciones y barreras que debería ofrecer el sistema ante posibles fallos humanos.

**Fase 3: Validación de cambios entre versiones (Semanas 4 a 6)**

**Técnicas:**

* Haciendo uso de los artefactos visuales producidos por la batería de pruebas ejecutada por los scripts de la fase 2, proponemos hacer uso de pruebas de regresión visual (VRT) con el fin de detectar posibles cambios entre dos escenarios probados entre diferentes versiones o diferentes despliegues de esta versión

**Niveles:**

* Nivel de Sistema: Dado que estas pruebas validan que una funcionalidad se ejecute paso a paso y cubren toda la ejecución de un escenario, analizaremos estas pruebas a nivel sistema pues nos aportan información acerca del funcionamiento de este y su respuesta a una batería de pruebas.

**Tipos:**

* **No Funcionales:** Planteamos el monitorizar el uso de recursos del sistema ante las pruebas realizadas.
* **Funcionales:** Con el fin de definir scripts automatizables, cuando esto sea posible, planteamos realizar pruebas que verifiquen las funcionalidades esperadas a partir de la funcionalidad que se descubra del sistema.

Con este fin, se plantea ejecutar pruebas positivas que corroboren que dichas funcionalidades operan como se esperaría y pruebas negativas que pongan a prueba las validaciones y barreras que debería ofrecer el sistema ante posibles fallos humanos.