Estrategia de Pruebas

1. Aplicación Bajo Pruebas

1.1. **Nombre Aplicación:** Ghost
1.2. **Versión:** 5.18.0
1.3. **Descripción**:

Ghost es una plataforma de publicación profesional de código abierto basada en una pila de tecnología moderna de Node.js, diseñada para equipos que necesitan potencia, flexibilidad y rendimiento.

Ghost es un CMS (por sus siglas en ingles Content Management System) en la nube, el cual te permite crear, organizar, publicar y eliminar contenidos de tu sitio web. Las CMS proveen 3 funciones principales:

- Creación de webs.
- Gestión y mantenimiento del sitio web.
- Administración de páginas web y del propio CMS.

Otras plataformas (en comparación con Ghost) abiertas son generalmente antiguas, lentas e infladas, mientras que otras plataformas cerradas no le otorgan absolutamente ningún control o propiedad de su contenido. Ghost ofrece lo mejor de ambos mundos y más.

1.4. Enlace al video de Explicación:

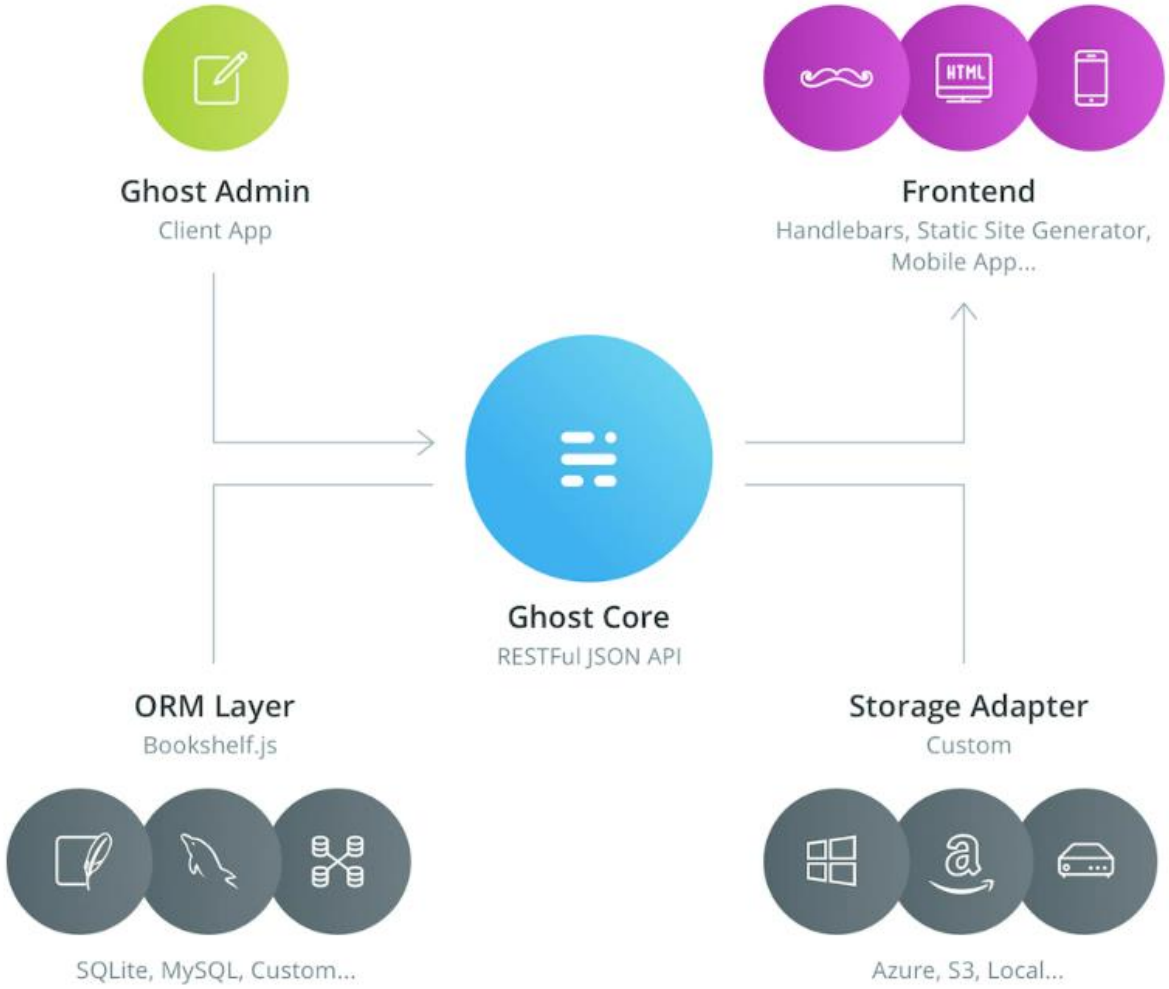
https://youtu.be/w1rRZ4wlZTQ

1.5. Funcionalidades Core:

Ghost provee una serie de funcionalidades entre las cuales encontramos:

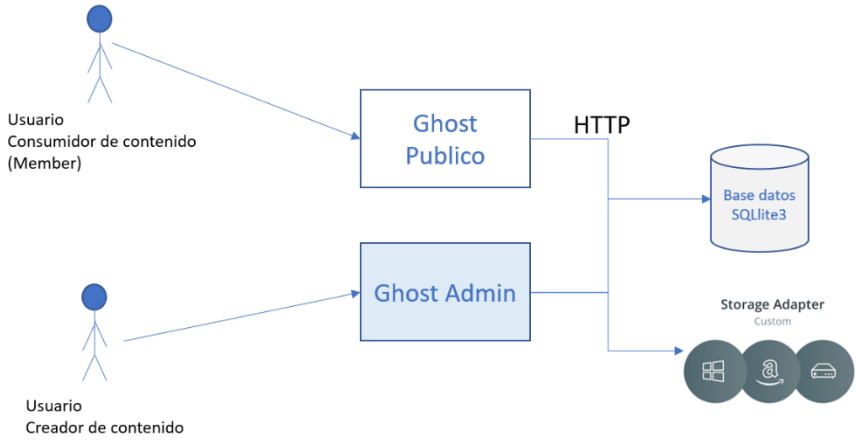
- Editar, crear y publicar un post.
- Borrar, despublicar y programar un post.
- Editar, crear, y publicar una página.
- Borrar, y despublicar una página.
- Editar, crear, y guardar un tag.
- Borrar, y despublicar un tag.
- Administrar el blog con distintos usuarios y roles específicos.
- Publicar en una fecha específica.

1.6. Diagrama de Arquitectura:



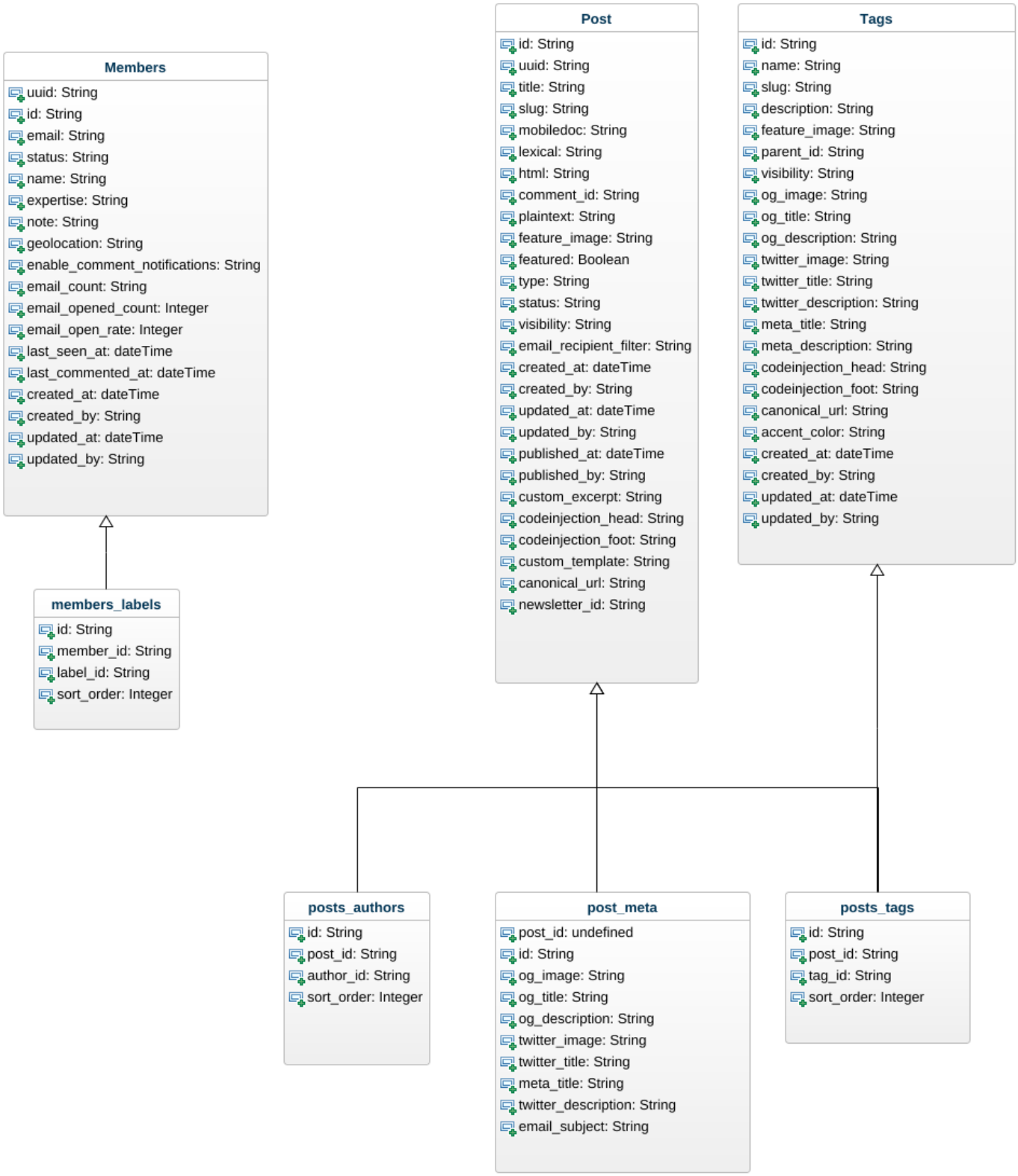
Si la imagen no se aprecia del todo una copia se encuentra dentro del zip de entrega o en el siguiente enlace: https://1drv.ms/u/s!As_QMF2T0qpGgbpEXH1l_ksDyV2GcQ?e=lIg8Vf

1.7. Diagrama de Contexto:



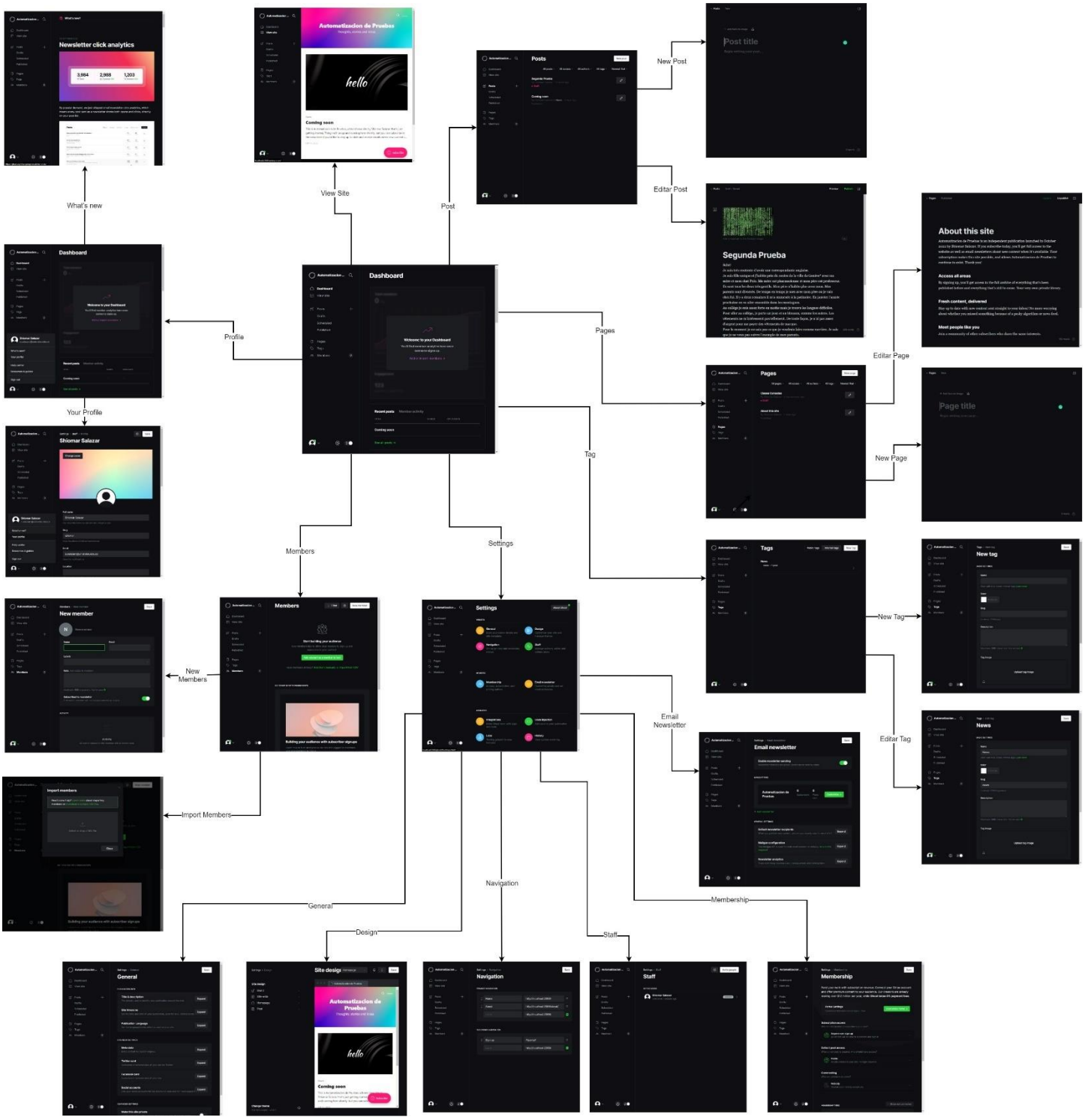
Si la imagen no se aprecia del todo una copia se encuentra dentro del zip de entrega o en el siguiente enlace: https://1drv.ms/u/s!As_QMF2T0qpGgbpCd1Ihx6P5wSTafg?e=fWpYof

1.8. Modelo de Datos:



Si la imagen no se aprecia del todo una copia se encuentra dentro del zip de entrega o en el siguiente enlace: https://1drv.ms/u/s!As_QMF2T0qpGgbpBYUsPm1EPM9ojyA?e=ZucZ2N

1.9. Modelo de GUI:



Si la imagen no se aprecia del todo una copia se encuentra dentro del zip de entrega o en el siguiente enlace: https://1drv.ms/u/s!As_QMF2T0qpGgbpAaZqL_GglUdPoVw?e=ylNb41

2. Contexto de la estrategia de pruebas

2.1. Objetivos:

El objetivo de esta estrategia es la detección de la mayor cantidad de errores posibles en el periodo de la iteración, es por esto por lo que se enfocara más que nada en la parte funcional de la ABP, pero como el recurso local (Ingeniero Automatizador) es de poca experiencia, se espera que este haga y documente pruebas manuales exploratorias que realice durante parte de la iteración, como parte de su proceso de entrada al proyecto.

Segundo objetivo es la entrega de pruebas automatizadas (tanto la proporcionadas por el servicio externo contratado como por el equipo interno), en donde se entreguen al menos 60 pruebas automatizadas (50 por parte del proveedor externo del tipo Caja Negra de GUI y 10 pruebas de caja blanca unitarias por parte del equipo interno.

Tomando en cuenta que en la actualidad no se cuenta ni con inventario de pruebas realizadas (ni manuales ni automatizadas) ni reportes de defectos, se espera con esta estrategia aumentar rápidamente el número de pruebas en el catálogo y utilizando el menor número de recursos disponibles.

2.2. Duración de la iteración de pruebas:

Estas pruebas serán ejecutadas en un periodo de 2 semanas, esto con la finalidad de alinearse con los estándares y buenas prácticas de los equipos agiles. También nos permitirá evaluar la estrategia después de un periodo relativamente corto y determinar los cambios necesarios.

2.3. Presupuesto de pruebas:

Concepto	Costo Unitario	Cantidad	Monto Total
Hora de Trabajo Ingeniero Junior	$USD 5	30 horas	$USD 150
Hora maquina Amazon AWS	$USD 96	1 instancia (400 horas)	$USD 96
Hora de Trabajo Ingeniero Senior	$USD 10	40 horas	$USD 400
Renta de Equipo de Computo	$USD 50	1 equipos	$USD 50
contratación de Servicios Externos	$USD 500	1 sprint	$USD 500
Total			**$USD 1196**

Para 60 pruebas a ejecutar en esta estrategia, tendríamos un valor por prueba de $USD19.93

2.3.1. Recursos Humanos

Para esta estrategia se espera contar con el siguiente personal:

- 1 ingeniero automatizador junior (30 horas/persona), el cual deberá tener los conocimientos teóricos en Pruebas Automatizadas, así como conocimientos relevantes en el uso de AWS (proyectos escolares, tutoriales de al menos 20 horas).
- 4 ingenieros de Software Senior con dedicación limitada (10 horas/persona) que ya poseen experiencia con la ABP y pueden proveer guía y ayuda a la hora de ser necesario al equipo interno y supervisión del equipo externo.

2.3.2. Recursos Computacionales

Se espera para esta estrategia se utilicen los siguientes recursos tecnológicos:

- 400 horas/máquina en Amazon AWS, estas horas maquina serán utilizadas para correr las pruebas automatizadas entregadas tanto por el equipo interno (al Ingeniero automatizador junior), como las entregadas por el servicio externo contratado.
    - 20 horas/máquina de pruebas de reconocimiento Ripper
    - El resto de las horas serán usadas por el ingeniero Junior para sus pruebas automatizadas y manuales y para las pruebas de los servicios externos.
- 1 computador correspondiente al Ingeniero automatizador junior. Estos ordenadores son MacBook Air modelos 2020 en adelante.
- 4 computadores una correspondiente a cada Ingeniero Software Senior. Estos ordenadores son MacBook Air modelos 2020 en adelante.

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

En esta estrategia se considera la siguiente contratación de servicios:

- 500 USD para contratación de servicios de outsourcing de pruebas, en el contrato se estableció la cuota de 10 USD por prueba automatizada funcionando, con lo cual se espera recibir de la empresa 50 pruebas automatizadas sobre la ABP distribuidas entre los diferentes Dispositivos y Navegadores (Android, IOS, Firefox, Chrome y Safari).
    - Las pruebas automatizadas contratadas con el outsourcing sobre cada funcionalidad serán completas, ejemplo: Post, debe incluir desde crear y programar fecha futura con al menos 10 tipos diferentes de contenido (texto, imagen, video, sonido, código HTML, links, botón, lista desplegable, etc.), como también, listar, editar todos los campos utilizados y eliminar un Post, esto implica ejecutar 2 tipos de prueba por funcionalidad, una positiva y una negativa, lo que en total permitirá barrer 25 funcionalidades diferentes.
    - El costo por prueba se asumió basando en comentarios de algunos compañeros perteneciente al sector de Calidad en donde se nos compartió cuanto tiempo se tarda en generar las mismas y cuánto ganan al día.

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

Nivel	Tipo	Técnica	Objetivo
Sistema	Funcional Caja Negra Positivas	Automatizado	Se espera con estas pruebas sirvan para el incremento exponencial del catálogo de pruebas automatizadas del proyecto y a su vez nos sirvan para la detección de errores en el estado actual del entregable.
Unidad	Funcional Caja Blanca Positivas	Automatizada	Se espera con estas pruebas sirvan para el incremento exponencial del catálogo de pruebas automatizadas del proyecto y a su vez nos sirvan para la detección de errores en el estado actual del entregable. Se espera de igual forma ayudar a hacer más robusto el proceso de desarrollo e iniciar un proceso automático de CI que ejecute estas pruebas cada Pull Request
Sistema	Funcional Caja Negra Positivas	Manual	Son las pruebas exploratorias del Ingeniero Junior, la intención es que se familiaricen con la ABP y se ubiquen en las vistas y elementos de esta. Se espera un reporte manual con pasos y resultados.

Sistema	Funcional Caja Negra Positivas	Automatizada	Son las pruebas Ripper que hará el ingeniero Junior, esto con intención que la prueba genere los estados de la aplicación y de un mayor entendimiento de la ABP. Se espera 1 script de pruebas Ripper con diferentes niveles.

2.5. Distribución de Esfuerzo

En esta estimación se espera poder dividir el tiempo del equipo interno en dos bloques de la siguiente forma:

1. Pruebas exploratorias manuales 5 horas/persona, ya que el ingeniero Junior necesita tiempo para conocer la ABP.
    a. Este apartado incluirá un espacio para establecer el ambiente en donde se harán las pruebas manuales que no deberá ser mayor a 2 horas.
    b. El resto del tiempo será para ejecución de las pruebas manuales exploratorias esperando que el Ingeniero lleve un control de los pasos realizados para poder automatizarlos luego.
2. 5 horas de creación de script para pruebas Ripper.
    a. Este apartado incluirá un espacio para establecer el ambiente de prueba nocturna en donde se correrán las de tipo Ripper (1 horas), en donde se incluirán las reglas a ser ejecutadas cada noche en los servidores AWS contratados.
    b. Este apartado incluirá un espacio para analizar los reportes obtenido al final del sprint, cuantos niveles fueron probados, desde que nivel se presentan fallas o alertas de mejora y la generación de estas en el sistema de reporte de errores, se espera que esto no tome más de 2 horas al final del sprint.
    c. El resto del tiempo será utilizado en la escritura de las pruebas de tipo Ripper.
3. Prueba Unitarias Automatizadas 20 horas/persona, ya que así se podrán encontrar aún más errores diseccionando el código de la ABP y se empezara la construcción de catalogo formal de pruebas.
    a. Este apartado incluirá un espacio para establecer el ambiente de Integración Continua en donde se correrán las pruebas unitarias (2 horas), en donde se incluirán las reglas a ser ejecutadas cada Pull Request.
    b. Este apartado incluirá un espacio para analizar los reportes obtenido al final del sprint, este análisis debe incluir Pull Request rechazados, cuales son las pruebas que con más frecuencia fallan y quien recibe errores más frecuentes en sus Pull Request, se espera que esto no tome más de 2 horas al final del sprint.
    c. El resto del tiempo será utilizado en la escritura de las pruebas unitarias.

La intención es que a medida que las iteraciones pasen se puedan agregar más pruebas Unitarias y de Cobertura de código y se vaya empujando al equipo a llegar a la pirámide de automatización Ideal.

En el siguiente diagrama se incluyen las pruebas de los servicios contratados, en este caso al ser un equipo externo y ajeno a nosotros se diagramo en proporción a la cantidad de pruebas entregadas en comparaciones a las pruebas ejecutadas por el equipo interno.

3. Ventajas y Desventajas de la Propuestas

3.1. Ventajas

- Menor Costo a comparación de la estrategia 1.
- Mayor número de pruebas End to End de GUI.
- Genera flexibilidad en el uso de recursos al poder conseguir siempre personas con experiencia en pruebas en el outsourcing, ya que les pagamos por prueba realizada y no tenemos que encargarnos de su curva de aprendizaje.
- Entrega de valor alto al principio.
- Podemos obtener benchmarking de estrategias de pruebas y otros conocimientos de la industria a través de un tercero que nos permee de metodologías nuevas.
- A través de las pruebas Ripper se genera un mayor conocimiento de la ABP al ingeniero junior.

3.2. Desventajas

- A largo plazo la entrega de valor no se evidenciaría, ya que internamente la organización cuenta con el conocimiento del código y del funcionamiento del sistema a través de la madurez de su equipo de pruebas.
- Menor número de pruebas en general.
- Menor número de pruebas unitarias.
- Necesidad de supervisión.
- Poco crecimiento del equipo interno.
- Detección tardía de errores que aumenta la complejidad de reparación.
- Riesgo de Calidad de los entregables por parte de los servicios externos dado el poco conocimiento de la ABP.
- Se presenta el riesgo de que las pruebas no tengan repetibilidad, ya que se pueden estar cambiado de personal que realiza las pruebas manuales.
- Al solo tener pruebas Ripper, perdemos la oportunidad de mayor cobertura de escenarios que son difíciles de detectar para el ingeniero Junior.

4. Análisis de Herramientas Automatizadas para Pruebas de Reconocimiento

En esta estrategia se utilizan solamente Rippers ya que, por limitaciones de recursos humanos, y ya que el Ripper propone ventajas diferentes como generación de vistas y documentación de eventos e interacciones, es la herramienta seleccionada en este caso.

Esta sección contendrá el resultado de utilizar las herramientas automatizadas llamadas Monkey Test (implementada en Cypress) y Ripper Test (implementada con Ripuppet).

4.2 Ripper Test con Ripuppet

Para esta evaluación de la herramienta automatizada se utilizó una configuración de un nivel, esto con la intención de obtener una primera iteración de documentación y poder evaluar a profundidad el sistema.

Para poder iniciar corregimos errores de dependencia, eliminando el archivo package-log.json y luego instalando las librerías con npm install, lo que permite tener un código que corra sin problemas, luego habilitamos el inputValue del archivo config.json con los values de correo y password que se requieren para ingresar a Ghost Admin,

Luego corregimos el método getbuttons para poder obtener los elementos de tipo Button y lanzar eventos a dichos botones, con esto pudimos avanzar en el login y permitirle al Ripper ingresar al Admin.

La prueba es recursiva, es decir identifica el árbol DOM para 3 tipos de elementos: TextInput, Buttons y dropdowns, y para cada tipo de elemento se buscan los siguientes eventos:

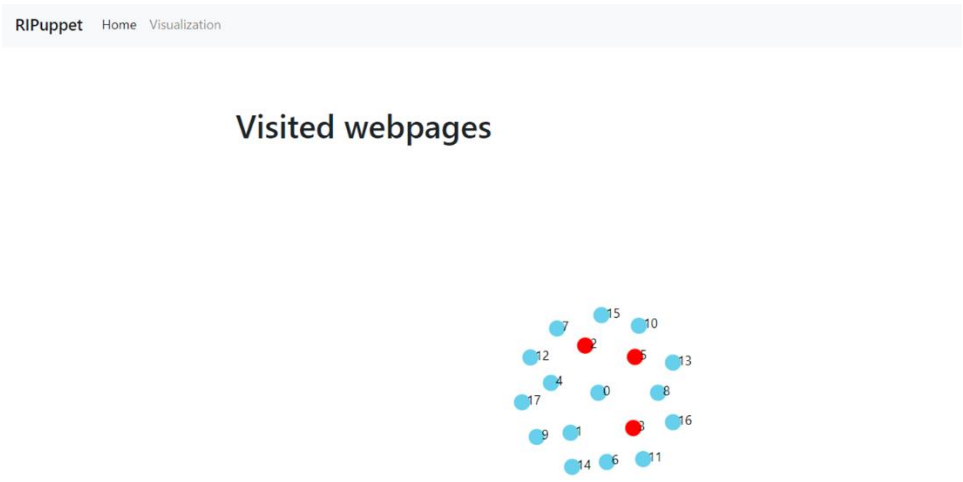
Button: mouse hover y click

TextInput: ingresa el texto que le definamos en el config o genera datos aleatorios generados por un Faker para cada tipo de caja de texto, Ej: tipo email, tipo password, tipo tel, tipo número...

Dropdown: Mouse hover y Click

Al configurar la prueba sobre Ghost para que navegue en el primer nivel, esto nos genera 17 paginas testeadas y dependiendo de los elementos que encontraba los iba probando de acuerdo con lo mencionado anteriormente.

Las pruebas nos arrojan 3 páginas con errores de las 17 escaneadas,



1 http://localhost:2368/ghost/#/site/: { "_type": "info", "_text": "An element was lazyloaded with loading=lazy, but had no dimensions specified. Specifying dimensions improves performance. See https://crbug.com/954323", "_args": [], "_location": { "url": "http://localhost:2368/ghost/#/site/" } }

Issue Jira: https://asolerf.atlassian.net/browse/PAU-12

2 http://localhost:2368/ { "_type": "info", "_text": "An element was lazyloaded with loading=lazy, but had no dimensions specified. Specifying dimensions improves performance. See https://crbug.com/954323", "_args": [], "_location": { "url": "http://localhost:2368/" } }

Issue Jira: https://asolerf.atlassian.net/browse/PAU-14

3 http://localhost:2368/ghost/#/editor/post/ { "_type": "info", "_text": "Autofocus processing was blocked because a document already has a focused element.", "_args": [], "_location": { "url": "http://localhost:2368/ghost/#/editor/post/" } }

Issue Jira: https://asolerf.atlassian.net/browse/PAU-15

4.2.1. Ventajas de la Herramienta

- Se genera documentación de la vistas y eventos posibles lo que facilita un método de exploración.
- Se valida de forma ordenada la ABP.
- Los eventos son un poco más inteligentes en comparación a los Monkey.
- No depende de la modificación del layout de la página para la prueba, dado que el busca los elementos que se definieron y ejecuta los eventos sobre estos para generar los resultados
- Nos permite identificar enlaces rotos según configuremos los elementos de prueba.
- A partir del análisis del árbol DOM podemos determinar qué elementos queremos buscar y probar en la navegabilidad de los niveles de páginas definidos, esto permite generar una estructura de pruebas.

4.2.2. Desventajas de la Herramienta

- La capacidad de la herramienta para detectar errores es baja a comparación de Monkey.
- Posibles resultados falsos si los eventos son posibles, pero no las respuestas posibles.