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:

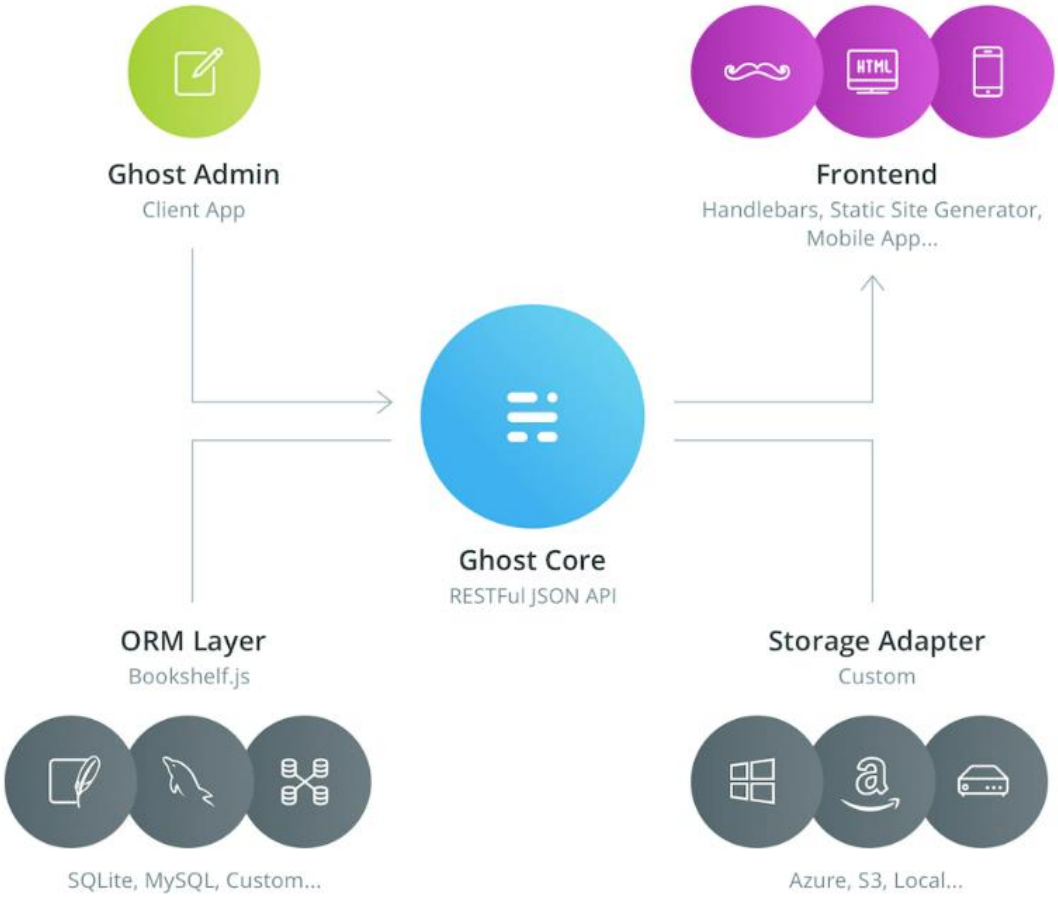
https://youtu.be/KeWcz_PQz4Y

1.5. Funcionalidades Core:

Ghost provee una serie de funcionalidades entre las cuales encontramos:

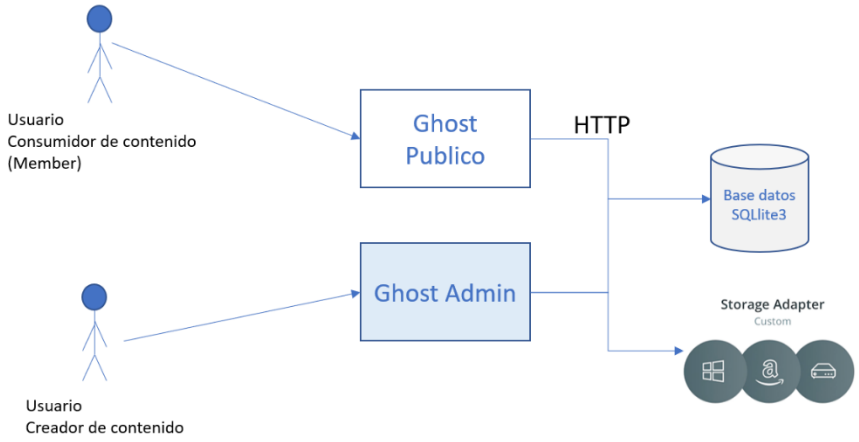
- Editar, crear y publicar un post.
- Borrar, despublicar y programar un post.
- Editar, crear, y publicar una página.
- Borrar, y despublicar una página.
- Editar, crear, y guardar un tag.
- Borrar, y despublicar un tag.
- Administrar el blog con distintos usuarios y roles específicos.
- Publicar en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

1.6. Diagrama de Arquitectura:



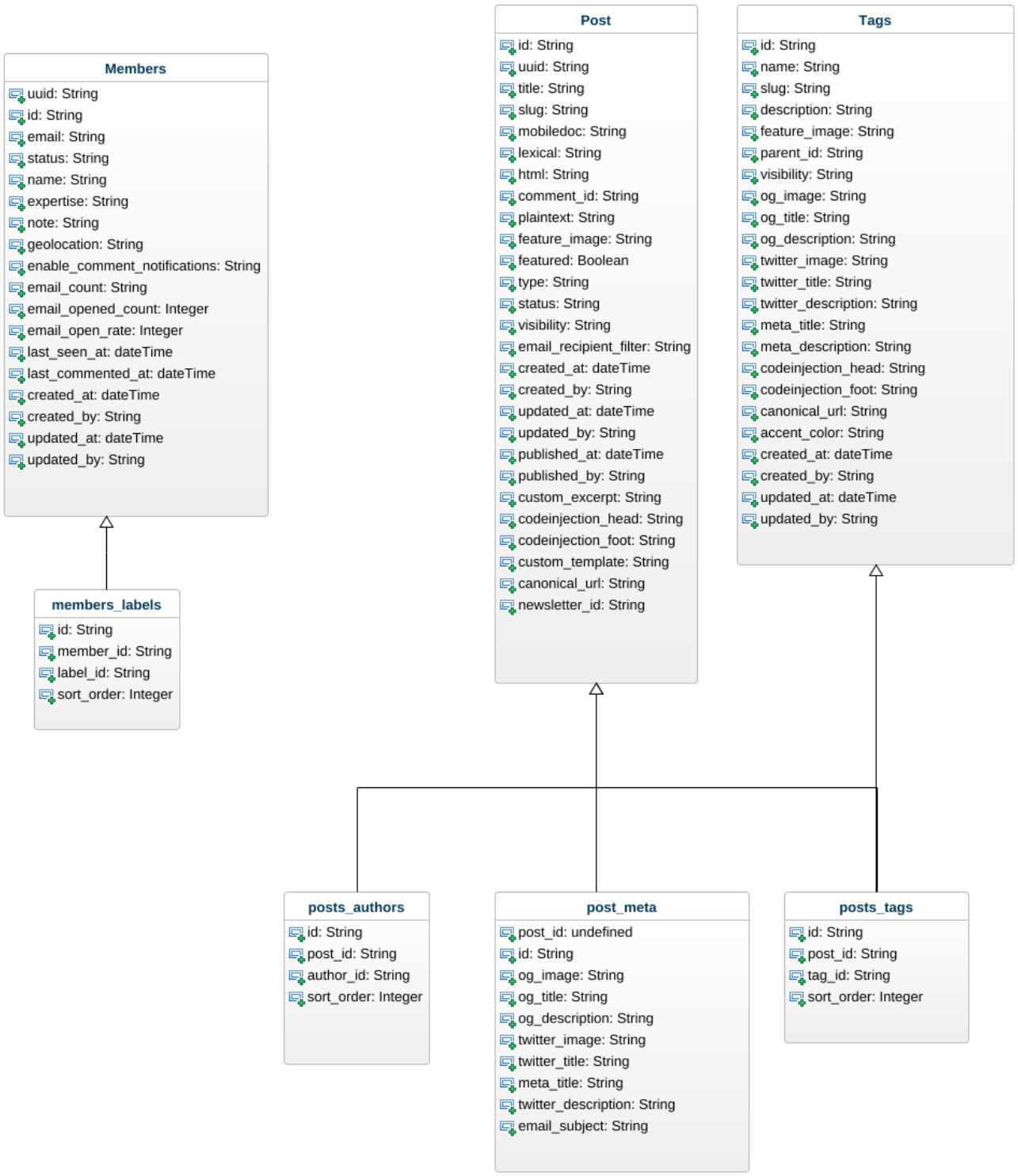
Si la imagen no se aprecia del todo una copia se encuentra dentro del zip de entrega o en el siguiente enlace: https://1drv.ms/u/s!As_QMF2T0qpGgbpEXH1l_ksDyV2GcQ?e=llg8Vf

1.7. Diagrama de Contexto:



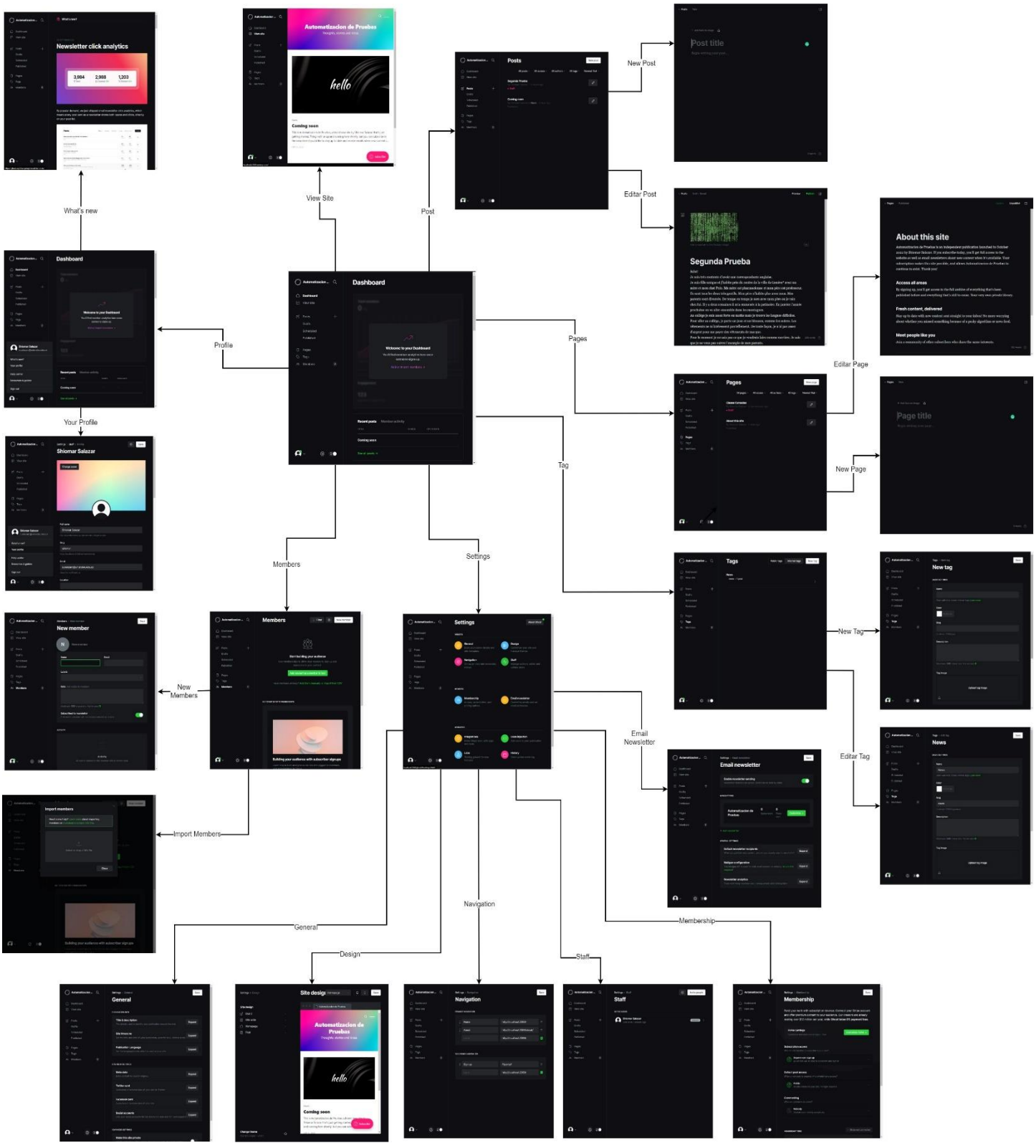
Si la imagen no se aprecia del todo una copia se encuentra dentro del zip de entrega o en el siguiente enlace: https://1drv.ms/u/s!As_QMF2T0qpGgbpCd1Ihx6P5wSTafg?e=fWpYof

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_QMF2T0gpGgbpAaZqL_GglUdPoVw?e=ylNb41

2. Contexto de la estrategia de pruebas

2.1. Objetivos:

La intención de este periodo de pruebas es el de empezar a tener un equipo estructurado de pruebas que no solo tenga un aumento de conocimientos sobre la ABP, sino que además empiece a desarrollar elementos de pruebas automatizadas que nos permita tener mejores resultados cada periodo subsecuente, es decir aumentar el catálogo de pruebas existente.

Tomando en cuenta que en la actualidad no se cuenta ni con inventario de pruebas realizadas (ni manuales ni automatizadas) ni reportes de defectos, ni documentos o artefactos de diseños, que permitan al equipo interno de pruebas alivianar la curva de aprendizaje, se espera que con el desarrollo de este equipo de pruebas internos se puedan sentar bases solidad para futuros entregables.

Generar una curva de aprendizaje sobre las pruebas exploratorias Monkey/Ripper generando que hacia una futura iteración o sprint las pruebas Monkey evolucionen en pruebas exploratorias Smart Monkey.

Como segundo objetivo se espera la entrega del siguiente paquete de 65 pruebas:

- 45 pruebas Unitarias Automatizadas.
- 18 pruebas de GUI Automatizadas.
- 1 script de pruebas Monkey.
- 1 script de pruebas Ripper.

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	150 horas	$USD 750
Hora maquina Amazon AWS	$USD 96	1 instancia	$USD 96
Hora de Trabajo Ingeniero Senior	$USD 10	40 horas	$USD 400
Renta de Equipo de Computo	$USD 50	3 equipos	$USD 150
Total			**$USD 1396**

Para 65 pruebas a ejecutar en esta estrategia, tendríamos un valor por prueba de $USD21.48

2.3.1. Recursos Humanos

En esta estrategia se consideran los siguientes equipos de trabajo:

- 3 ingenieros automatizadores Junior (50 horas/persona), los cuales deberán 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.

2.3.2. Recursos Computacionales

En esta estrategia se consideran los siguientes equipos tecnológico:

- 200 horas/máquina en Amazon AWS, estas horas maquina serán utilizadas para correr las pruebas automatizadas y/o de reconocimiento entregadas tanto por el equipo de ingenieros automatizadores junior.
    - 80 horas/máquina (nocturnas) de pruebas aleatorias de reconocimiento (75% Monkey y 25% Rippers), el porcentaje de uso se puede explicar en la sección correspondiente al análisis.
    - El resto del tiempo maquina se usará para pruebas de unidad automatizadas y pruebas GUI exploratorias.
- 3 computadores correspondientes a cada 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.
- 1 dispositivos Móviles con sistema Operativo Android (Samsung Galaxy S20) que ya formaban parte del equipo de la empresa.
- 1 dispositivos Móviles con sistema Operativo iOS (IPhone 11) que ya formaban parte del equipo de la empresa.

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

En esta estrategia no se considerará el uso de Servicios o Personal Externo, ya que la intención principal es la formación de un equipo que entregará resultado de pruebas y a su vez se incrementará sus conocimientos en la ABP y creará valor a la empresa a lo largo del tiempo.

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

Nivel	Tipo	Técnica	Objetivo
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, esto será parte del escalamiento del equipo Se espera un reporte manual con pasos y resultados.
Sistema	Funcional Caja Negra Positivas	Automatizada	En estas pruebas se espera la automatización de las pruebas exploratorias realizadas. Este apartado cumple el aumentar el catálogo de pruebas Se espera el Script de automatización a ser corrido en los servidores AWS.
Unidad	Funcional Caja Blanca Positivas y Negativas	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 / Negativo	Automatizada	Son las pruebas Ripper que harán los ingenieros Junior, esto con intención que las prueba generen los estados de la aplicación y de un mayor entendimiento de la ABP. Este apartado cumple los dos objetivos de escalar el conocimiento del equipo y aumentar el catálogo de pruebas. Se espera 1 script de pruebas Ripper con diferentes niveles y un script con las pruebas de Monkey.

2.5. Distribución de Esfuerzo

En esta estimación se espera poder dividir el tiempo (de 2 semanas) de la siguiente forma:

1. Pruebas exploratorias manuales 10 horas, ya que los ingenieros Junior necesitan 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 4 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. Pruebas aleatorias 10 horas en la creación del script pruebas Monkey (4 horas) y Ripper (2 horas).
    a. Este apartado incluirá un espacio para establecer el ambiente de prueba nocturna en donde se correrán las de tipo Ripper y Monkey, en donde se incluirán las reglas a ser ejecutadas cada noche en los servidores AWS contratados, esto deberá tardar 1 hora.
    b. Este apartado incluirá un espacio para analizar los reportes de las pruebas de Ripper 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 1 horas al final del sprint.
    c. Este apartado incluirá un espacio para analizar los reportes de las pruebas de Monkey a lo largo de la iteración de pruebas para analizar si el Monkey es funcional o necesita cambios, reportar los errores encontrados y hacer cambios mínimos necesarios, esto no debe tomar mas de 2 horas.
    d. El resto del tiempo será utilizado en la escritura de las pruebas de los scripts.
3. Pruebas de GUI Automatizadas 40 horas, ya que podrán automatizar ya sea los errores encontrados por los Ingenieros Seniors en iteraciones pasadas o errores encontrados durante la etapa anterior.
    a. Este apartado incluirá un espacio para establecer el ambiente de en donde se correrán las pruebas Automatizadas (2 horas), en donde se incluirán las reglas de ejecución permitiéndole a los ingenieros hacer uso de los Servidores AWS también.
    b. Este apartado incluirá un espacio para analizar los reportes obtenido al final del día, este análisis debe incluir errores reportados, estabilidad de las pruebas ejecutadas y el reporte de defectos en la plataforma correspondiente, se espera que esto no tome más de 1 hora al día (total 10 horas).
    c. El resto del tiempo será utilizado en la escritura de las pruebas automatizadas
4. Prueba Unitarias Automatizadas 90 horas, 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, cuáles 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.

3. Análisis de la Propuestas

3.1 Ventajas

- En esta estrategia se busca que el equipo contratado obtenga mayores conocimientos a lo largo del tiempo, haciendo que su entrega de valor a la empresa sea mayor a medida que pasa el tiempo.
- Una gran ventaja es que una vez el equipo interno realiza pruebas exploratorias, se puede enfocar en desarrollar y generar conocimiento en el entendimiento del código para así asegurar las pruebas unitarias, lo cual minimizara los errores en la aplicación, lo que con el tiempo los puede llevar a tener un mayor foco en pruebas automatizadas GUI de sistema, integración y aceptación, dado que ya tienen un control del Código con la automatización de pruebas unitarias levantadas en etapas tempranas de este proceso.
- En esta estrategia se espera un mayor número de pruebas unitarias entregadas, haciendo mayor referencia a la pirámide de automatización esperada.
- A largo plazo se espera que el equipo haya entregado suficientes pruebas unitarias para poder enfocarse en pruebas End to End.
- Al tener en esta estrategia pruebas Monkey y Rippers, podemos obtener resultados de pruebas aleatorias que serían difícil de encontrar por parte de los ingenieros junior.

- Al tener también en cuenta pruebas Rippers podemos saber en cual estado de la aplicación se presentan fallas.
- Al tener mayor cantidad de horas/maquina en pruebas Monkey vs Rippers, tendremos más aleatoriedad en los eventos generados.

3.2 Desventajas

- Los resultados iniciales son limitados ya que se espera que al principio se entrene al equipo y tengan mayor conocimiento respecto a la ABP
- la Curva de aprendizaje es larga ya que los ingenieros que se están contratando no tienen conocimientos anteriores en pruebas automatizadas.
- Necesidad de Mantenimiento de pruebas unitarias por parte del equipo de desarrollo.
- se debe hacer una inversión inicial alta en generar una curva de aprendizaje en los ingenieros internos que generen los resultados esperados en corto tiempo, esperando que a largo plazo este valor agregado a la empresa sea mayor.
- Al tener que crear un script para pruebas Monkey y uno para Rippers esto consumirá más tiempo de los ingenieros Junior.
- Al tener que separar entre pruebas Monkey y Rippers el tiempo en el que el servidor no esté en uso, esto hará que las pruebas aleatorias sean menos efectivas ya que se ejecutan por cortos periodos de tiempo.

4. Análisis de Herramientas Automatizadas para Pruebas de Reconocimiento

En esta estrategia se utilizan más pruebas Monkey ya que con estas pruebas podemos tener mayor aleatoriedad de eventos y cobertura sobre casos que no se les ocurriría a los ingenieros junior, esto genera que las pruebas Monkey corran por más tiempo que las pruebas Ripper (75% Monkey y 25% Ripper).

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

4.1. Monkey Test con Cypres

Para esta evaluación de la herramienta automatizada se utilizaron dos estrategias diferentes:

- Un Monkey simple: la herramienta solo ejecuta 100 pasos de forma aleatorias, esto sin importar la vista o si el paso es posible en la misma vista.
  El Monkey simple hace (o intenta hacer) 100 eventos iniciando sobre la página de Ghost Admin, en esta versión de la herramienta se distribuyen los eventos aleatorios de forma homogénea siendo los eventos posibles los siguientes:
  - Clic derecho, doble clic o clic izquierdo sobre posiciones X y Y aleatoria.
  - Pasar el cursor por encima de posiciones X y Y aleatoria o presionar la tecla tab.
  - Hacer clic en la tecla aleatoria.
  - Hacer clic en la tecla aleatoria del tipo letra.
  - Hacer clic en la tecla especial aleatoria o la tecla enter.
  - Cambiar la vista de la pantalla (tamaño y orientación) o refrescar la página.

- Un smarter Monkey: los eventos seguían un patrón repetible y esperado, aunque a diferencia de los Monkey este tipo de pruebas tiene más posibilidades de configuración de eventos y esto genera más eficiencia en los resultados. Los tipos de eventos generados fueron:
  - Clic derecho en posición aleatoria.
  - Clic izquierdo en posición aleatoria.
  - Scroll horizontal derecho
  - Scroll horizontal izquierdo
  - Hover en una posición aleatoria
  - Tabulador
  - Escribir una letra
  - Hacer clic en una tecla especial aleatoria
  - Hacer clic en la tecla enter.
  - Refrescar la página actual.
  - Devolverse 1 pantalla.
  - Ir a delante 1 pantalla.
  - Cambiar el viewport de la aplicación
  - Limpiar cookies
  - Limpiar el local storage
  - Llenar un elemento input
  - Limpiar un input
  - Dar clic en un link aleatorio.
  - Dar clic en un elemento botón aleatorio.

4.1.1. Ventajas de la Herramienta

- Rápida ejecución
- Facilidad de creación.
- Bajo costo.

4.1.2. Desventajas de la Herramienta

- Limitaciones de eventos generados ligado a la ABP especifica.
- Baja cobertura de casos de uso.
- Baja reusabilidad en caso de cambios en la ABP.
- Facilidad de tener falsos negativos.
- Posibilidad de quedar en un ciclo repetitivo sin salida.

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,

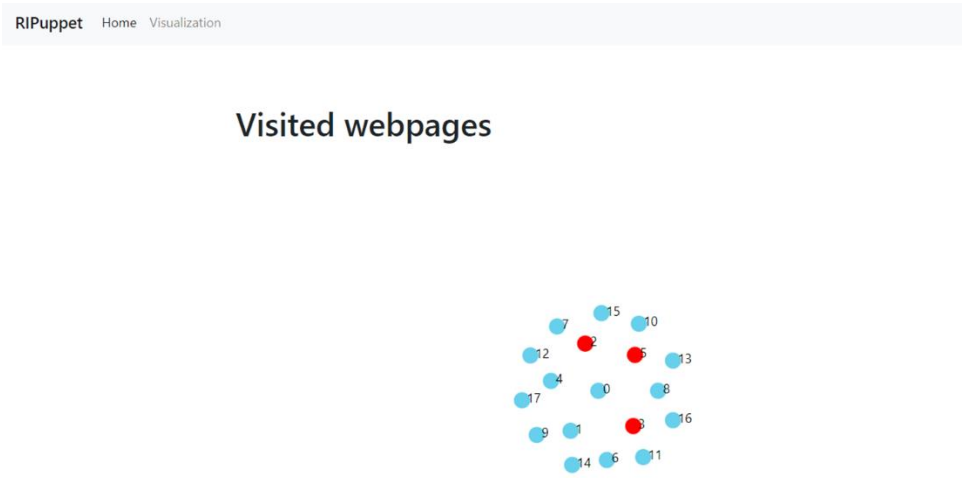
Luego corregimos el método getbuttons para poder obtener los elementos de tipo Button y lanzar eventos a dichos botones, con esto pudimos avanzar en el login y permitirle al Ripper ingresar al Admin.

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.