



# Estrategia de pruebas

1. Aplicación Bajo Pruebas
  - 1.1. Nombre Aplicación
  - 1.2. Versión
  - 1.3. Descripción
  - 1.4. Funcionalidades Core
  - 1.5. Diagrama de Arquitectura
  - 1.6. Diagrama de Contexto
  - 1.7. Modelo de Datos
  - 1.8. Modelo de GUI
2. Contexto de la estrategia de pruebas
  - 2.1. Objetivos
  - 2.2. Duración de la iteración de pruebas
  - 2.3. Presupuesto de pruebas
  - 2.4. TNT (Técnicas, Niveles y Tipos) de pruebas
  - 2.5. Distribución de Esfuerzo
  - 2.6. Video justificación

## 1. Aplicación Bajo Pruebas

### 1.1. Nombre Aplicación

Ghost

### 1.2. Versión

Ghost 3.42.5


### 1.3. Descripción

Ghost es una plataforma web orientada en ofrecer herramientas open source a periodistas y escritores, para eso soporta una plataforma que permite a sus usuarios publicar, editar, dar soporte, compartir un blog con un contenido propio entre otras funcionalidades que mencionadas en este reporte.

### 1.4. Funcionalidades Core

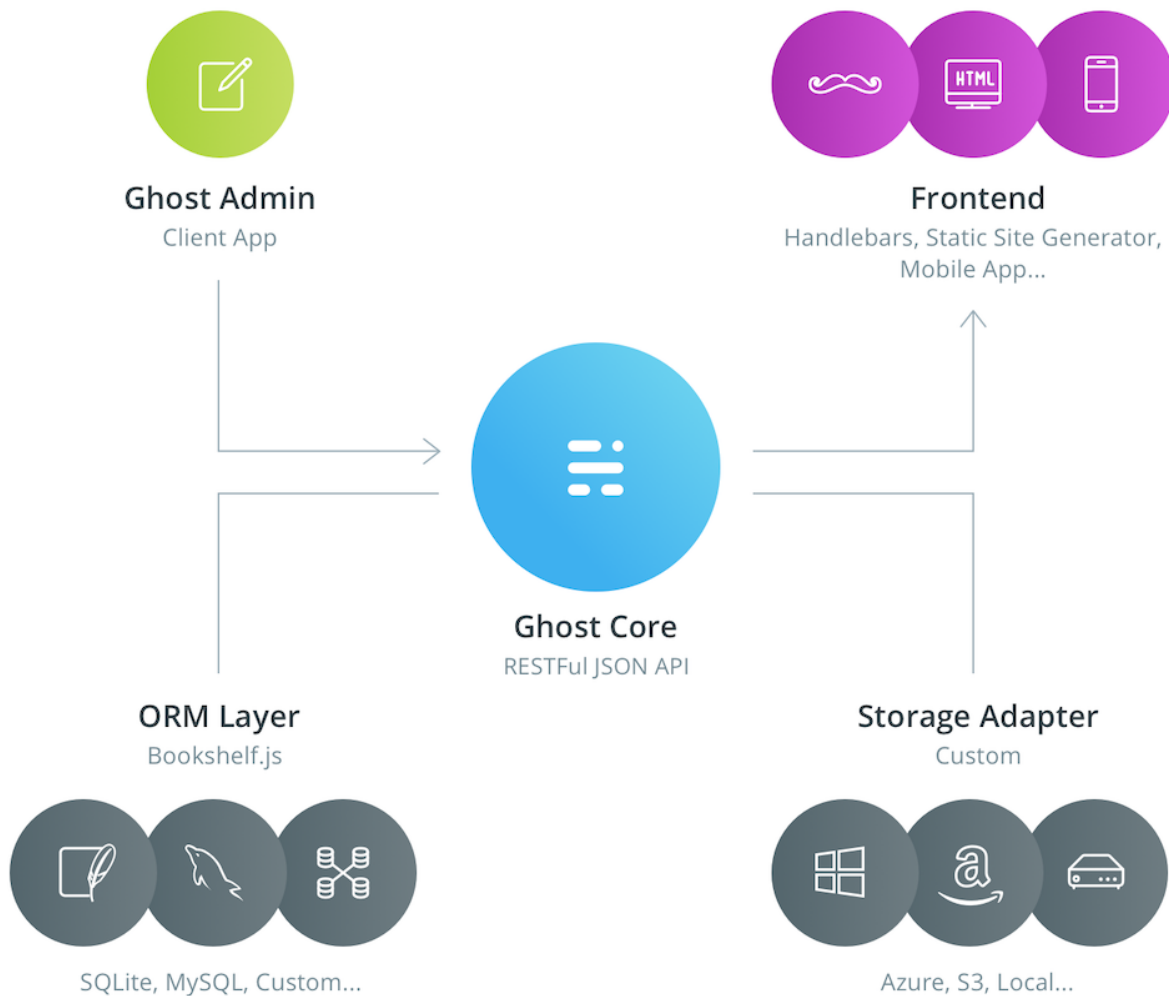
A continuación se presenta el listado inicial con las funcionalidades presentadas por el sistema, este listado pretende plantear una base inicial de los escenarios que serán probados y analizados mas no presenta el listado completo de toda la plataforma.

### Lista de funcionalidades

|  ID |  Nombre de la funcionalidad |  Descripción   |
|--|--|---|
| F01  | <u><a href="#">Crear post</a></u>  | Crear una nueva entrada en la página web con un título y contenido específico.  |
| F02  | <u><a href="#">Editar post</a></u>   | Editar el título y contenido de un post que fue previamente creado por un usuario.  |
| F03  | <u><a href="#">Filtrar posts</a></u>   | Seleccionar posts existentes que cumplen con los valores de los campos status (draft, published, scheduled) y tag.  |
| F04  | <u><a href="#">Crear página</a></u>  | Crear una nueva página para el blog.  |
| F05  | <u><a href="#">Editar página</a></u>   | Actualizar la información de una página existente, tanto su título y contenido, como su metadata.   |
| F06  | <u><a href="#">Filtrar páginas</a></u>   | Seleccionar páginas existentes que cumplen con los valores del campo status.  |
| F07  | <u><a href="#">Cambiar contraseña</a></u>  | Actualizar la contraseña para acceder al sistema, para esto es necesario tener la contraseña actual y que la nueva contraseña tenga por lo menos 10 caracteres. |
| F08  | <u><a href="#">Crear tag</a></u>   | Crear un nuevo tag que va a ser usado para distinguir páginas y posts.  |
| F09  | <u><a href="#">Editar perfil</a></u>   | Actualizar la información del perfil del usuario, tal como slug, correo y website.  |

## 1.5. Diagrama de Arquitectura

El siguiente diagrama muestra la arquitectura de Ghost, esta fue tomada del sitio oficial:

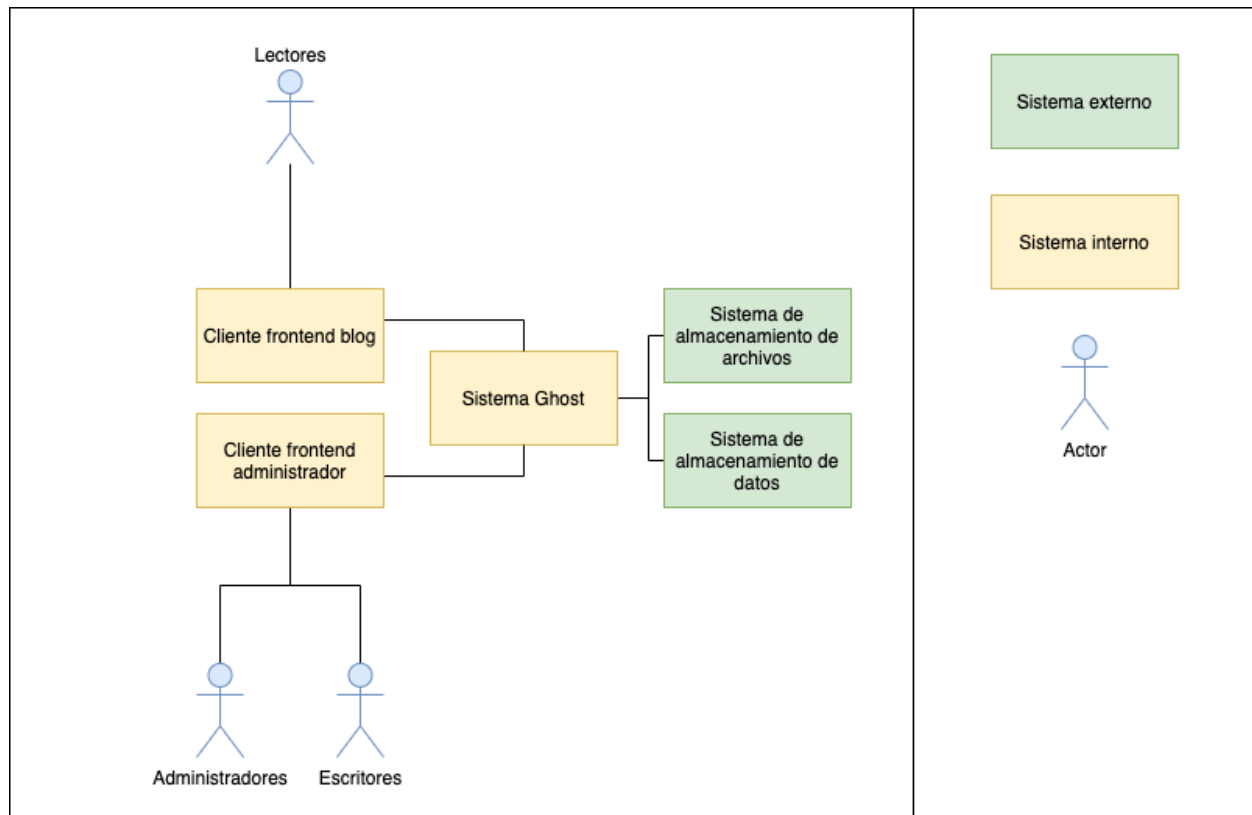


Tomado de <https://ghost.org/docs/architecture>

- **Frontend:** Es encargado de presentar la interfaz al usuario lector que interactuará con el sistema.
- **Ghost admin:** Es encargado de presentar la interfaz al usuario administrador encargado de publicar y hacer las modificaciones necesarias al blog antes de ser publicado.
- **Ghost Core:** API REST que se encarga de proveer la información que será visualizada o servicios necesarios para interactuar con el sistema presentado por el frontend.
- **ORM Layer:** Encargado de la persistencia y manipulación de datos que son almacenados por el sistema haciendo uso de una base de datos relacional.

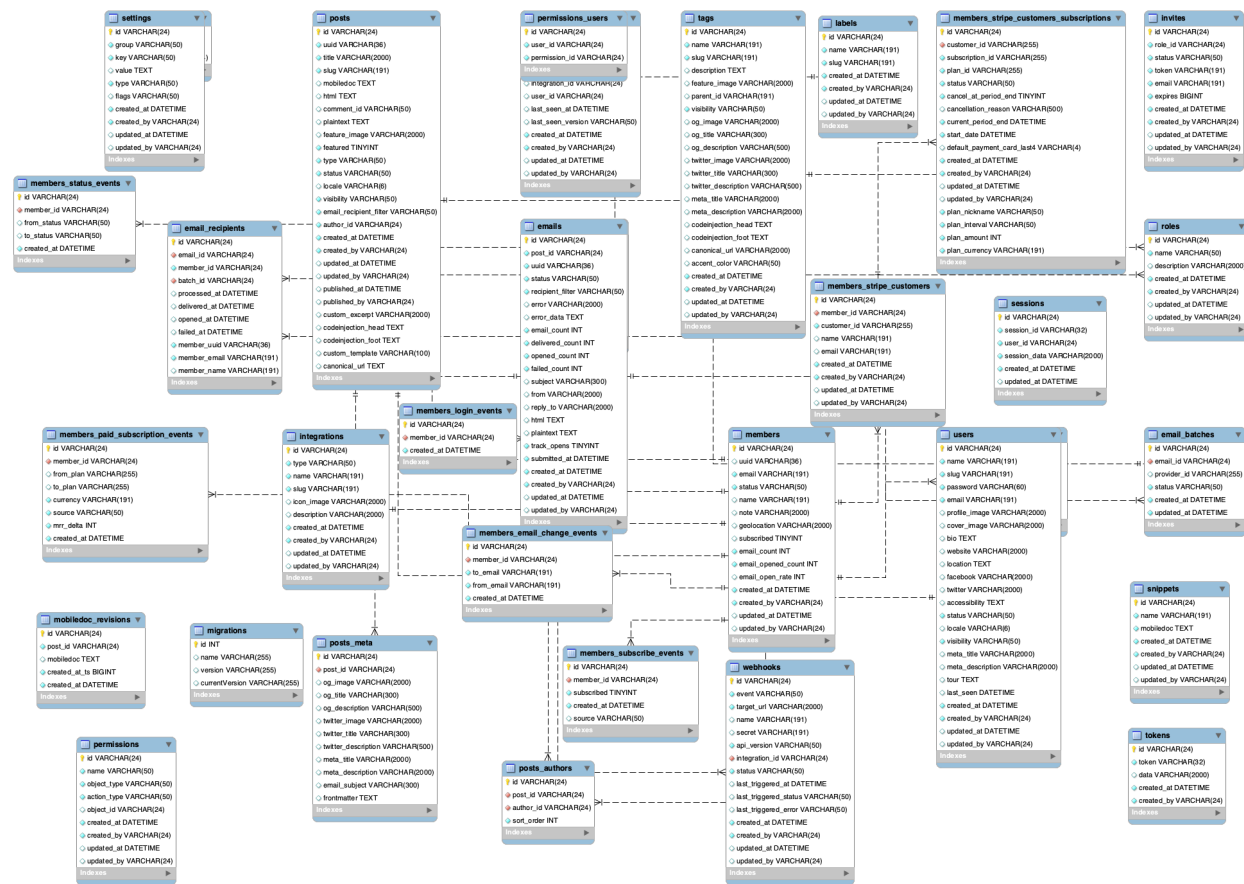
## 1.6. Diagrama de Contexto

La siguiente imagen muestra el diagrama de contexto de la aplicación Ghost, se identificaron 3 actores principales, Lectores, Administradores y Escritores, los cuales interactúan con el sistema por medio de clientes front-end:



## 1.7. Modelo de Datos

En el siguiente diagrama se presenta el modelo de datos extraído de la base de datos de Ghost, en este se pueden encontrar las tablas que hacen parte de la base de datos así como las relaciones entre ellas:



## 1.8. Modelo de GUI

En el siguiente link se encuentra el diagrama que representa el modelo de GUI:

<https://drive.google.com/file/d/1WPwqCWIH7qdwnsRw3NqWaZHZCqgdjDTFI/view>, de igual forma se adjunta en el zip de la entrega.

## 2. Contexto de la estrategia de pruebas

### 2.1. Objetivos

Los objetivos principales del periodo de pruebas es en primera instancia probar la parte funcional de la aplicación asegurándose que los flujos y funcionalidades principales tengan un correcto funcionamiento dando énfasis a la detección de defectos. A continuación se hace un resumen de los objetivos:

- **OBJ01:** Ejecutar los flujos principales de la aplicación detallados en el listado de funcionalidades.
- **OBJ02:** Detectar defectos en las funcionalidades principales de la aplicación administradora.

- **OBJ03:** Detectar cambios visuales en diferentes versiones de la aplicación.
- **OBJ04:** Generar la primera versión del inventario de pruebas por medio de pruebas manuales exploratorias.
- **OBJ05:** Encontrar y solucionar errores antes de que salgan a producción por medio del uso de pruebas.

## 2.2. Duración de la iteración de pruebas

La iteración de pruebas se desarrollará en 8 semanas empezando en la semana del 24 de Mayo del 2021 al 30 de Mayo del 2021.

## 2.3. Presupuesto de pruebas

### 2.3.1. Recursos Humanos

Para la realización de las pruebas se cuenta con cuatro ingenieros automatizadores senior, lo cual equivale alrededor de 4 años o más de experiencia de cada ingeniero, se contara con la disponibilidad de 64 horas/persona por semana, para la segunda interacción del proyecto que esta comprendida por 8 semanas.

#### Listado de personal

| <u>Aa</u> Personal     | <u>≡</u> Disponibilidad (Horas) | <u>≡</u> Experiencia |
|------------------------|---------------------------------|----------------------|
| <u>Automatizador 1</u> | 64                              | Senior               |
| <u>Automatizador 2</u> | 64                              | Senior               |
| <u>Automatizador 3</u> | 64                              | Senior               |
| <u>Automatizador 4</u> | 64                              | Senior               |

Con lo anterior se llega a un total de 256 horas hombre disponibles para pruebas que también equivale a 15.360 minutos, 3.840 por automatizador.

### 2.3.2. Recursos Computacionales

El listado de dispositivos y capacidades computacionales con los que se cuentan para la realización de las pruebas se lista a continuación.

#### Hardware

| <u>Aa</u> Nombre | <u>≡</u> Disponibilidad (Horas) |
|------------------|---------------------------------|
| <u>Máquina 1</u> | 64                              |
| <u>Máquina 2</u> | 64                              |

| Aa Nombre        | ☰ Disponibilidad (Horas) |
|------------------|--------------------------|
| <u>Máquina 3</u> | 64                       |
| <u>Máquina 4</u> | 64                       |

En principio se cuenta con la máquina dedicada de cada automatizador, como cada uno esta disponible 64 horas se tiene este tiempo también disponible por máquina.

Por otro lado se lista los software y servicios cloud disponibles para probar la plataforma.

### Software

| Aa Nombre             | ☰ Precio             | ☰ Tipo              | ☰ Link  |
|-----------------------|----------------------|---------------------|---|
| <u>Cypress</u>        | \$0<br>(Open source) | E2E testing         | <a href="https://www.cypress.io/">https://www.cypress.io/</a>   |
| <u>Playwright</u>     | \$0<br>(Open source) | E2E testing         | <a href="https://playwright.dev/">https://playwright.dev/</a>   |
| <u>Puppeteer</u>      | \$0<br>(Open source) | E2E testing         | <a href="https://pptr.dev/">https://pptr.dev/</a>   |
| <u>Kraken</u>         | \$0<br>(Open source) | E2E testing         | <a href="https://github.com/TheSoftwareDesignLab/KrakenMobile">https://github.com/TheSoftwareDesignLab/KrakenMobile</a>           |
| <u>RIPuppet</u>       | \$0<br>(Open source) | GUI Ripping         | <a href="https://github.com/TheSoftwareDesignLab/RIPuppetCoursera/">https://github.com/TheSoftwareDesignLab/RIPuppetCoursera/</a> |
| <u>monkey-cypress</u> | \$0<br>(Open source) | Random testing      | <a href="https://github.com/TheSoftwareDesignLab/monkey-cypress">https://github.com/TheSoftwareDesignLab/monkey-cypress</a>       |
| <u>Backstop.js</u>    | \$0<br>(Open source) | Visual regression   | <a href="https://github.com/garris/BackstopJS">https://github.com/garris/BackstopJS</a>   |
| <u>Resemble.js</u>    | \$0<br>(Open source) | Visual regression   | <a href="https://github.com/rsmb/Resemble.js">https://github.com/rsmb/Resemble.js</a>   |
| <u>Mockaroo</u>       | \$0 (1k filas)       | Generación de datos | <a href="https://www.mockaroo.com/">https://www.mockaroo.com/</a>   |

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

Para esta estrategia de pruebas no se cuenta con recursos económicos dedicados para tercerizar tareas ya que se cuenta con testers in-house.

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

En la siguiente tabla se presenta las técnicas, niveles y tipos de pruebas que serán utilizados para cumplir con los objetivos propuestos:

### TNT

| <u>Aa</u> Nivel   | <u>≡</u> Tipo                                      | <u>≡</u> Técnica  | <u>≡</u> Objetivo           |
|-------------------|--|-------------------|-----------------------------|
| <u>Sistema</u>    | Funcionales, caja negra, positivas y negativas     | Exploratorias     | OBJ01, OBJ02, OBJ04 y OBJ05 |
| <u>Sistema</u>    | Funcionales y caja negra                           | Monkey testing    | OBJ02 y OBJ05               |
| <u>Sistema</u>    | Funcionales, caja negra positivas y negativas      | GUI Ripping       | OBJ01, OBJ02, OBJ04 y OBJ05 |
| <u>Sistema</u>    | E2E, Funcionales, caja negra positivas y negativas | E2E Automatizadas | OBJ01, OBJ02, OBJ04 y OBJ05 |
| <u>Aceptación</u> | No funcionales, caja negra                         | Regresión visual  | OBJ02, OBJ03 y OBJ05        |
| <u>Aceptación</u> | Funcionales, caja negra, positivas y negativas     | Manuales          | OBJ01, OBJ02, OBJ04 y OBJ05 |
| <u>Untitled</u>   |  |                   |                             |

- **OBJ01:** Ejecutar los flujos principales de la aplicación detallados en el listado de funcionalidades.

Para ejecutar los principales flujos de la aplicación se utiliza pruebas manuales, E2E automatizadas y pruebas de reconocimiento. De esta forma, se puede probar de diferentes formas que los flujos principales de la aplicación estén funcionando de forma adecuada.

- **OBJ02:** Detectar defectos en las funcionalidades principales de la aplicación administradora.

Para detectar defectos en principio se ejecutan pruebas manuales exploratorias negativas, con el fin de encontrar posibles defectos usando datos inválidos o secuencias de pasos complicadas. Después, se hace uso de pruebas de reconocimiento automatizadas de tal forma que se pueden generar eventos en menos tiempo y generan eventos random que un tester humano puede que no tenga en cuenta. Después, se automatizan pruebas E2E usando flujos que puedan fallar, para comprobar si en efecto la



aplicación falla o si tiene algún mecanismo de feedback o recuperación. Finalmente, a estas pruebas se le añaden datos a priori, dinámicos y aleatorios, para así probar si diferentes entradas pueden causar fallas en la aplicación.

- **OBJ03:** Detectar cambios visuales en diferentes versiones de la aplicación.

Se implementa la toma de screenshots luego de cada paso de las pruebas E2E y se ejecuta en diferentes versiones de la aplicación. Después, se comparan las imágenes de ambas versiones usando herramientas de regresión visual y de esta forma se puede analizar si las versiones de la UI son estables a lo largo del tiempo o si hay cambios drásticos en los componentes.

- **OBJ04:** Generar la primera versión del inventario de pruebas por medio de pruebas manuales exploratorias.

Una vez desarrollado el set de pruebas positivas y negativas E2E y de exploración manuales, se contará con un primer inventario de pruebas que podrá ejecutarse en próximas iteraciones a menos que haya un cambio significativo en la GUI.

- **OBJ05:** Encontrar y solucionar errores antes de que salgan a producción por medio del uso de pruebas basadas.

En caso de encontrarse defectos con las pruebas realizadas se reportarán los errores a los desarrolladores para que puedan solucionar los errores antes del release y así ejecutar de nuevo las pruebas validando y verificando su funcionamiento.

## 2.5. Distribución de Esfuerzo

Para la distribución de esfuerzo se planea utilizar como patrón la pirámide de automatización contemplando para esta estrategia la realización de pruebas manuales, pruebas de reconocimiento, pruebas de extremo a extremo, pruebas de regresión visual y escenarios de validación de datos.

Las siguientes tablas presentan la distribución y tareas que desarrollará cada automatizador senior durante cada semana.

### Semana 1

| <u>Aa</u> Tarea  |  Ingeniero 1 |  Ingeniero 2 |  Ingeniero 3 |  Ingeniero 4 |
|--|---|---|---|---|
| <u>Planeación</u>  | 1   | 1   | 1   | 1   |
| <u>Elaboración de la estrategia de pruebas con el formato preestablecido</u> | 4   | 4   | 4   | 4   |





| <u>Aa</u> Tarea  | Ingeniero<br>1 | Ingeniero<br>2 | Ingeniero<br>3 | Ingeniero<br>4 |
|--|----------------|----------------|----------------|----------------|
| <u>Elaboración de inventario de pruebas exploratorias para las funcionalidades definidas con un vídeo por cada una</u> | 3              | 3              | 3              | 3              |
| <u>TOTAL</u>   | 8              | 8              | 8              | 8              |

## Semana 2

| <u>Aa</u> Tarea   | Ingeniero<br>1 | Ingeniero<br>2 | Ingeniero<br>3 | Ingeniero<br>4 |
|---|----------------|----------------|----------------|----------------|
| <u>Elaboración de script pruebas aleatorias automatizadas (Monkey).</u>     | 1              | 1              | 0              | 0              |
| <u>Elaboración de script pruebas exploración de GUI (Ripper).</u>           | 0              | 0              | 1              | 1              |
| <u>Ejecución de Monkey y monitoreo de pruebas</u>                           | 2              | 2              | 0              | 0              |
| <u>Ejecución de Ripper y monitoreo de pruebas</u>                           | 0              | 0              | 2              | 2              |
| <u>Análisis y documentación resultados Monkey.</u>                          | 1              | 1              | 0              | 0              |
| <u>Análisis y documentación resultados Ripper</u>                           | 0              | 0              | 1              | 1              |
| <u>Definición y ejecución de pruebas manuales sobre las funcionalidades</u> | 2              | 2              | 2              | 2              |
| <u>Análisis y documentación resultados de pruebas manuales</u>              | 1              | 1              | 1              | 1              |
| <u>Creación de reporte de issues encontrados</u>                            | 1              | 1              | 1              | 1              |
| <u>TOTAL</u>  | 8              | 8              | 8              | 8              |

## Semana 3

| <u>Aa</u> Tarea  | Ingeniero<br>1 | Ingeniero<br>2 | Ingeniero<br>3 | Ingeniero<br>4 |
|--|----------------|----------------|----------------|----------------|
| <u>Creación de los scripts para las pruebas E2E de los escenarios asignados, en la herramienta Playwright.</u> | 4              | 4              | 4              | 4              |
| <u>Ejecución de los scripts de las pruebas E2E en el navegador Chrome.</u>                                     | 1              | 1              | 1              | 1              |
| <u>Ejecución de los scripts de las pruebas E2E en el navegador Firefox.</u>                                    | 1              | 1              | 1              | 1              |

| <u>Aa</u> Tarea   |  Ingeniero 1 |  Ingeniero 2 |  Ingeniero 3 |  Ingeniero 4 |
|---|---|---|---|---|
| <u>Análisis y documentación de los resultados.</u>  | 1   | 1   | 1   | 1   |
| <u>Reporte de issues encontrados en la ejecución de pruebas E2E en el sistema de issues tracker</u> | 1   | 1   | 1   | 1   |
| <u>TOTAL</u>  | 8   | 8   | 8   | 8   |

En la semana 3, el equipo iniciara con las pruebas de extremo a extremo (E2E), iniciando con la creación de los scripts para la herramienta Playwright del 50% de los escenarios que fueron asignados a cada uno de los integrantes del equipo en la planeación. Luego se realizara la ejecución de los scripts generados en los navegadores Chrome y Firefox. Una vez se culmine con la ejecución de las pruebas se realizara por cada integrante un análisis de los resultados obtenidos, en caso de encontrar issues se deberán dejar registrados en el sistema de issues tracker.

#### Semana 4

| <u>Aa</u> Tarea  |  Ingeniero 1 |  Ingeniero 2 |  Ingeniero 3 |  Ingeniero 4 |
|--|---|---|---|---|
| <u>Creación de los scripts para las pruebas E2E de los escenarios asignados, en la herramienta kraken.</u> | 4   | 4   | 4   | 4   |
| <u>Ejecución de los scripts de las pruebas E2E en el navegador Chrome.</u>                                 | 1   | 1   | 1   | 1   |
| <u>Ejecución de los scripts de las pruebas E2E en el navegador Firefox.</u>                                | 1   | 1   | 1   | 1   |
| <u>Análisis y documentación de los resultados.</u>   | 1   | 1   | 1   | 1   |
| <u>Reporte de issues encontrados en la ejecución de pruebas E2E en el sistema de issues tracker</u>        | 1   | 1   | 1   | 1   |
| <u>TOTAL</u>   | 8   | 8   | 8   | 8   |

En la semana 4, el equipo continuara con las pruebas de extremo a extremo (E2E), iniciando con la creación de los scripts para la herramienta Kraken del 50% restante de los escenarios que fueron asignados a cada uno de los integrantes del equipo en la planeación. Luego se realizara la ejecución de los scripts generados en los navegadores Chrome y Firefox. Una vez se culmine con la ejecución de las pruebas se realizara por

cada integrante un análisis de los resultados obtenidos, en caso de encontrar issues se deberán dejar registrados en el sistema de issues tracker.

### Semana 5

| <u>Aa</u> Tarea  | Ingeniero<br>1 | Ingeniero<br>2 | Ingeniero<br>3 | Ingeniero<br>4 |
|--|----------------|----------------|----------------|----------------|
| <u>Modificar scripts para toma de screenshots</u>                | 1              | 1              | 1              | 1              |
| <u>Ejecución pruebas VRT en Chrome</u>                           | 1              | 1              | 1              | 1              |
| <u>Ejecución pruebas VRT en Firefox</u>                          | 1              | 1              | 1              | 1              |
| <u>Ejecución pruebas VRT con pantalla responsive modo mobile</u> | 1              | 1              | 1              | 1              |
| <u>Generación reporte VRT (Resemble, Backstop).</u>              | 3              | 3              | 3              | 3              |
| <u>Análisis y documentación resultados</u>                       | 1              | 1              | 1              | 1              |
| <u>TOTAL</u>   | 8              | 8              | 8              | 8              |

En la semana 5 el equipo se enfocará en implementar la toma de screenshots luego de cada paso de las pruebas E2E. Se definirán diferentes rutas para almacenar las imágenes de la versión 3.3.0 y 3.42.5 para identificar a que versión pertenecen las imágenes. Además, se ejecutarán las pruebas en diferentes navegadores, dado que en muchas ocasiones las aplicaciones varían según el navegador y también en modo mobile para así detectar cambios correspondientes a la adaptación a la pantalla (responsive). Finalmente, se generarán los reportes que comparan las imágenes de las diferentes versiones usando Resemble y Backstop.

### Semana 6

| <u>Aa</u> Tarea                              | Ingeniero<br>1 | Ingeniero<br>2 | Ingeniero<br>3 | Ingeniero<br>4 |
|--|----------------|----------------|----------------|----------------|
| <u>Almacenamiento screenshots</u>            | 1              | 1              | 1              | 1              |
| <u>Scripts generación de datos a priori</u>  | 4              | 4              | 4              | 4              |
| <u>Scripts generación de datos dinámicos</u> | 3              | 3              | 3              | 3              |
| <u>TOTAL</u>                                 | 8              | 8              | 8              | 8              |

En la semana 6 se subirán los screenshots tomados en la semana anterior, dado que se toman luego de cada paso y teniendo en cuenta que son 120 escenarios, el tiempo para

subir la imágenes es alto y también el uso de red. Después, los esfuerzos se enfocarán en implementar la generación y uso de datos a priori y dinámicos usando Mockaroo en las pruebas E2E. Para esto, hay que definir los schemas que van a almacenar los datos, los endpoints que se van a consumir, la frecuencia de generación y los .json que se van a usar. Después, hay que modificar las pruebas E2E para que usen estos datos.

## Semana 7

| <u>Aa</u> Tarea  |  Ingeniero 1 |  Ingeniero 2 |  Ingeniero 3 |  Ingeniero 4 |
|--|---|---|---|---|
| <u>Continuar scripts generación de datos dinámicos</u> | 1   | 1   | 1   | 1   |
| <u>Scripts generación de datos aleatorios</u>          | 4   | 4   | 4   | 4   |
| <u>Análisis y documentación resultados</u>             | 1   | 1   | 1   | 1   |
| <u>Reporte de issues</u>                               | 2   | 2   | 2   | 2   |
| <u>TOTAL</u>   | 8   | 8   | 8   | 8   |

En la semana 7 continuaremos con las pruebas de generación de datos, sin embargo, esta vez utilizaremos una técnica diferente a las usadas anteriormente que es Escenarios aleatorios. Para ello se tomarán los escenarios de pruebas realizados anteriormente con Playwright y se modificaran para que utilicen la librería FakerJS que generará datos en runtime según la categoría seleccionada en el script. Al finalizar este proceso de pruebas los ingenieros se tomaran un tiempo para analizar y documentar los resultados y finalmente reportar todas las incidencias encontradas al realizar las pruebas en el issue tracker.

## Semana 8

| <u>Aa</u> Tarea                  |  Ingeniero 1 |  Ingeniero 2 |  Ingeniero 3 |  Ingeniero 4 |
|----------------------------------|---|---|---|---|
| <u>Retrospectiva</u>             | 3   | 3   | 3   | 3   |
| <u>Presentación</u>              | 2   | 2   | 2   | 2   |
| <u>Video resultados</u>          | 2   | 2   | 2   | 2   |
| <u>Planeación próximos pasos</u> | 1   | 1   | 1   | 1   |
| <u>TOTAL</u>                     | 8   | 8   | 8   | 8   |

En esta ultima semana se realizará una retrospectiva entre todos los integrantes para analizar y concluir como fue todo el proceso de las 8 semanas y encontrar puntos de mejora, seguido a esto, se realizará una presentación de toda la estrategia de pruebas

que fue realizada a lo largo de este tiempo y que sera presentada a los directivos de la compañía. Una vez realizada la presentación los integrantes grabaran un video en el que expliquen la presentación realizada anteriormente y para finalizar se reunirán a discutir cuales serán los siguientes pasos para las próximas iteraciones del proyecto recopilando los resultados de la suite de pruebas inicial.

## **2.6. Video justificación**

En el siguiente enlace se puede encontrar el video con la justificación de las decisiones tomadas para la estrategia de pruebas:

<https://youtu.be/LNToTWBDICQ>