Estrategia de Pruebas

1. Aplicación Bajo Pruebas
   1. **Nombre Aplicación:** Ghost.
   2. **Versión:** 5.18 - 5.22 -3.42
   3. **Descripción**: Ghost es una poderosa aplicación para creación de contenido para personas que quieren compartir y crecer alrededor de su contenido, funciona como una API poderosa que ofrece servicios para tener un front altamente customizable.
   4. **Funcionalidades Core**:

* Posts
* Pages
* Tags
* Members
* Dashboard
* Settings: Navigation, General, Brand, Staff, Membership, Email Newsletter, Integrations, Code Injection, Lab, History
* View site
* Profile
* Login/Sing up
  1. **Diagrama de Arquitectura:**

[Diagrama de Arquitectura](https://uniandes-my.sharepoint.com/:i:/g/personal/h_patarroyo_uniandes_edu_co/EbWDWNi3kkNGqzI5q1f4D8YBGEt1yBIgh7kSoi78yWEkkA?e=g7zSIG)

* 1. **Diagrama de Contexto:**

[Diagrama de Contexto](https://uniandes-my.sharepoint.com/:i:/g/personal/h_patarroyo_uniandes_edu_co/EQ3rt7dDer9Fv0E6DgjYGlIB8l-_4fJhVGNJFcTaHf4QOg?e=y3cBqH)

* 1. **Modelo de Datos:**

[Modelo de datos](https://uniandes-my.sharepoint.com/:i:/g/personal/h_patarroyo_uniandes_edu_co/EWgc3DLRC9BDq7ZVwHGIg_8B05IQSOLxBHvc9xSzl7a-NA?e=3S1vra)

* 1. **Modelo de GUI:**

[ModeloGUI](https://uniandes-my.sharepoint.com/:u:/g/personal/h_patarroyo_uniandes_edu_co/EQFGP3NcKlBAmTCFPgqINzcBX7udkAZaJcd0Nt2XipHvYg?e=0HpggT)

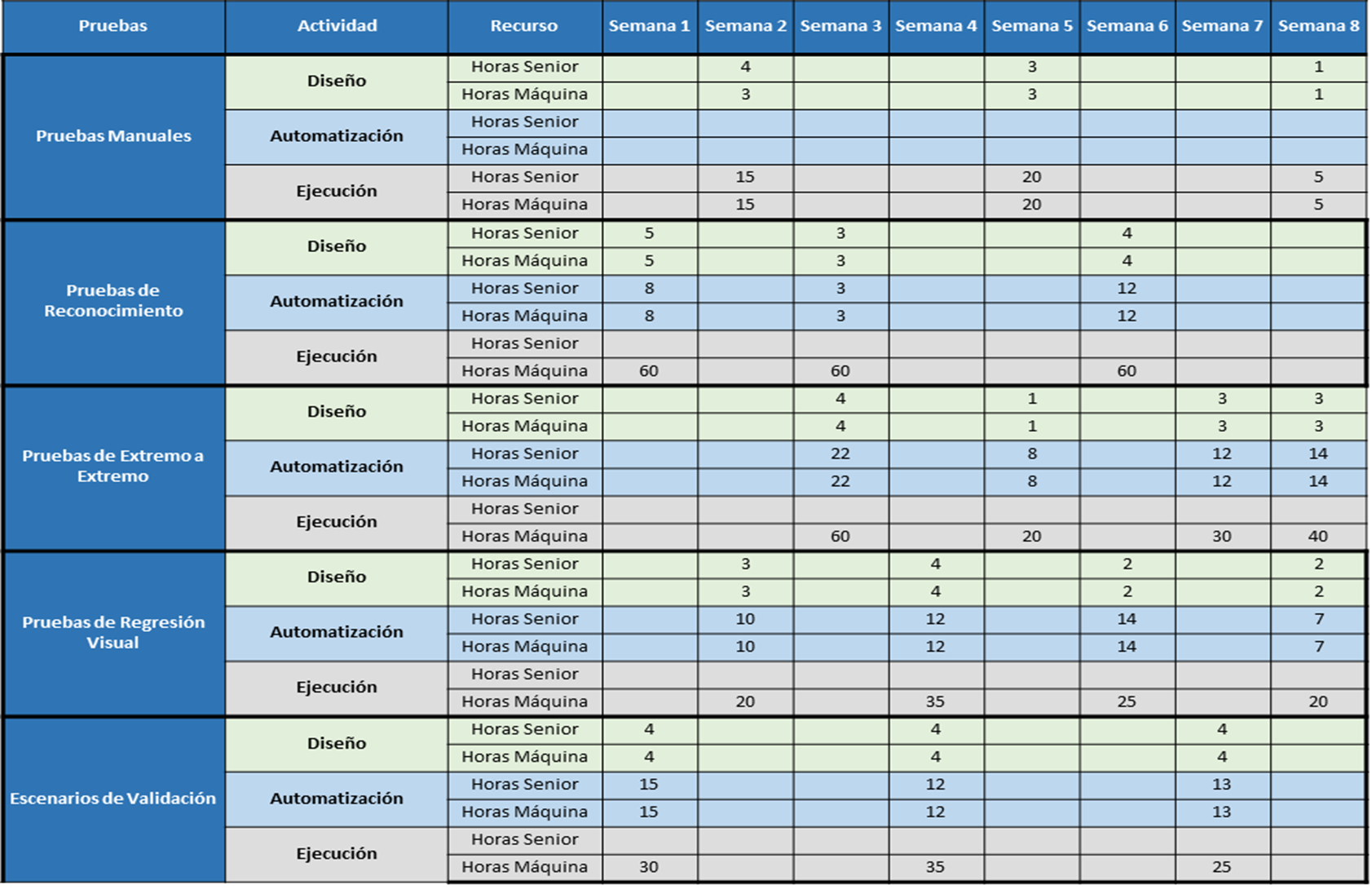
1. **Contexto de la estrategia de pruebas**
   1. **Objetivos:**

Teniendo en cuenta la etapa del proceso y el tiempo disponible para esta iteración se determinó que el alcance de esta iteración está restringido a las funcionalidades principales de la aplicación. Las cuales se listan a continuación:

* Login / Sign In
* Posts
* Pages
* Tags
* Settings
* Members
* Integrations
* Profile

Considerando el alcance anteriormente descrito, los objetivos definidos para esta estrategia de pruebas son:

* Validar el correcto desarrollo de las funcionalidades con respecto a los requerimientos establecidos por el usuario mediante la realización de pruebas manuales. (Validación Interna).
* Evaluar el desempeño de las funcionalidades principales de la aplicación bajo escenarios de prueba negativos y positivos utilizando las siguientes estrategias para la generación del pool de datos: a priori, aleatorio dinámico y aleatorio.
* Implementar pruebas automatizadas E2E para garantizar que los flujos principales de la aplicación se prueben de manera continua.
* Generar estrategias de mejora continua que permitan reducir la probabilidad de ocurrencia de bugs en iteraciones futuras.
* Determinar el grado de cumplimiento de las funcionales principales de acuerdo al punto de vista del usuario mediante la realización de UAT (User Acceptance Testing). (Validación Externa).
* Agregar mayor cobertura de pruebas a través del uso de Rippers y Monkeys.
* Validar que los cambios realizados a la aplicación desde el último release no afectan de manera negativa la apariencia visual de la Interfaz de usuario a través de la implementación de VRT.
  1. **Duración de la iteración de pruebas:**



* 1. **Presupuesto de pruebas:**
     1. **Recursos Humanos**

Para la ejecución de esta estrategia contamos con 4 Testers Senior con una dedicación de 8 horas por persona semanalmente.

* + 1. **Recursos Computacionales**

|  |  |
| --- | --- |
| Dispositivos | Capacidad Computacional |
| AWS EC2 C5 | * Las instancias C5 ofrecen una variedad de procesadores en función del tamaño de la instancia. * Las instancias C5 y C5d 12xlarge, 24xlarge y metal incluyen procesadores Intel Xeon Scalable personalizados de 2.ª generación (Cascade Lake 8275CL) con una frecuencia Turbo estable en todos los núcleos de 3,6 GHz y una frecuencia Turbo de núcleo individual hasta 3,9 GHz. * Otros tamaños de instancias C5 se lanzarán en los procesadores Intel Xeon Scalable de 2.ª generación (Cascade Lake 8223CL) o en un procesador Intel Xeon Platinum serie 8000 (Skylake 8124M) de primera generación con una frecuencia Turbo estable en todos los núcleos de 3,4 GHz y una frecuencia Turbo de núcleo individual hasta 3,5 GHz. * El nuevo tamaño de instancia más grande 24xlarge ofrece 96 CPU virtual, 192 GiB de memoria, y SSD opcional basado en NVMe local de 3,6 TB * Necesita unas AMI HVM que incluyan controladores para ENA y NVMe * Con las instancias C5d, los SSD locales basados en NVMe están conectados físicamente al servidor host y suministran almacenamiento a nivel de bloque durante la vida útil de la instancia C5 * Elastic Network Adapter (ENA) proporciona instancias C5 con hasta 25 Gbps de ancho de banda de red y hasta 19 Gbps de ancho de banda dedicado a Amazon EBS. * Con tecnología de AWS Nitro System, una combinación de hardware dedicado e hipervisor ligero. |

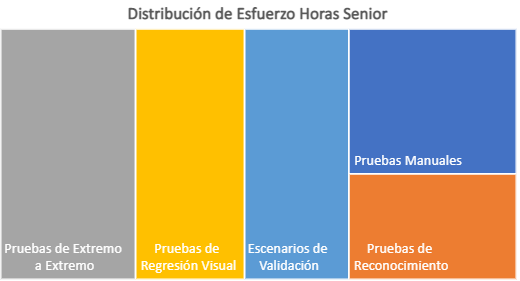
* + 1. **Recursos Económicos para la contratación de servicios/personal:**

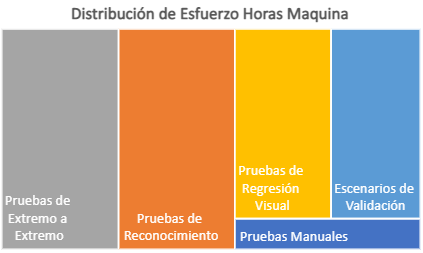
No contamos con recursos monetarios para contratar servicios externos.

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

|  |  |  |  |
| --- | --- | --- | --- |
| Nivel | Tipo | Técnica | Objetivo |
| Unidad | Funcional  Positiva | Limit Values | Validar el correcto desarrollo de las funcionalidades con respecto a los requerimientos establecidos por el usuario mediante la realización de pruebas manuales. |
| Unidad | Funcional  Negativa | Check List Based Testing | Evaluar el desempeño de las funcionalidades principales de la aplicación bajo escenarios de prueba negativos y positivos utilizando las siguientes estrategias para la generación del pool de datos: a priori, aleatorio dinámico y aleatorio. |
| Unidad | Funcional  Positiva/Negativa | Transición de Estado | Evaluar el desempeño de las funcionalidades principales de la aplicación bajo escenarios de prueba negativos y positivos utilizando las siguientes estrategias para la generación del pool de datos: a priori, aleatorio dinámico y aleatorio. |
| Aceptación | Funcional  Positiva | User Journey Test | Determinar el grado de cumplimiento de las funcionales de acuerdo con el punto de vista del usuario mediante la realización de UAT (User Acceptance Testing). (Validación Externa). |
| Sistema | No funcionales  Caja Negra | GUI Ripping | Agregar mayor cobertura de pruebas a través del uso de Rippers y Monkeys |
| Sistema | Funcionales  Caja Blanca | Smoke Testing | Generar estrategias de mejora continua que permitan reducir la probabilidad de ocurrencia de bugs en iteraciones futuras. |
| Sistema | Funcionales  Caja Negra | VRT | Validar que los cambios realizados a la aplicación desde el último release no afectan de manera negativa la apariencia visual de la Interfaz de usuario a través de la implementación de VRT. |
| Integración | Funcionales  Caja Blanca | Component testing | Implementar pruebas automatizadas E2E para garantizar que los flujos. |

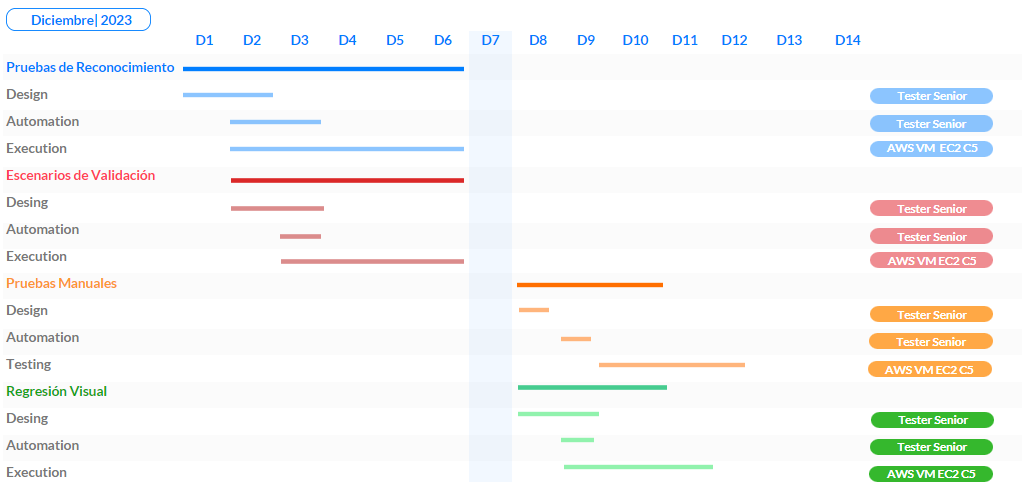
* 1. **Distribución de Esfuerzo**
     1. **Relación de recursos por prueba**



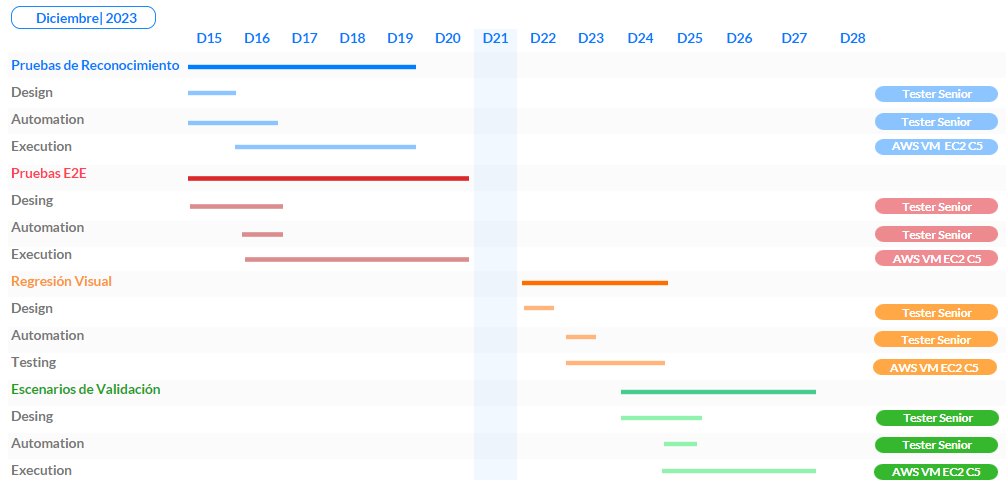


* + 1. **Diagramas de Gantt**

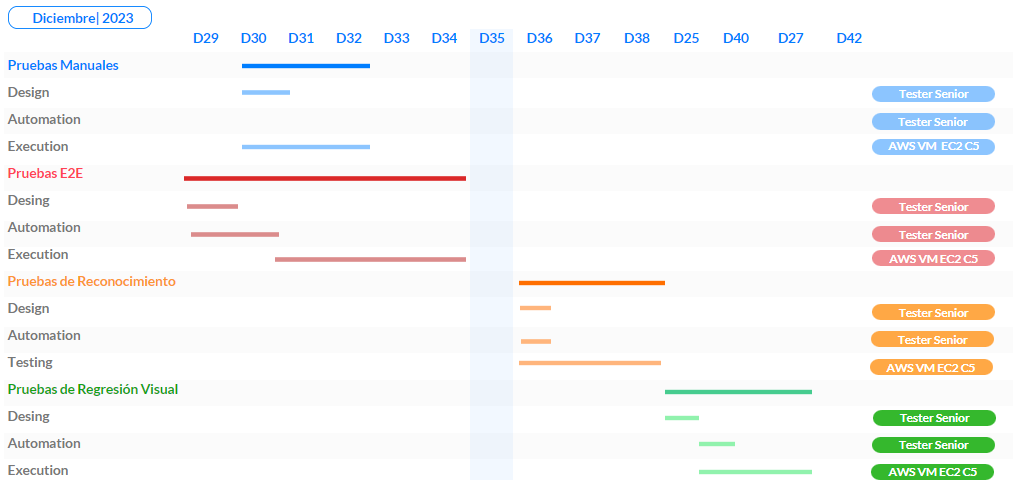
**Semana 1 y 2**



**Semana 3 y 4**



**Semana 5 y 6**



**Semana 7 y 8**



1. **Inventario Escenarios de Prueba** 
   1. **Pruebas Manuales**

Se realizan pruebas exploratorias para casos positivos y negativos para cada una de las distintas funcionalidades, por favor visitar el link a continuación para ver el detalle:

[Listado de pruebas manuales](https://uniandes-my.sharepoint.com/:x:/g/personal/h_patarroyo_uniandes_edu_co/EeEP0DUYvn9IgYN-DOR4U00BTM5znDIMGqITETB2cyfUnQ)

* 1. **Pruebas de Reconocimiento**

Para las pruebas de reconocimiento se utilizan las estrategias de monkeys y rippers, haciendo uso de las herramientas de cypress y puppeter para ejecutar cada escenario aplicado a cada una de las funcionalidades descritas en la identificacion de las funcionalidades.

Se pueden encontrar más detalles del código en el repositorio en las carpetas cypressMonkey y Ripuppet

* 1. **Pruebas de Extremo a Extremo**

Las pruebas de extremo a extremo se realizan de acuerdo a la experiencia que tiene el equipo y poder realizarlas de la manera más eficiente posible, se evaluan dos herramientas Kraken y Playwright y se incluyen sus estrategias a nivel de implementación en el repositorio del proyecto.

El inventario de pruebas realizado incluye diferentes flujos de principio a fin para las principales funcionalidades como crear un post o una página, entre otras, se detalla a continuación:

**Escenario 1:** Crear Miembro

**Escenario 2:** Crear Tag

**Escenario 3:** Crear página

**Escenario 4:** Crear Post

**Escenario 5:** Editar Miembro

**Escenario 6:** Editar Tag

**Escenario 7:** Editar página

**Escenario 8:** Editar Post

**Escenario 9:** Edición Perfil de Usuario

**Escenario 10:** Sign In

**Escenario 11:** Crear Integración

**Escenario 12:** Configuración Settings Post

**Escenario 13:** Configuración Settings Page

**Escenario 14:** Editar Integración

**Escenario 15:** Crear Post Programado

* 1. **Pruebas de Regresión Visual**

Para las pruebas de regresión visual se contempla la siguiente estrategia, realizar las pruebas con la herramienta backstop js a traves de la comparación de imágenes con respecto a la versión más reciente y la versión anterior y de esta forma garantizar que se pueda mantener una línea sobre la ejecución de las pruebas, las pruebas ejecutadas son las mismas que se ejecutaron en las pruebas E2E pero enfocadas a la toma de screenshots para realizar validaciones en tiempo real.

* 1. **Escenarios de Validación**

Para la validación de datos se generaron datos aleatorios usando tres estrategias: Generación de datos aleatorios, generación de datos pseudo aleatorios y generación de datos a priori, cada una de estas estrategias se realizó utilizando herramientas de generación de datos en línea (Mockaroo) y generación de datos mediante librerías (Faker). El siguiente es el listado de escenarios:

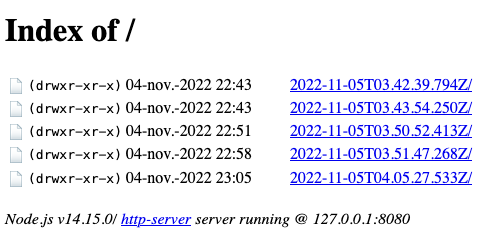
* Create Member
* Edit Member
* Edit Staff
* Create Integration
* Edit Integration
* Create Scheduled Post
* Edit Page Config
* Create Page
* Edit Page
* Delete Page
* Create Tag
* Edit Tag
* Delete Tag
* Create a new post
* Edit Post
* Delete a post
* Edit Config settings post
* Sign in Sign Out

1. **Resultados de la estrategia para la primera semana**
   1. **Ejecución Monkeys**

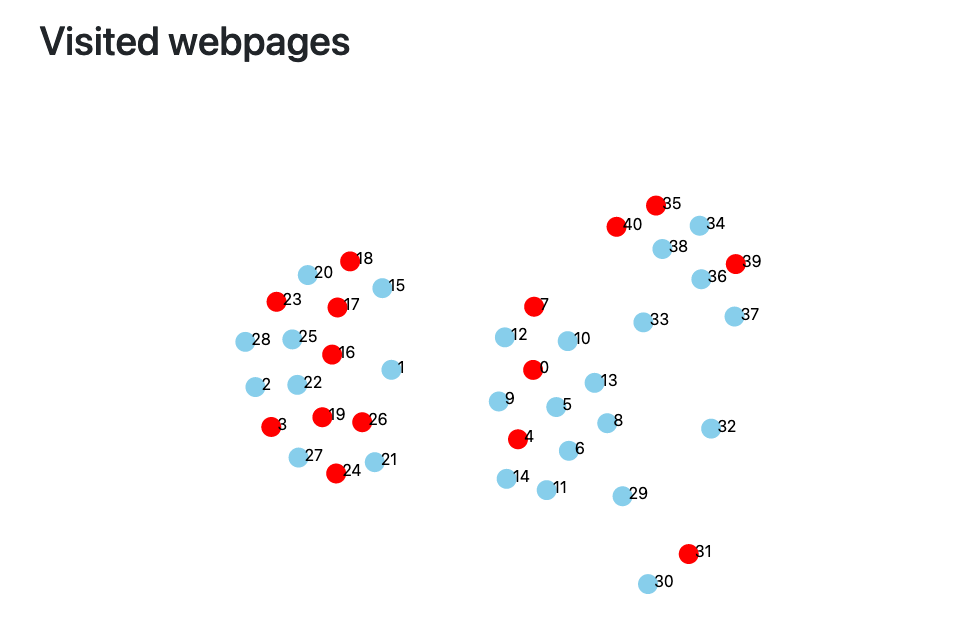
Se realizaron pruebas automatizadas sobre el software usando dos herramientas RIPuppet y Cypress Monkey

* 1. RIPuppet

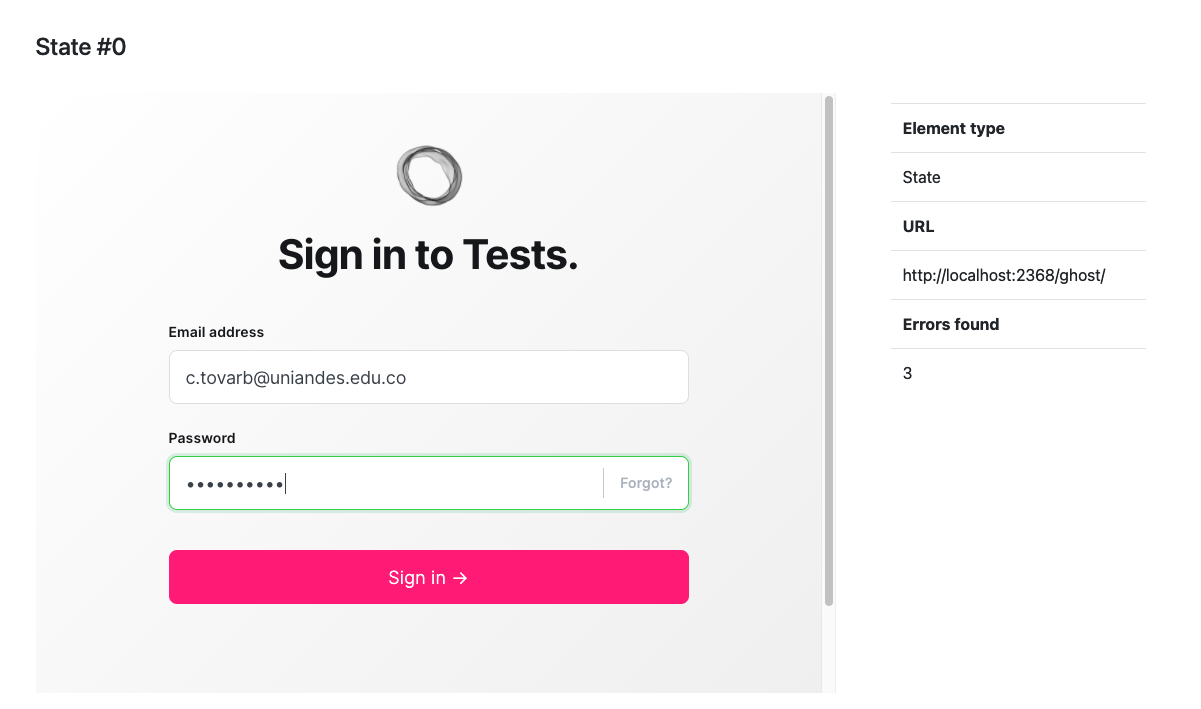
Al ejecutar las pruebas sobre el software se genera una carpeta de resultados por cada ejecución, estas carpetas contienen archivos tales como screenshots de las pruebas y archivos de formato json con los cuales se creará un reporte final, al consultar los resultados la interfaz nos muestra los links a las ejecuciones realizadas.



El siguiente es el reporte asociado a los resultados de la ejecución de las pruebas utilizando RIPuppet, el reporte genera un grafo indicando los states en las pruebas, en azul se marcan los states que terminaron de forma correcta y en rojo los que generaron algún tipo de error.



Al seleccionar uno de los nodos de la figura el reporte muestra el state asociado a ese nodo, un screenshot del estado de la página web en el momento así como la URL y el número de errores encontrados en ese state.

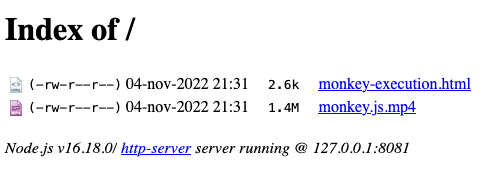


En la misma página, en la parte inferior se listan los errores asociados y por cada uno se muestra el error que generó la aplicación al intentar realizar la prueba.

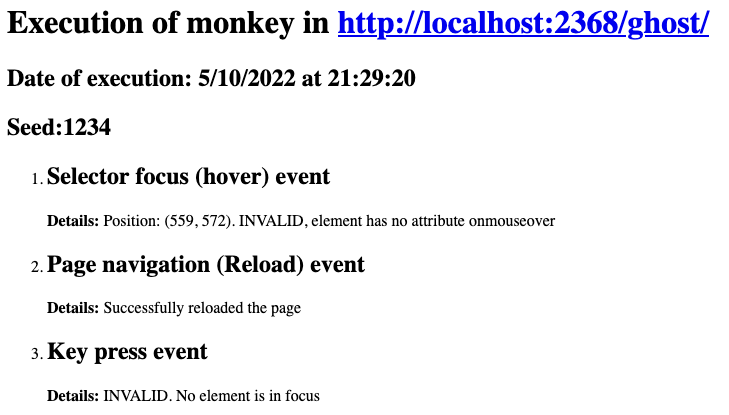


* 1. Cypress Monkey

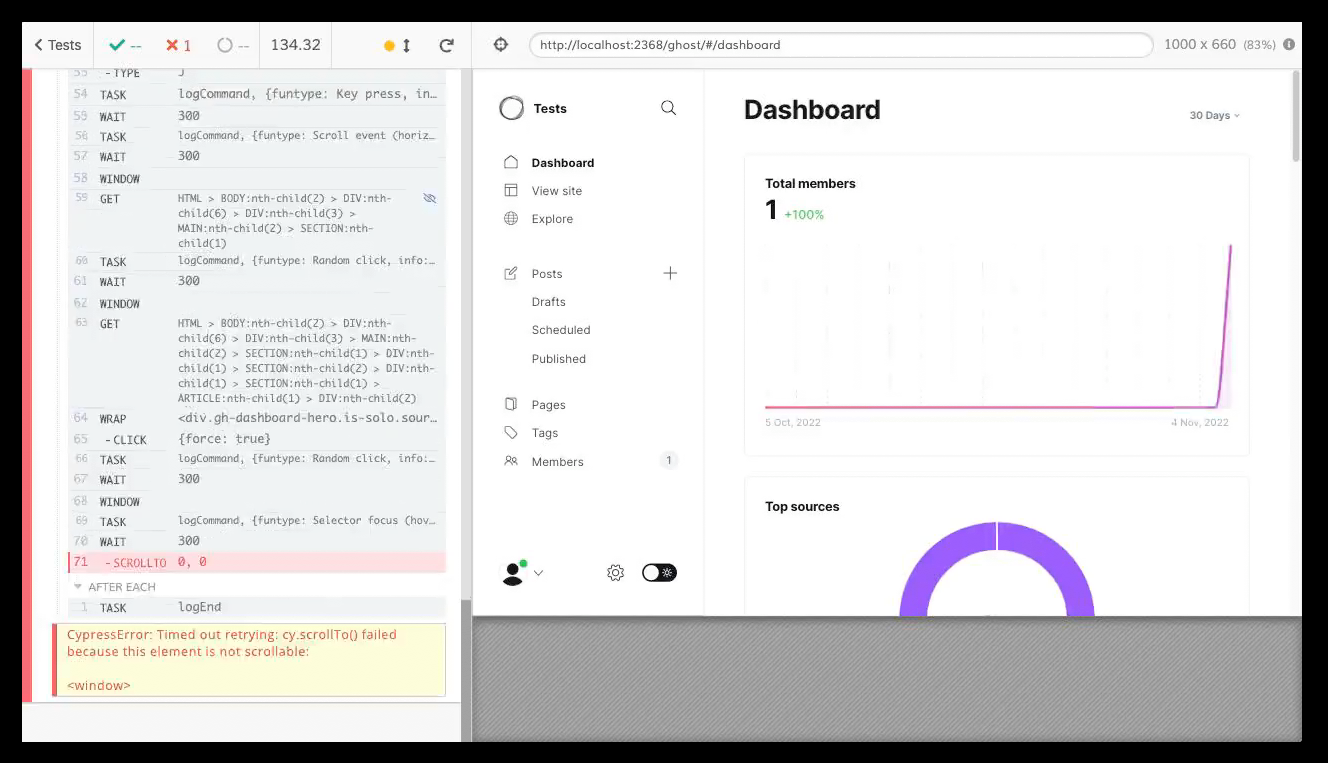
Al realizar las pruebas sobre el software se genera una carpeta con los resultados de la misma, cabe resaltar que cada vez que se ejecuta una nueva prueba los resultados anteriores son reemplazados por los de la nueva ejecución. Los archivos generados contienen un archivo html y un archivo mp4.



El archivo html contiene información de la ejecución tal como la página en la que se encuentra el software desplegado, la fecha y hora de la ejecución, la semilla y un listado de pruebas realizadas sobre el software y el resultado de la misma.



El archivo mp4 es un video de la ejecución de las pruebas en la parte izquierda se muestran las tareas realizadas en la ejecución de las pruebas y las respuestas de la aplicación en cada paso. Al generar un error este se muestra en el listado. En el lado derecho se muestra la página web y la interacción de la herramienta con la misma.

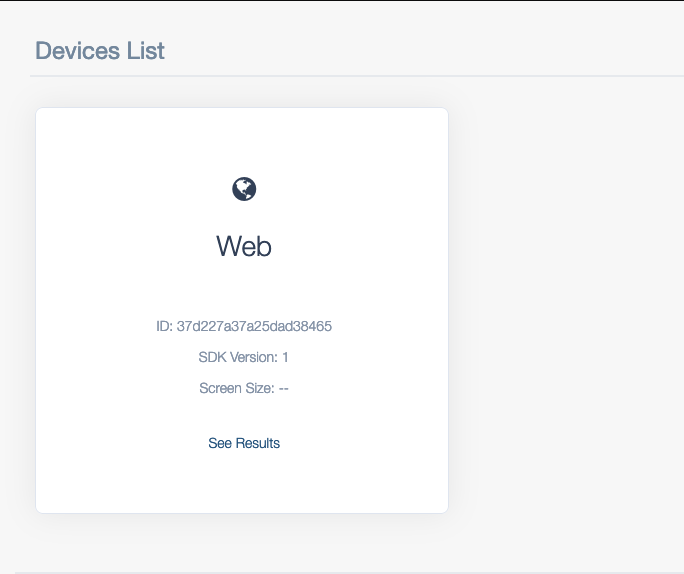


* 1. **Escenarios de Validación**

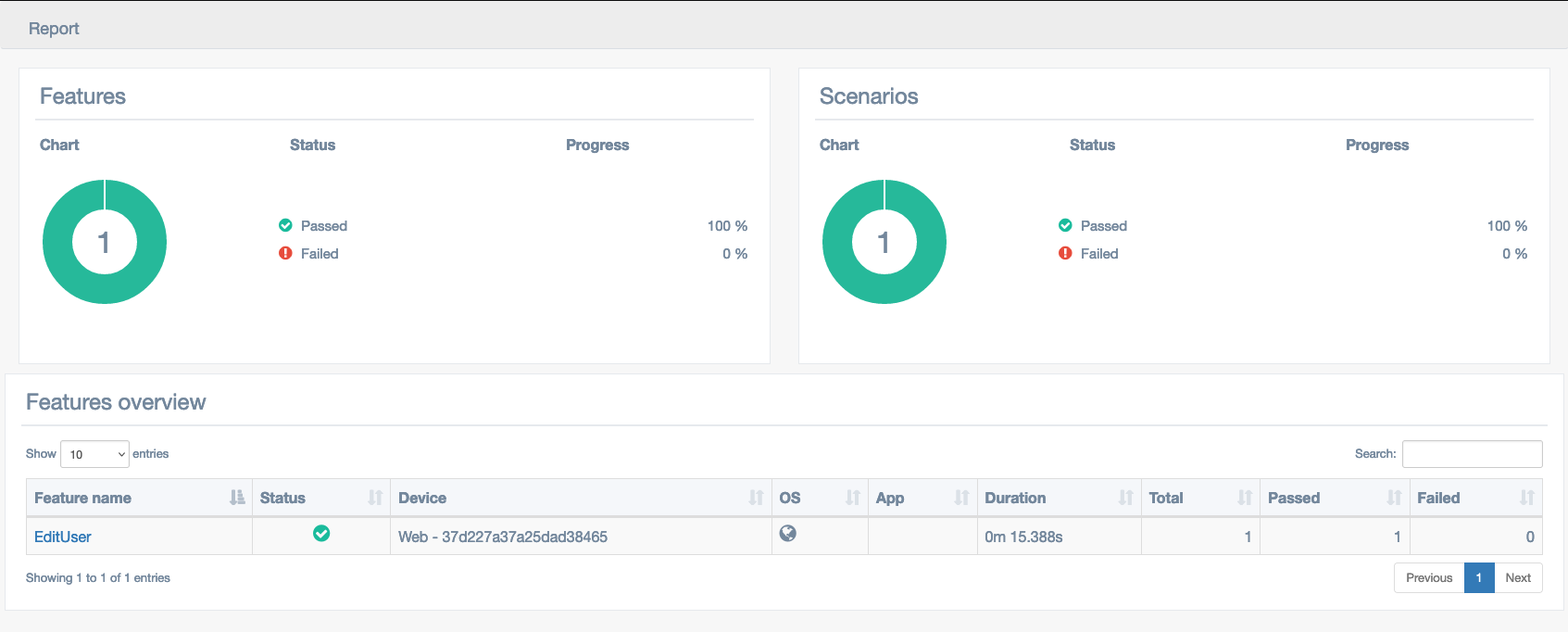
La ejecución de pruebas de los datos aleatorios se realizó usando las herramientas Playwright y Kraken.

Al ejecutar las pruebas de Kraken, el software genera una carpeta de resultados llamada reports y dentro de ella se genera una carpeta con un identificador único por cada prueba.

Al abrir el archivo html contenido en la carpeta se muestra el reporte de cada prueba en el que indica el dispositivo en el que se ejecutó.



Al hacer click en See Results nos muestra el detalle de la prueba con el número de scenarios por archivo y un gráfico que indica la pruebas que terminaron correctamente y las que fallaron.



Al seleccionar el feature, en este caso EditUser, se muestra el detalle de la prueba con cada uno de los pasos generados en el archivo y al final del reporte los screenshots tomados en la ejecución de los pasos.

