

**ESTRATEGIA DE PRUEBAS DE GHOST**

**Presupuesto Final**

Joan Manuel Blanco Vásquez

Roberto Pablo Carreño Zambrano

Camilo Ernesto Perafan Del Campo

Pedro Damián Guillen Garza

Versión 1.0

Semana 8

# Tabla de contenido

[***1.***](#_heading=h.gjdgxs) ***Tabla de contenido 2***

[***2.***](#_heading=h.30j0zll) ***APLICACIÓN BAJO PRUEBAS 3***

[*2.1.*](#_heading=h.1fob9te) *Nombre Aplicación 3*

[*2.2.*](#_heading=h.3znysh7) *Versión 3*

[*2.3.*](#_heading=h.2et92p0) *Descripción 3*

[*2.4.*](#_heading=h.tyjcwt) *Funcionalidades Core 3*

[*2.5.*](#_heading=h.3dy6vkm) *Diagrama de Arquitectura: 4*

[*2.6.*](#_heading=h.1t3h5sf) *Diagrama de Contexto 5*

[*2.7.*](#_heading=h.4d34og8) *Modelo de Datos 5*

[*2.8.*](#_heading=h.2s8eyo1) *Modelo de GUI 6*

[***3.***](#_heading=h.17dp8vu) ***CONTEXTO DE LA ESTRATEGIA DE PRUEBAS 7***

[*3.1.*](#_heading=h.3rdcrjn) *Objetivos 7*

[*3.2.*](#_heading=h.26in1rg) *Duración de la iteración de pruebas 8*

[*3.3.*](#_heading=h.lnxbz9) *Presupuesto de pruebas 8*

[3.3.1.](#_heading=h.35nkun2) Recursos Humanos 8

[3.3.2.](#_heading=h.1ksv4uv) Recursos Computacionales 8

[3.3.3.](#_heading=h.44sinio) Recursos Económicos para la contratación de servicios/personal 9

[*3.4.*](#_heading=h.2jxsxqh) *TNT (Técnicas, Niveles y Tipos) de pruebas 9*

[*3.5.*](#_heading=h.z337ya) *Distribución de Esfuerzo 9*

[***4.***](#_heading=h.3j2qqm3) ***ENLACE DEL VIDEO JUSTIFICACIÓN DEL PRESUPUESTO 10***

[***5.***](#_heading=h.1y810tw) ***ANALISIS HERRAMIENTAS DE PRUEBAS 10***

[*5.1.*](#_heading=h.4i7ojhp) *Monkey-Cypress: 10*

[*5.2.*](#_heading=h.2xcytpi) *Resultado esperado vs resultado real 10*

# APLICACIÓN BAJO PRUEBAS

## Nombre Aplicación

Ghost

## Versión

La versión de la aplicación bajo pruebas es 3.42.5.

* 1. **Descripción**

Ghost es un gestor de contenidos o CMS por sus siglas en ingles Content Manager System, basado en la plataforma NodeJS y que, según se define en su propio sitio, “*está estructurada como una aplicación web moderna y desacoplada con una arquitectura sensata basada en servicios”.* Es de código abierto y de publicación on-line, ideal para la creación y gestión de blogs. Su primera versión fue lanzada en el año 2013 y como parte de sus características destacables está la posibilidad de uso de HTML puro, así como la sintaxis Markdown a cambio del tradicional editor WYSIWYG (What You See Is What You Get).

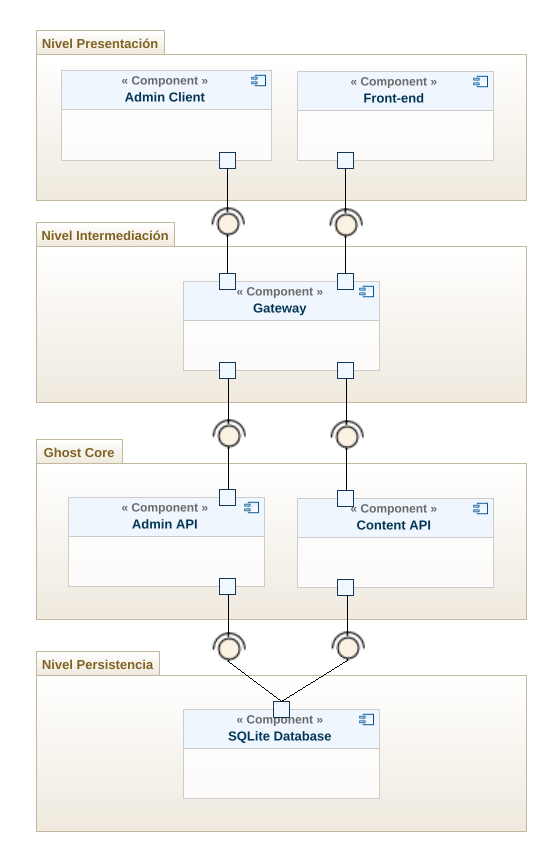
* 1. **Funcionalidades Core**

Las siguiente son las funcionalidades clave a probar:

|  |  |
| --- | --- |
| NOMBRE | DESCRIPCIÓN |
| Funcionalidad MIEMBROS | Tiene la opción de agregar, editar, eliminar miembros para la conformación de un equipo de trabajo. |
| Funcionalidad PAGES | Tiene la opción de agregar, editar, eliminar paginas dentro del CMS. |
| Funcionalidad  TAGS | Tiene la opción de agregar, editar, eliminar tags dentro del CMS. |
| Funcionalidad  POST | Tiene la opción de agregar, editar, eliminar posts dentro del CMS. |
| Funcionalidad  DASHBOARD | Se puede navegar dentro de las funcionalidades principales del CMS. |
| Funcionalidad  LOGIN | Hacer Log in a la aplicación de Ghost. |

* 1. **Diagrama de Arquitectura:**

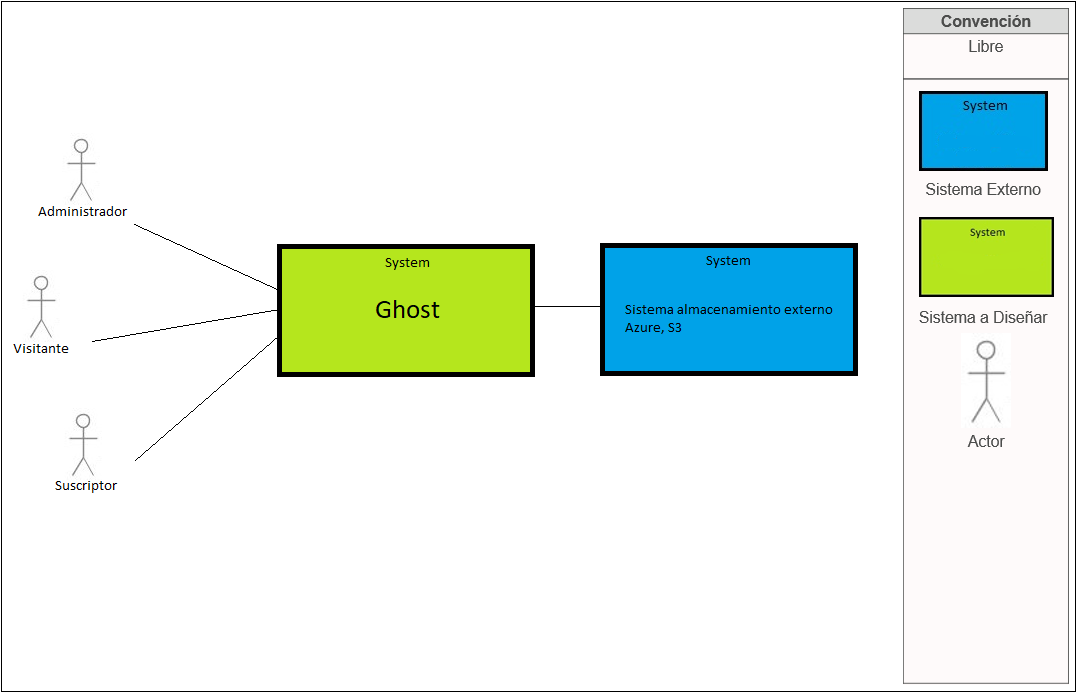
En el diagrama de arquitectura se evidencia la separación de las interfaces de administración y front end así como en la capa de lógica de negocio el manejo independiente de estas dos interfaces por medio de API’s diferentes conectadas a la misma base de datos embebida.



*En el siguiente* [*enlace*](https://github.com/jblancov7764/aplicacion-ghost/blob/main/component-diagram.png) *se puede apreciar mejor el diagrama.*

* 1. **Diagrama de Contexto**

El diagrama de contexto a continuación Muestra los posibles actores que interactúan con la aplicación Ghost como lo son el administrador de la página, los visitantes y los usuarios que se encuentran suscritos a la página, adicional se muestran los posibles sistemas externos con los que interactúa la aplicación, como sistemas de almacenamiento externos en la nube.



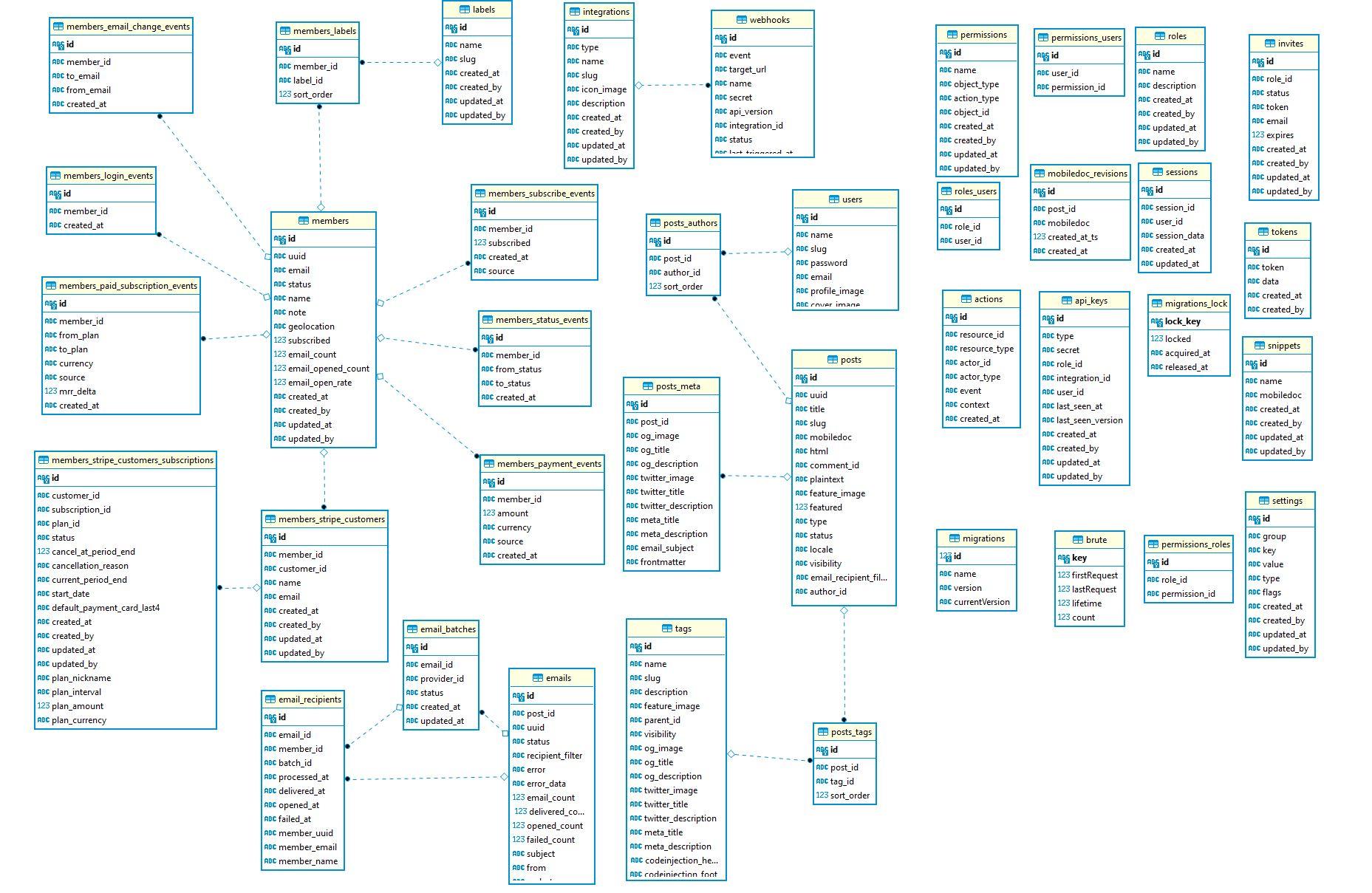
*En el siguiente* [*enlace*](https://github.com/jblancov7764/aplicacion-ghost/blob/main/diagrama-contexto.png) *se puede apreciar mejor el diagrama.*

* 1. **Modelo de Datos**

El modelo a continuación relaciona las distintas clases de las que se compone el sistema Ghost.

Como principales clases, se evidencian la clase Post que se relaciona con las clases PostAuthors, PostTags y PostMeta. Por otro lado, la clase Members que se relaciona con 8 clases más, de ellas las más destacadas, las clases MemebersLabels, MembersEmail, MemebersPaidSubscription y MemebersPayment. Finalmente, la clase Roles y la clase Permissions que se relaciona con la clase PermissionRoles, PermissionsUsers.

Estas clases serán clave en el desarrollo del plan de pruebas sobre las cuales se derivan las Funcionalidades Core mencionadas en el numeral 2.4.



*En el siguiente* [*enlace*](https://raw.githubusercontent.com/robertopabloc/images/main/Screenshots/Modelo%20de%20dominio%20Ghost.JPG) *se puede apreciar mejor el diagrama.*

* 1. **Modelo de GUI**

En el siguiente modelo se puede apreciar el flujo de comunicación entre las distintas vistas del aplicativo. Para este caso, se relacionan las interfaces graficas tanto del front-end como del panel de configuración del sistema Ghost.

Se resalta el flujo de comunicación entre el main del panel de configuración y las funcionalidades administración de páginas, administración de posts y administración de miembros, entre otros, que serán objeto del presente plan de pruebas.



*En el siguiente* [*enlace*](https://raw.githubusercontent.com/robertopabloc/images/main/Screenshots/Modelo%20GUI_2.png) *se puede apreciar mejor el diagrama.*

# CONTEXTO DE LA ESTRATEGIA DE PRUEBAS

## Objetivos

Se cuenta con el siguiente presupuesto: cuatro testers senior, durante 8 semanas, en una dedicación de 8 horas por persona semanalmente, 200 horas/máquina en Amazon AWS.

Como objetivo general, con esta estrategia de pruebas se busca diseñar e implementar los casos, como se realizarán y los componentes que abarcarán. Se pretende reducir el impacto en errores en producción, desarrollando diferentes pruebas como son: **pruebas manuales, pruebas de reconocimiento, pruebas de extremo a extremo, pruebas de regresión visual y escenarios de validación de datos**, documentando los resultados y asegurando que los hallazgos con el trabajo realizado sean reparados por el equipo de desarrollo para ser nuevamente probados.

A continuación, se detallan los objetivos específicos que apuntarán al logro del objetivo general de la estrategia de pruebas.

* Desarrollar y Ejecutar Pruebas manuales: por medio de pruebas exploratorias manuales de tipo funcional se identificarán problemas en la funcionalidad básica de la aplicación.
* Desarrollar y Ejecutar Pruebas de reconocimiento: por medio de la implementación de pruebas de reconocimiento se implementará el uso de Monkeys y Rippers para aumentar la cobertura de entradas y salidas sin sesgos.
* Desarrollar y Ejecutar Pruebas de extremo a extremo: por medio de la implementación de la herramienta Cypress se llevarán a cabo pruebas E2E para validar la funcionalidad de la aplicación.
* Desarrollar y Ejecutar Pruebas de regresión visual: para la implementación de VRT se desarrollarán las pruebas en la herramienta ResembleJs, con el objetivo para poder comparar el estado actual y pasado de la aplicación con el objetivo de asegurar que el estado actual es mejor o igual que el anterior.
* Desarrollar y Ejecutar Pruebas de validación de Datos: para las pruebas de validación de datos se implementarán scripts por medio de las siguientes estrategias: a priori – pseudo aleatorios - aleatorios con las siguientes herramientas: Faker y Mockaroo.
  1. **Pre-requisitos:**

Para la ejecución de las pruebas de Integración y el trabajo que se va a realizar por las personas dentro del presupuesto se deben tener en cuenta los siguientes pre-requisitos:

* Que el código cuente con pruebas unitarias
* Las pruebas Unitarias se ejecuten en el pipeline de desarrollo y deben de haber pasado en su totalidad.
* Para poder ejecutar el pipeline de las pruebas de integración los dos puntos anteriores se deben cumplir.

## Duración de la iteración de pruebas

|  |  |
| --- | --- |
| **Iteración – Descripción** | **Semana** |
| 1 – Pruebas manuales | Semana 1 |
| 2 – Pruebas de reconocimiento / MONKEYS | Semana 2 |
| 3 – Pruebas de reconocimiento / Rippers | Semana 3 |
| 4 – Pruebas de extremo a extremo | Semana 4 y 5 |
| 5 – Pruebas de regresión visual | Semana 6 |
| 6 – Escenarios de validación de datos | Semana 7 y 8 |

## Presupuesto de pruebas

### Recursos Humanos

|  |  |  |
| --- | --- | --- |
| **Cantidad** | **Rol** | **Experiencia** |
| 1 | Ingeniero pruebas automatizador para pruebas en ambiente AWS y local. | Entre 1 y 3 años |
| 1 | Ingeniero pruebas automatizador para pruebas en ambiente AWS y local. | Entre 1 y 3 años |
| 1 | Ingeniero pruebas automatizador para pruebas en ambiente AWS y local. | Entre 1 y 3 años |
| 1 | Ingeniero pruebas automatizador para pruebas en ambiente AWS y local. | Entre 1 y 3 años |
| **4** | **TOTAL** | |

### Recursos Computacionales

Para el trabajo se requerirán 3 equipos personales y equipos móviles con las siguientes características:

|  |  |
| --- | --- |
| **Requisitos** | **Descripción** |
| Procesador | Intel Core i5 o superior |
| RAM | 8 GB o superior |
| Espacio DD | 10 Gb o superior |
| Sistema Operativo | Windows 7 o superior, MacOS 10.12+, Ubuntu LTS releases 18.04 o superior |
| Browser | Chrome v77+, Firefox v68+, Safari v12+, Edge v44+ |
| Móvil | iOS: iPhone 5s o superior con iOS 11+  Android: Dispositivo con Android 7+ |

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

No se contará con contratación de recurso externo por outsourcing.

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Nivel** | **Tipo** | **Técnica** | **Objetivo** |
| Unidad | Caja blanca - Pruebas unitarias | Ejecución automática definidas previamente por el equipo, en 2 iteraciones de pruebas con validación de que son ejecutadas y pasan exitosamente. | Detectar fallas en tiempo de ejecución en el código por medio de la ejecución de pruebas unitarias. |
| Integración | Caja negra - Prueba de integración | Ejecución y eventos generados de forma automática. Desarrollar una prueba de integración con duración de 2 semanas. Herramienta Monkey-Cypress  Herramienta Rippers-Cypress | Detectar problemas de integración por medio de pruebas de integración automatizadas. |
| Sistema | Positivas / Negativas - Pruebas exploratorias | Ejecución y eventos generados de forma automática. Desarrollar pruebas exploratorias manuales, pruebas de extremo a extremo, Escenarios de validación de datos con las siguientes Herramienta Cypress/ Page Object y Mockaroo. Con duración de 4 semanas | Detectar la mayor cantidad posible de fallas en la UI del aplicativo por medio de pruebas automatizadas de GUI. |
| Aceptación | Funcionales y no funcionales | Ejecución y eventos definidos por un humano. Desarrollar pruebas de regresión visual y generación de datos aleatorios. Usando las herramientas ResembleJs y Faker. Con una duración de 2 semanas. | Detectar errores en la interfaz de usuario por medio de pruebas exploratorias. |

## Distribución de Esfuerzo

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Iteración – Descripción** | **RRHH** | **Hrs/Hombre** | **Total hrs Desarrollo**  **Pruebas** | **Total, hrs Ejecución**  **Manual** | **Total, hrs Ejecución Automatizadas** |
| 1 – Pruebas manuales | 4 | 8 | 0 | 32 | 0 |
| 2 – Pruebas de reconocimiento / MONKEYS | 4 | 8 | 4 | 0 | 28 |
| 3 – Pruebas de reconocimiento / Rippers | 4 | 8 | 4 | 0 | 28 |
| 4 – Pruebas de extremo a extremo | 4 | 8 | 20 | 0 | 44 |
| 5 – Pruebas de regresión visual | 4 | 8 | 8 | 0 | 24 |
| 6 – Escenarios de validación de datos | 4 | 8 | 8 | 0 | 56 |
| **TOTALES: 256 hrs** | | | **44** | **32** | **180** |

# ENLACE DEL VIDEO JUSTIFICACIÓN DEL PRESUPUESTO

A continuación, se relaciona el enlace del video de justificación de los presupuestos y su relación para cada una de las pruebas propuestas dentro de este documento:

1 – Pruebas manuales

2 – Pruebas de reconocimiento / MONKEYS

3 – Pruebas de reconocimiento / Rippers

4 – Pruebas de extremo a extremo

5 – Pruebas de regresión visual

6 – Escenarios de validación de datos

# ANALISIS HERRAMIENTAS DE PRUEBAS

### [Pros y Contras de las herramientas Cypress/Kraken](https://github.com/pruebas-miso/ghost-cypress-e2e/wiki/ProsContras)

### [Pros y Contras de las herramientas - Resemble y Backstop](https://github.com/pruebas-miso/ghost-cypress-e2e/wiki/ProsContras1)

# CODIGO PARA LA EJECUCIÓN DE DIFERENTES TIPOS DE PRUEBAS

|  |  |
| --- | --- |
| **Tipo de Prueba** | **Semana** |
| 1 – Pruebas manuales | Excel |
| 2 – Pruebas de reconocimiento / MONKEYS | [LINK](https://github.com/pruebas-miso/monkeys) |
| 3 – Pruebas de reconocimiento / Rippers | [LINK](https://github.com/pruebas-miso/rippers) |
| 4 – Pruebas de extremo a extremo | [LINK](https://github.com/pruebas-miso/ghost-cypress-e2e/tree/v3.42.5) |
| 5 – Pruebas de regresión visual | [LINK](https://github.com/pruebas-miso/vrt) |
| 6 – Escenarios de validación de datos | [LINK](https://github.com/pruebas-miso/ghost-cypress-e2e/tree/v3.42.5) |

# SISTEMA DE REGISTRO DE INCIDENCIAS

<https://github.com/pruebas-miso/vrt/issues>