Estrategia de Pruebas

Versión final

Equipo # 3

Laura Alejandra Carrillo Guzmán

Sandra Victoria Hurtado Gil

Leidy Viviana Osorio Jiménez

Tim Ulf Pambor

1. Aplicación Bajo Pruebas

1.1. Nombre Aplicación: Ghost

1.2. Versión: 3.41.1

Nota: se considera la versión 4.44.0 en desarrollo y, por lo tanto, solo se evaluará su comportamiento en cuanto a regresión visual.

1.3. Descripción:

"Ghost es un sistema manejador de contenidos, también conocido como CMS, cuya funcionalidad consiste en centralizar, por medio de una plataforma, la creación y gestión de contenido para sitios web. La funcionalidad de Ghost es bastante cercana a la de otras herramientas como WordPress o Joomla, que son un poco más conocidas y utilizadas para la creación de blogs" (Espitia Acero, 2020).

Ghost es una aplicación Open-Source construida con Node.js, y cuyo código fuente puede encontrarse en GitHub: Ghost · GitHub, bajo licencia MIT. Fue creado en el año 2013 por John O'Nolan y Hannah Wolfe, quienes buscaban ofrecer una alternativa para las personas que deseaban publicar contenido en Internet en un sito propio. Actualmente, de acuerdo con la documentación oficial de Ghost, es uno de los CMS Open Source más populares del mundo (Ghost Foundation, 2023).

1.4. Funcionalidades Core:

Las principales funcionalidades de esta aplicación son:

1. **Módulo Posts**:
    a. Crear un Post: Permite crear un nuevo post (una entrada o publicación con título y texto) en el sitio web, puede ser para publicar inmediatamente o para publicar después, en una fecha y hora programadas. También es posible crear un post "borrador", es decir, que todavía no está publicado ni programado, para continuar en otro momento.
    b. Editar un Post: Permite modificar la información de un post existente, como el título, el texto, los tags, entre otros. También incluye la posibilidad de quitar una publicación de un post (es decir, quedaría en el listado para su gestión, pero no aparecería en el sitio web), o publicar uno que estaba en borrador.
    c. Borrar un Post: Permite eliminar un post, de manera que ya no aparezca en la lista de post ni en la página inicial del sitio web.

d. Consultar Posts: Permite ver la lista de post registrados, con su título, autor, tiempo desde la última modificación y estado. Se pueden ordenar por fecha de creación o de actualización.
e. Consultar Post con filtros: Permite ver la lista de post registrados, con su título, autor, tiempo desde la última modificación, pero solo los que cumplan con los filtros seleccionados: por estado (Draft, Published, Scheduled o Featured), por un autor o por una etiqueta/tag. Puede ser un solo filtro o la combinación de varios.

2. **Módulo Tags**:
   a. Crear tag: Permite crear una nueva etiqueta para clasificar los post o páginas. Las etiquetas, además de su nombre y su alias ("slug"), pueden tener una descripción y una imagen asociada.
   b. Editar tag: Permite modificar la información de una etiqueta existente, como su nombre, alias, descripción o imagen.
   c. Borrar tag: Permite eliminar un tag, de manera que ya no puede ser usado para clasificar posts o páginas.
   d. Consultar tags: Permite consultar las etiquetas que se han creado hasta el momento, mostrando un listado con el nombre, alias y cantidad de post que tiene asociados.
3. **Módulo usuarios (Staff)**:
   a. Crear cuenta de administrador por primera vez: Cuando se acaba de instalar Ghost, permite crear una cuenta de administrador para poder tener acceso a las diferentes funciones
   b. Invitar/crear usuario: Permite crear un usuario mediante una invitación a su correo electrónico. El usuario puede tener diferentes permisos: Administrator, Editor, Author, Contributor.
   c. Editar usuario: Permite actualizar la información de un usuario existente, como su imagen, alias, rol, sitio web, descripción, entre otros.
   d. Ingresar ("sign in"): Permite que un usuario registrado (incluyendo el administrador) tenga acceso a las funciones del sistema. Para ingresar debe escribir su correo, que lo identifica, y una contraseña. También se tiene la opción de recuperar la contraseña si se ha olvidado.
   e. Cerrar sesión ("sign out"): Permite salir de la aplicación y terminar la sesión iniciada al hacer el ingreso. Después de cerrar la sesión no es posible usar las funciones del sistema (hasta no ingresar de nuevo).
   f. Cambiar contraseña de usuario: Permite a un administrador cambiar la contraseña de un usuario existente.
4. **Módulo configuración sitio web**:
   a. Visualizar sitio web: Permite tener una visión preliminar del sitio web (para saber cómo quedaría para los usuarios finales), con las diferentes páginas y posts que se publiquen.
   b. Editar el diseño del sitio web (Navegación): Permite definir el tema principal del sitio, al igual que los enlaces que aparecerán en la página inicial.
   c. Editar la configuración (settings) del sitio web.
5. **Módulo Páginas**:
   a. Crear una Página: Permite crear una nueva página para el sitio web (con nombre, texto y posible contenido multimedia), que tendrá un enlace para su consulta desde la página inicial.
   b. Editar una Página: Permite modificar la información de una página existente, como el título, el texto, los tags, entre otros.
   c. Borrar una Página: Permite eliminar una página, de manera que ya no aparezca en la lista de páginas ni en el sitio web.
   d. Consultar Páginas: Permite ver la lista de páginas creadas, con su título, autor, tiempo desde la última modificación y estado. Se pueden ordenar por fecha de creación o de actualización.
   e. Consultar Páginas con filtros: Permite ver la lista de páginas del sitio web, con su título, autor, tiempo desde la última modificación, pero solo los que cumplan con los filtros seleccionados: por estado (Draf, Published, Scheduled o Featured), por un autor o por una etiqueta/tag. Puede ser un solo filtro o la combinación de varios.

1.5. Diagrama de Arquitectura:

Vista general:

Ghost está estructurado como una aplicación web moderna y desacoplada con una arquitectura sensible basada en servicios.

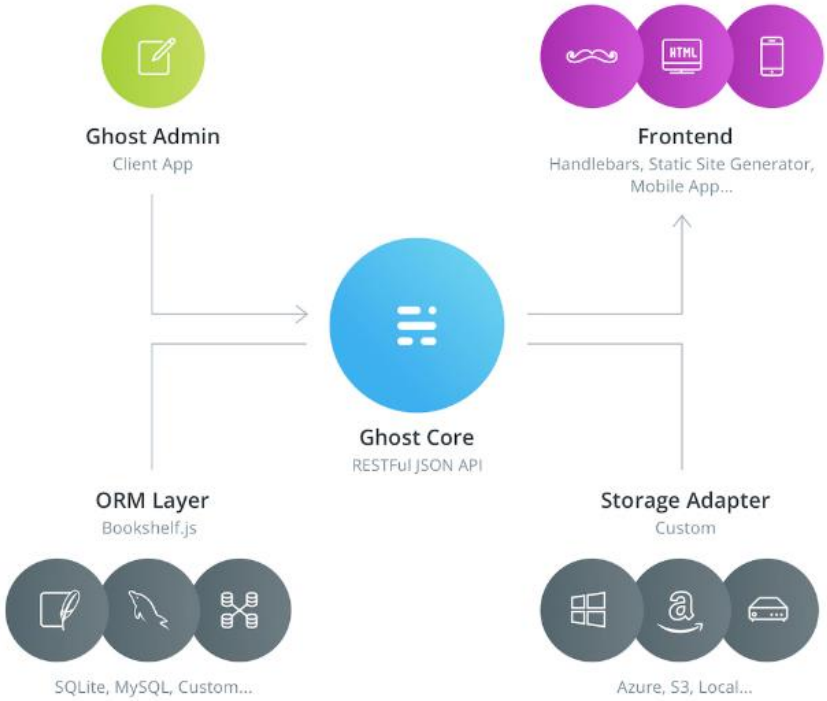


Imagen tomada de: https://ghost.org/docs/architecture/

Almacenamiento: SQLite3 es el predeterminado compatible en desarrollo, mientras que MySQL se recomienda para producción.

Frontend: Ghost es un CMS completamente autónomo que es completamente independiente de cualquier interfaz de usuario en particular o marco de sitio estático. Al igual que el cliente de administración de Ghost, su interfaz es opcional e intercambiable. Si bien la arquitectura inicial de Ghost representaba más una aplicación web monolítica estándar, ahora es compatible con casi cualquier interfaz que pueda lanzarle.

Storage Adapter: Es posible utilizar adaptadores de almacenamiento personalizados para que su sistema de archivos sea completamente externo. Hay una gama bastante amplia de adaptadores de almacenamiento prefabricados para Ghost que ya están disponibles para su uso.

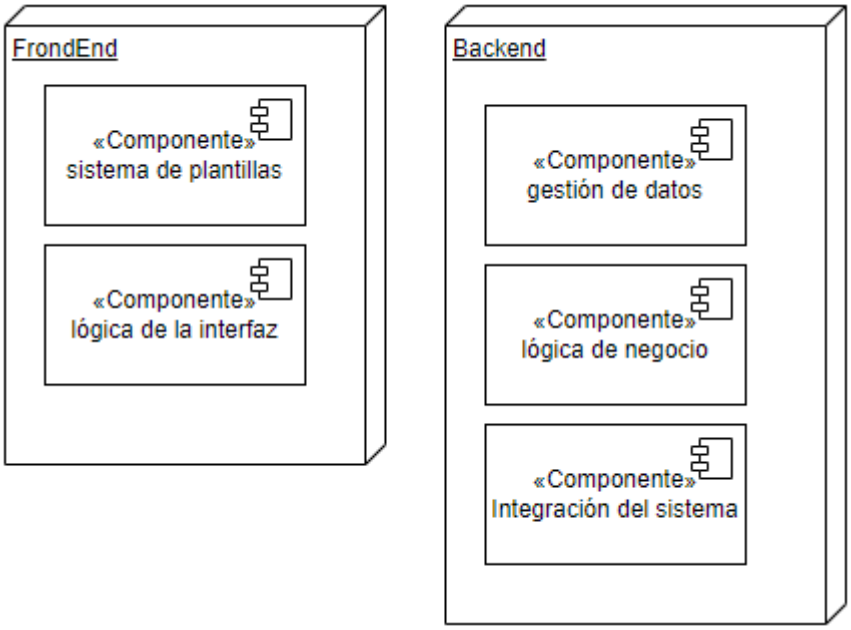
Ghost Core: La API de Ghost se divide por función en dos partes: contenido y administración. Cada uno tiene sus propios métodos de autenticación, estructura y amplias herramientas para que los casos de uso comunes de publicación se resuelvan con el mínimo esfuerzo.

La API de contenido público de Ghost es lo que entrega el contenido publicado al mundo y cualquier cliente puede acceder a él en modo de solo lectura para representarlo en un sitio web, una aplicación u otro medio incrustado.

Ghost Admin: Una interfaz de administración optimizada del lado del cliente para editores que necesitan una herramienta poderosa para administrar su contenido.

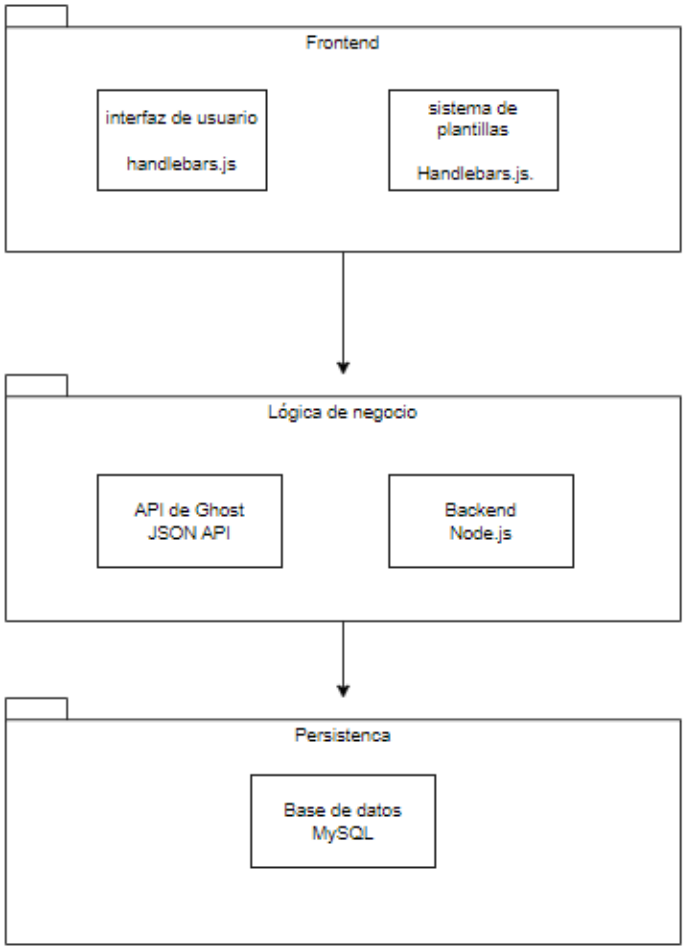
Vista de implementación:

Ghost está dividido en dos grandes componentes: el Frontend y el Backend. El frontend es el encargado de la presentación de la información al usuario, mientras que el backend se encarga de la gestión de datos y la lógica de negocio, como se presenta en el siguiente diagrama.



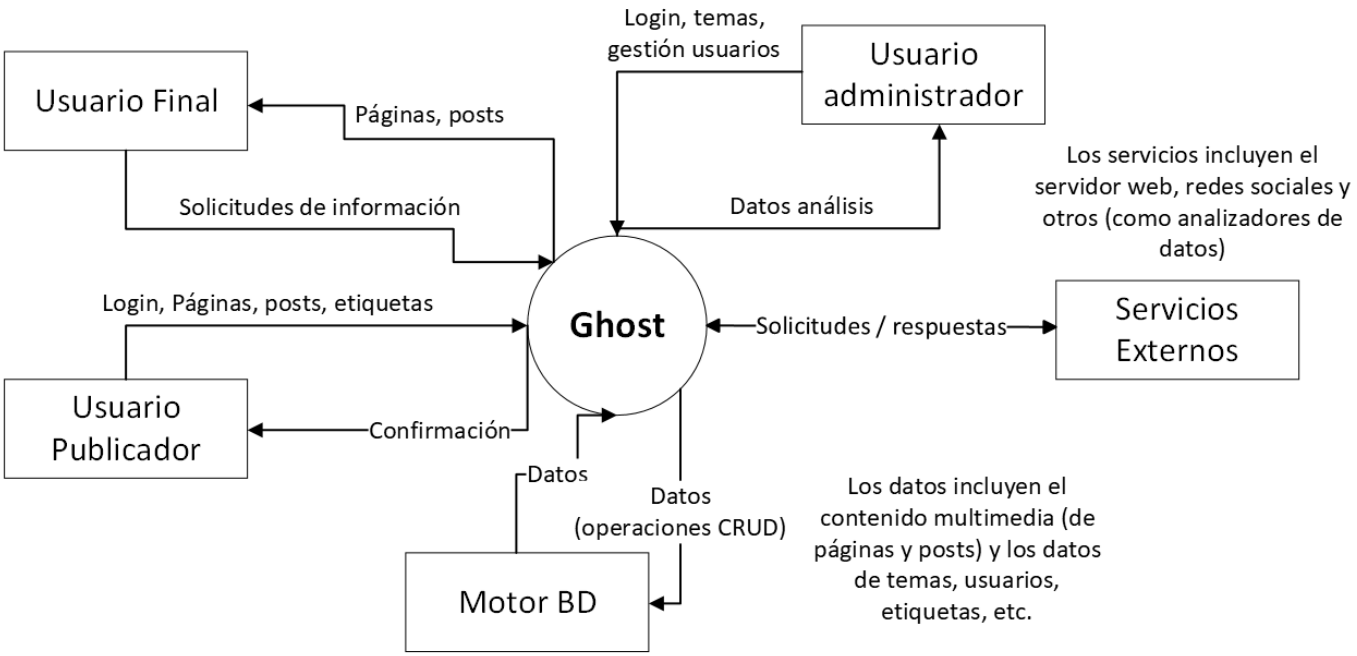
Vista lógica:

La arquitectura de Ghost se basa en un patrón de diseño Modelo-Vista-Controlador (MVC).

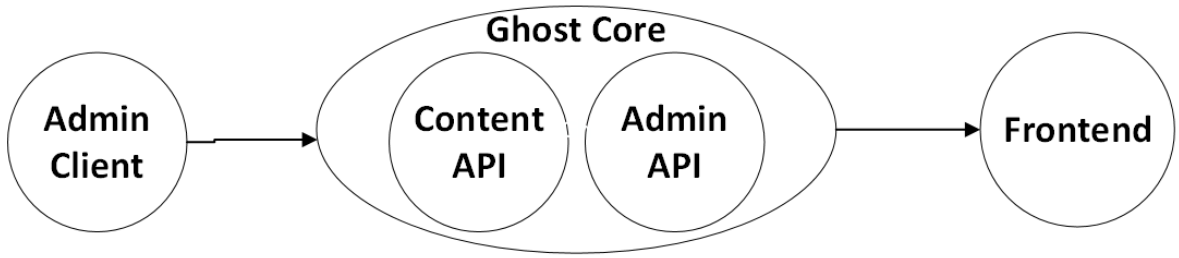


- Frontend construido en Ember.js: se encarga de la presentación de la información al usuario. Sistema de plantillas Handlebars.js: se utiliza para construir la interfaz de usuario y los temas personalizados.
- Lógica de negocio: Backend construido en Node.js: se encarga de la gestión de datos y la lógica de negocio. API potente construida usando JSON API: permite a los desarrolladores integrar fácilmente otras aplicaciones con la plataforma Ghost.
- Persistencia: Base de datos MySQL: almacena los datos del sistema.

1.6. Diagrama de Contexto:



Los principales componentes de Ghost, que pueden verse también como entidades que se relacionan con el Core:



1.7. Modelo de Datos:

El modelo de datos se puede encontrar en el repositorio, carpeta Estrategia, archivo **modelo-datos.pdf.** Enlace: https://github.com/tpambor/MISW4103-Final/blob/main/Estrategia/modelo-datos.pdf

1.8. Modelo de GUI:

El modelo de GUI se puede encontrar en el repositorio, carpeta Estrategia, archivo **modelo-gui.pdf**. Enlace: https://github.com/tpambor/MISW4103-Final/blob/main/Estrategia/modelo-gui.pdf

Una versión con mayor calidad se puede encontrar en https://miro.com/app/board/uXjVMZ7hOz0=/.

2. Contexto de la estrategia de pruebas

2.1. Objetivos:

Esta estrategia tiene en cuenta la disponibilidad de varios ingenieros automatizadores durante ocho semanas, al igual que la posibilidad de usar tiempo de máquina y algunos recursos adicionales (como API para generación de datos), de manera que se puedan realizar varios tipos de pruebas. Los objetivos son:

- **OBJ-1**: Mejorar la calidad de la aplicación y la experiencia de usuario al identificar y corregir errores de funcionalidad y usabilidad en los diferentes módulos de Ghost.
- **OBJ-2**: Disminuir riesgos de bloqueo de la aplicación por acciones de los usuarios, mediante la verificación de validaciones de datos (con datos generados a-priori y aleatoriamente) y respuesta a eventos aleatorios.
- **OBJ-3**: Garantizar la correcta visualización de la aplicación ("cross-browser visual testing") en los navegadores más usados actualmente: Chrome, Safari, Edge y Firefox, cada uno en la última versión y las tres anteriores (StatCounter, 2023).
- **OBJ-4**: Contribuir a la liberación más rápida y con mejor calidad (sin errores de regresión) de nuevas versiones de Ghost mediante el desarrollo de un conjunto de pruebas automatizadas.

2.2. Duración de la iteración de pruebas:

La estrategia está contemplada para 8 semanas, considerando 32 horas semanales dedicadas al proyecto por parte de los ingenieros automatizadores. Las fechas son:

- Semana 1: Del 23 al 27 de mayo
- Semana 2: Del 29 de mayo al 2 de junio
- Semana 3: Del 5 al 9 de junio
- Semana 4: Del 12 al 16 de junio
- Semana 5: Del 19 al 23 de junio
- Semana 6: Del 26 al 30 de junio
- Semana 7: Del 3 al 7 de julio
- Semana 8: Del 10 al 14 de julio

2.3. Presupuesto de pruebas:

2.3.1. Recursos Humanos

Para la ejecución de la estrategia de pruebas se contará con la participación de las siguientes personas:

Tipo de recurso	Cantidad	Horas dedicadas (semanal por persona)	Experiencia
Ingeniero automatizador senior	4	8	• Dos años de experiencia en el sector • Experiencia con pruebas exploratorios manuales

			• Experiencia con herramientas de automatización de pruebas:
			  ○ Cypress
			  ○ Kraken
			• Conocimiento de la aplicación y del negocio

Con base en un análisis de ofertas de empleo en el sector con un perfil adecuado para esta estrategia de pruebas y un análisis externo de talent.com, se calcula con un salario mensual de 4,500,000 COP (~1000 USD), el salario medio de las ofertas analizadas. Ofertas analizadas:

- https://www.elempleo.com/co/ofertas-trabajo/automatizador-medellin/1885701653
- https://www.elempleo.com/co/ofertas-trabajo/ingeniero-a-de-pruebas-162627789717/1885680768
- https://www.elempleo.com/co/ofertas-trabajo/analista-de-pruebas-senior-medellin/1885701631
- https://www.elempleo.com/co/ofertas-trabajo/automatizador-qa-junior-vivir-en-bogota/1885326945

Se calcula con costos totales mensuales de 6,608,490 COP (~1480 USD) para el empleo de cada ingeniero, lo que corresponde a 8,54USD/hora.

2.3.2. Recursos Computacionales

Para la ejecución de la estrategia de pruebas se contará con los siguientes recursos computacionales.

El plan para los recursos computacionales se base en la experiencia de la semana 6, en la que las pruebas de regresión visual se ejecutaron de forma totalmente automatizada utilizando Github Actions. Una sola ejecución de pruebas de regresión visual tardó 32 minutos o en promedio 90 segundos por prueba y ejecución.

Tipo de recurso	Cantidad	Horas totales	Características
Servidor Amazon AWS r6g.xlarge	1	400	• Procesador Graviton 64-bit ARM de 4 núcleos • 32GB de memoria RAM Servidores con 8GB de memoria RAM por núcleo utilizados para pruebas con consumo alto de recursos como las pruebas monkey, rippers y pruebas de regresión visual.
Servidor Amazon AWS a1.2xlarge	1	40	• Procesador Graviton 64-bit ARM de 8 núcleos • 16GB de memoria RAM Servidores con sólo 2GB de memoria RAM por núcleo, pero con un mayor número de núcleos, adecuados para realizar análisis estático de código y pruebas de integración.
API de generación de datos de Mockaroo	1	-	Plan Silver: • Accesos diarios ilimitados • 1 M en registros por día • Velocidad 8x
PC Portátil	4	256	• Procesador Intel Core i5 de 4 núcleos • 8GB de memoria RAM • Sistema operativo Windows 11 Se usa para el desarrollo de pruebas automatizadas.
Celular Samsung Galaxy S22	2	32	• Samsung Exynos 2200 8 núcleos, litografía 4 nm • 16GB de memoria RAM • Sistema operativo Android 12 con One UI 4.1 Se usa para pruebas exploratorias manuales
Celular Apple iPhone 13	2	16	• Procesador Chip A15 Bionic • 6GB de memoria RAM • Sistema operativo iOS 15 Se usa para pruebas exploratorias manuales

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

Para la ejecución de esta estrategia no se contratarán servicios adicionales o personal adicional.

Resumen Costos

Recursos Humanos				
	Cantidad	**Horas dedicadas por unidad**	**Valor Hora**	**Valor Total**
Ingeniero automatizador senior	4	64	US$8.54	US$2186.24
Recursos Computacionales				
Amazon AWS r6g.xlarge	1	400	US$0.2016	US$80,64
Amazon AWS a1.2xlarge	1	40	US$0.2040	US$8.16
API de Mockaroo	1	-	US$60.00	US$60.00
		TOTAL	US$ 2335.04	

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

Nivel	**Tipo**	**Técnica**	**Objetivos**
Unidad	Se buscará identificar en el código de la aplicación posibles mejoras relacionadas con buenas prácticas de programación (protección frente a entradas erróneas), usando **pruebas estáticas**.	Se realizarán pruebas estáticas, de tipo **análisis estático de código** con por lo menos dos analizadores (*linter*) como SonarQube y JSHint (**automatizadas**).	OBJ-2 y OBJ-4
Integración	Se diseñarán las pruebas **funcionales** con un enfoque de **caja negra basado en escenarios (positivos y negativos)** para el *backend* de la aplicación.	Se usará la técnica de **API de automatización**, usando un *framework* como Cypress (que permite hacer uso de *mocks* o *stubs* para este tipo de pruebas*).* Los *mocks* son para reemplazar el uso de servidores externos o bases de datos.	OBJ-1 y OBJ-4
Sistema	Se usarán los siguientes tipos de pruebas: • **Funcionales** usando un enfoque de **caja negra "end-to-end"**, buscando abarcar las principales funciones de la aplicación. Se combinarán **escenarios positivos y negativos**. • **Pruebas de reconocimiento** para validar el comportamiento	Para las pruebas de extremo a extremo (end-to-end) se usará la técnica de **API de automatización**, contando con los *frameworks* Cypress y Kraken. Los ingenieros automatizadores buscarán usar los patrones "Given-When-Then" y "Page Object" para que los scripts no dependan de aspectos concretos de pantalla. Esto para que estas pruebas puedan ser usadas más	OBJ-1, OBJ-2 y OBJ-4

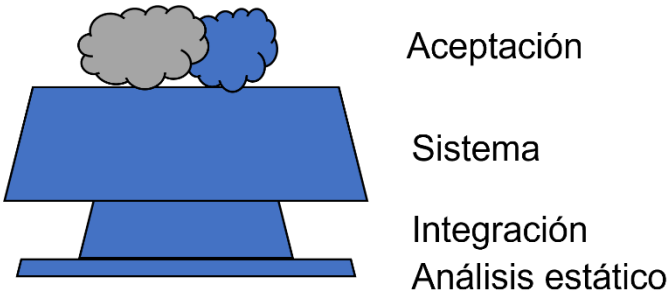
	del sistema ante eventos aleatorios. • Pruebas **funcionales** de **caja negra** con **escenarios de validación de datos.**	adelante como pruebas de regresión. Las pruebas de reconocimiento evaluarán navegabilidad y posibles errores de interfaz. Se tendrán **pruebas aleatorias ("Monkeys"),** con la librería monkey-cypress, y **pruebas sistemáticas ("Ripper")**, con la librería RIPuppet. Para las pruebas de escenarios de validación de datos se usarán estrategias **a-priori** (con datos manuales usando Scenario Outline con Kraken), **pseudo-aleatorio** (usando un API de Mockaroo para generar registros de datos, que se consume desde Cypress) y **aleatorios** (usando la librería faker.js en Kraken y Cypress). Para estas pruebas se considerarán técnicas **basadas en la experiencia, de frontera y de robustez**.	
Aceptación	Mediante **pruebas exploratorias** y **pruebas de regresión visual** se analizará la perspectiva de los usuarios publicador y administrador, quienes hacen uso de las principales funciones de la aplicación.	Las pruebas exploratorias se realizarán de forma **manual.** En este caso, como ya se tiene un conocimiento de la aplicación, estas pruebas ampliarán el inventario de pruebas ya elaborado, adicionando funcionalidades no evaluadas previamente. Las pruebas de regresión visual se realizarán usando **API de automatización.** Se ampliará el alcance de estas pruebas para que permitan evaluar diferencias entre navegadores y también entre versiones de Ghost.	OBJ1 y OBJ-3

2.5. Distribución de Esfuerzo

La distribución del esfuerzo que se desea lograr es similar a la de trofeo "Testing Trophy" (Dodds, s.f.), pero con algunos elementos de **cono de helado**: tiene una base de análisis estático, luego pruebas de integración y de sistema automatizadas y, por último, un conjunto de pruebas de aceptación (manuales y automatizadas).

La cantidad de pruebas de unidad y de integración es menor, considerando que ya se tiene un software desarrollado y se desea aprovechar la experiencia de los ingenieros automatizadores con herramientas a nivel de sistema y aceptación.

La distribución puede ver en la siguiente imagen, donde el color azul es para pruebas automatizadas y el gris para manuales:



Como ya se mencionó, se busca aprovechar la experiencia que tienen los ingenieros, para poder consolidar una base de pruebas automatizadas que sea de gran utilidad para la empresa y le permita ahorrar esfuerzos en las pruebas cuando se realicen cambios o nuevas versiones del programa. Se contempla el desarrollo de pruebas de reconocimiento, de extremo a extremo, pruebas de validación de datos (robustez, frontera) y pruebas de regresión visual. Esto se complementa con análisis estático de código, para verificar la calidad del código (buscando facilitar el mantenimiento) y para identificar posibles vulnerabilidades. En el nivel de aceptación se consideran unas pruebas manuales, principalmente pruebas exploratorias, para ir complementado el inventario de pruebas (que se puede usar luego en las automatizadas) y también contemplar aspectos de usabilidad.

Para distribuir las labores entre los ingenieros, por lo general se busca que algunos se especialicen en algunas herramientas, para que así sean más productivos, pero que siempre por lo menos dos personas trabajen cada herramienta, de manera que se pueda compartir la información y también ayudar a realizar revisiones.

En cuanto a los recursos computacionales requeridos, la experiencia de los ingenieros señala la necesidad de contar con tiempo en servidores para poder ejecutar las pruebas automatizadas, especialmente las de reconocimiento y regresión visual, que son demandantes en este tipo de recursos (tiempo de máquina). Por otra parte, para poder realizar validaciones de datos, se hace uso de un servicio especializado en la nube.

La distribución durante las ocho semanas (recordando que cada ingeniero tiene una dedicación de 8 horas semanales) es:

Semana	**Ingeniero 1**	**Ingeniero 2**	**Ingeniero 3**	**Ingeniero 4**	**Servidor AWS** r6g.xlarge	**Servidor AWS** a1.2xlarge
Uno	Definir estrategia pruebas y realizar pruebas exploratorias para complementar inventario de pruebas.	Definir estrategia pruebas y realizar pruebas de frontera para complementar inventario de pruebas.	Configurar entorno de pruebas (repositorio consolidado, ajustes en herramientas, seguimiento issues).	Configurar los analizadores estáticos, ejecutarlos y analizar los resultados.		Configurar y realizar análisis estático (8 horas).
Dos	Configurar, ejecutar y analizar resultados pruebas de reconocimiento aleatorias con monkey-cypress.	Configurar, ejecutar y analizar resultados pruebas de reconocimiento aleatorias con monkey-cypress.	Configurar, ejecutar y analizar resultados pruebas de reconocimiento sistemáticas con RIPuppet	Configurar, ejecutar y analizar resultados pruebas de reconocimiento sistemáticas con RIPuppet	Correr pruebas de reconocimiento (120 horas)	
Tres	Escribir y ejecutar pruebas de integración en Cypress (módulos del administrador).	Escribir y ejecutar pruebas de integración en Cypress (módulos del publicador).	Diseñar pruebas de integración y configurar los mocks y stubs.	Diseñar pruebas de integración y configurar los mocks y stubs.		Correr pruebas de integración (32 horas)
Cuatro	Diseñar y desarrollar pruebas e2e usando Cypress (módulos del publicador).	Diseñar y desarrollar pruebas e2e usando Cypress (módulos del publicador).	Diseñar y desarrollar pruebas e2e usando Kraken (módulos del publicador).	Diseñar y desarrollar pruebas e2e usando Kraken (módulos del publicador).	Correr pruebas e2e (32 horas)	
Cinco	Diseñar y desarrollar pruebas e2e usando Cypress (módulos del administrador).	Diseñar y desarrollar pruebas e2e usando Cypress (módulos del administrador).	Diseñar y desarrollar pruebas e2e usando Kraken (módulos del administrador).	Diseñar y desarrollar pruebas e2e usando Kraken (módulos del administrador).	Correr pruebas e2e (32 horas)	
Seis	Configurar API para generación de datos pseudo-aleatorios en Mockaroo y ajustar pruebas en Cypress para el uso de este API.	Ajustar pruebas en Cypress para usar datos aleatorios con faker.js	Ajustar pruebas en Kraken para usar datos aleatorios con faker.js	Ajustar pruebas en Kraken para usar datos a-priori (scenario outline).	Correr pruebas de validación de datos (80 horas)	
Siete	Ajustar pruebas para tomar screenshots en Cypress.	Elaborar script para realizar reporte de pruebas de	Ajustar configuración del entorno de pruebas para	Ajustar pruebas para tomar	Correr pruebas de regresión	

		regresión visual con ResembleJS.	diferentes navegadores y versiones de Ghost. Configurar GitHub actions.	screenshots en Kraken.	visual (120 horas)	
Ocho	Realizar pruebas extra, elaborar informe final.	Realizar pruebas extra, elaborar informe final.	Realizar pruebas extra, elaborar informe final.	Realizar pruebas extra, elaborar informe final.	Correr pruebas extra (16 horas)	

3. Diagrama de Gantt

Se muestra el diagrama por partes, dada su extensión. El documento completo puede encontrarse en el repositorio, carpeta Estrategia, archivo **diagramaGantt.pdf.** Enlace: https://github.com/tpambor/MISW4103-Final/blob/main/Estrategia/diagramaGantt.pdf

Semanas 1 y 2:

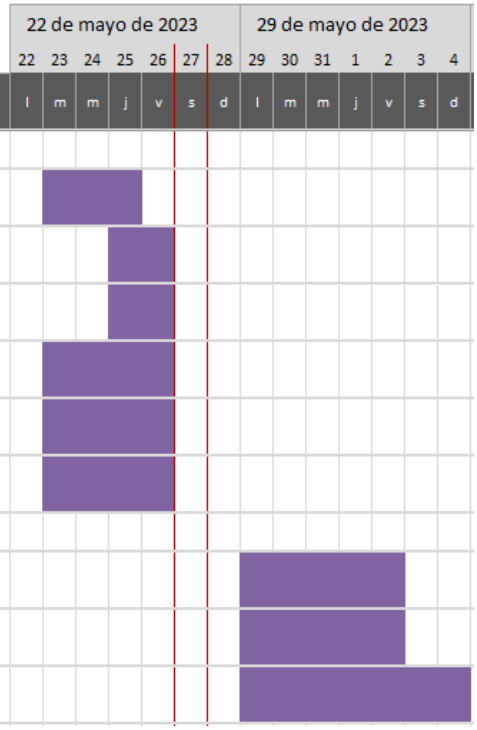
Estrategia de Pruebas - Grupo #3

Laura Alejandra Carrillo Guzmán, Sandra Victoria Hurtado Gil, Leidy Viviana Osorio Jiménez, Tim Ulf Pambor

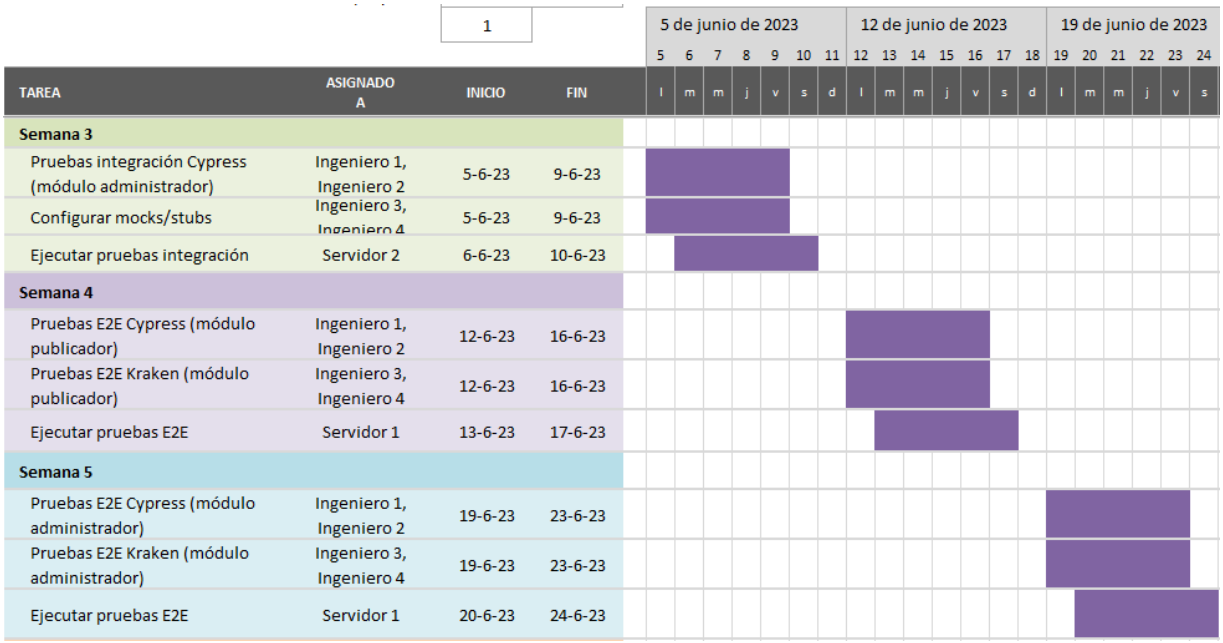
Inicio del proyecto: mar, 5/23/2023

1

TAREA	ASIGNADO A	INICIO	FIN
Semana 1			
Definir estrategia	Ingeniero 1, Ingeniero 2	23-5-23	25-5-23
Pruebas exploratorias	Ingeniero1	25-5-23	26-5-23
Pruebas de frontera	Ingeniero 2	25-5-23	26-5-23
Configuraciones	Ingeniero 3	23-5-23	26-5-23
Análisis estático	Ingeniero 4	23-5-23	26-5-23
Ejecutar análisis estático	Servidor 2	23-5-23	26-5-23
Semana 2			
Pruebas reconocimiento aleatorias	Ingeniero 1, Ingeniero 2	29-5-23	2-6-23
Pruebas reconocimiento sistemáticas	Ingeniero 3, Ingeniero 4	29-5-23	2-6-23
Ejecutar pruebas reconocimiento	Servidor 1	29-5-23	4-6-23

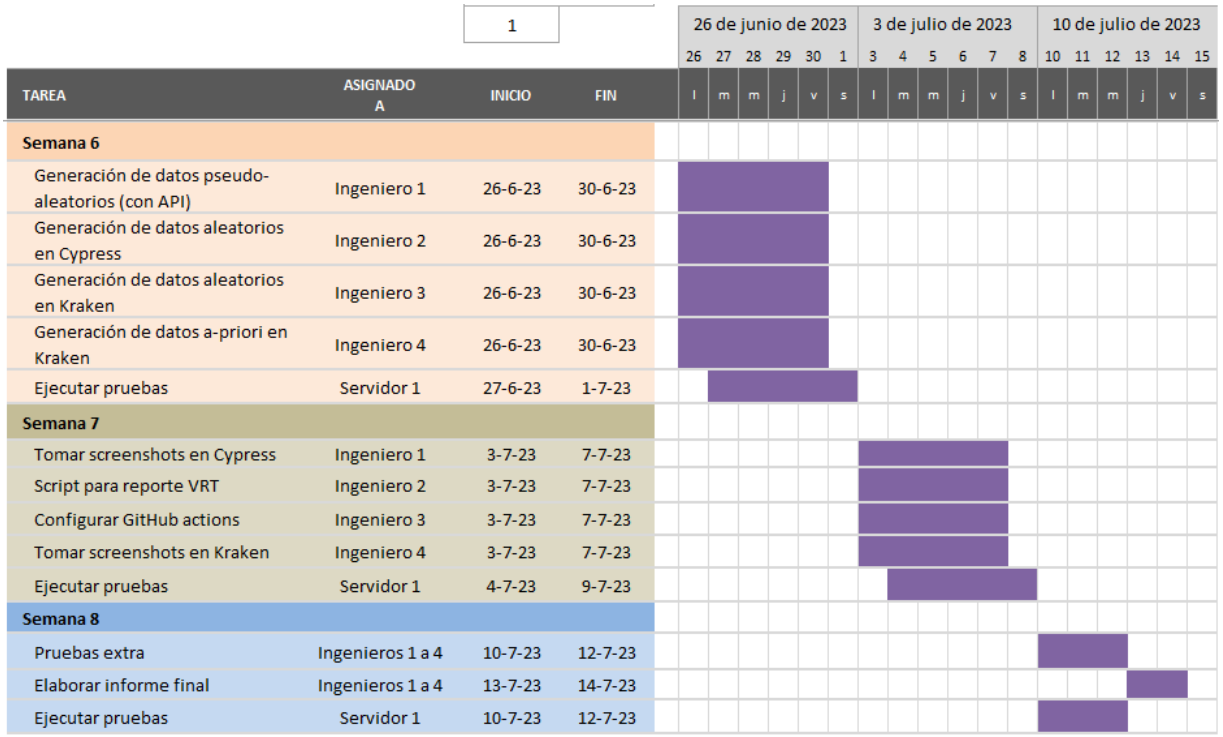


Semanas 3, 4 y 5:



TAREA	ASIGNADO A	INICIO	FIN
Semana 3			
Pruebas integración Cypress (módulo administrador)	Ingeniero 1, Ingeniero 2	5-6-23	9-6-23
Configurar mocks/stubs	Ingeniero 3, Ingeniero 4	5-6-23	9-6-23
Ejecutar pruebas integración	Servidor 2	6-6-23	10-6-23
Semana 4			
Pruebas E2E Cypress (módulo publicador)	Ingeniero 1, Ingeniero 2	12-6-23	16-6-23
Pruebas E2E Kraken (módulo publicador)	Ingeniero 3, Ingeniero 4	12-6-23	16-6-23
Ejecutar pruebas E2E	Servidor 1	13-6-23	17-6-23
Semana 5			
Pruebas E2E Cypress (módulo administrador)	Ingeniero 1, Ingeniero 2	19-6-23	23-6-23
Pruebas E2E Kraken (módulo administrador)	Ingeniero 3, Ingeniero 4	19-6-23	23-6-23
Ejecutar pruebas E2E	Servidor 1	20-6-23	24-6-23

Semanas 6, 7 y 8:



TAREA	ASIGNADO A	INICIO	FIN
Semana 6			
Generación de datos pseudo-aleatorios (con API)	Ingeniero 1	26-6-23	30-6-23
Generación de datos aleatorios en Cypress	Ingeniero 2	26-6-23	30-6-23
Generación de datos aleatorios en Kraken	Ingeniero 3	26-6-23	30-6-23
Generación de datos a-priori en Kraken	Ingeniero 4	26-6-23	30-6-23
Ejecutar pruebas	Servidor 1	27-6-23	1-7-23
Semana 7			
Tomar screenshots en Cypress	Ingeniero 1	3-7-23	7-7-23
Script para reporte VRT	Ingeniero 2	3-7-23	7-7-23
Configurar GitHub actions	Ingeniero 3	3-7-23	7-7-23
Tomar screenshots en Kraken	Ingeniero 4	3-7-23	7-7-23
Ejecutar pruebas	Servidor 1	4-7-23	9-7-23
Semana 8			
Pruebas extra	Ingenieros 1 a 4	10-7-23	12-7-23
Elaborar informe final	Ingenieros 1 a 4	13-7-23	14-7-23
Ejecutar pruebas	Servidor 1	10-7-23	12-7-23

4. Referencias Bibliográficas

Mario Linares-Vásquez y Camilo Escobar-Velásquez (2020). Pruebas automáticas de software. https://miso-4208-labs.gitlab.io/book/

Ghost Foundation (2023). Introduction. Developer Docs. https://ghost.org/docs/introduction/

StatCounter (2023). Browser Market Share Worldwide: Mar 2022 – Mar 2023. https://gs.statcounter.com/browser-market-share

Talent s.f). Salario medio para Automatizador De Pruebas en Colombia 2023 https://co.talent.com/salary?job=automatizador+de+pruebas