Estrategia de Pruebas 1

Link video: _____

1. Aplicación Bajo Pruebas

1.1. Nombre Aplicación:

Ghost

1.2. Versión:

V3.42.5

1.3. Descripción:

Ghost es un gestor de contenido o CMS profesional de código abierto y de publicación en línea. Sirve para crear y gestionar blogs. Permite la integración con otras plataformas, así como con múltiples motores de base de datos como SQLite o MySQL. Esta construida en Node.js, ofreciendo un alto rendimiento, una gran personalización y un método de escritura muy notable.
Esta versión en especifico de Ghost esta soportada en sqlite3 (https://www.sqlite.org/index.html)

1.4. Funcionalidades Core:

- Iniciar sesión
- Cerrar sesión
- Agregar publicación
- Editar publicación
- Eliminar publicación
- Agregar página
- Editar página
- Eliminar página
- Agregar tag
- Editar tag
- Eliminar tag
- Asociar tag
- Desasociar tag

1.5. Diagrama de Arquitectura:

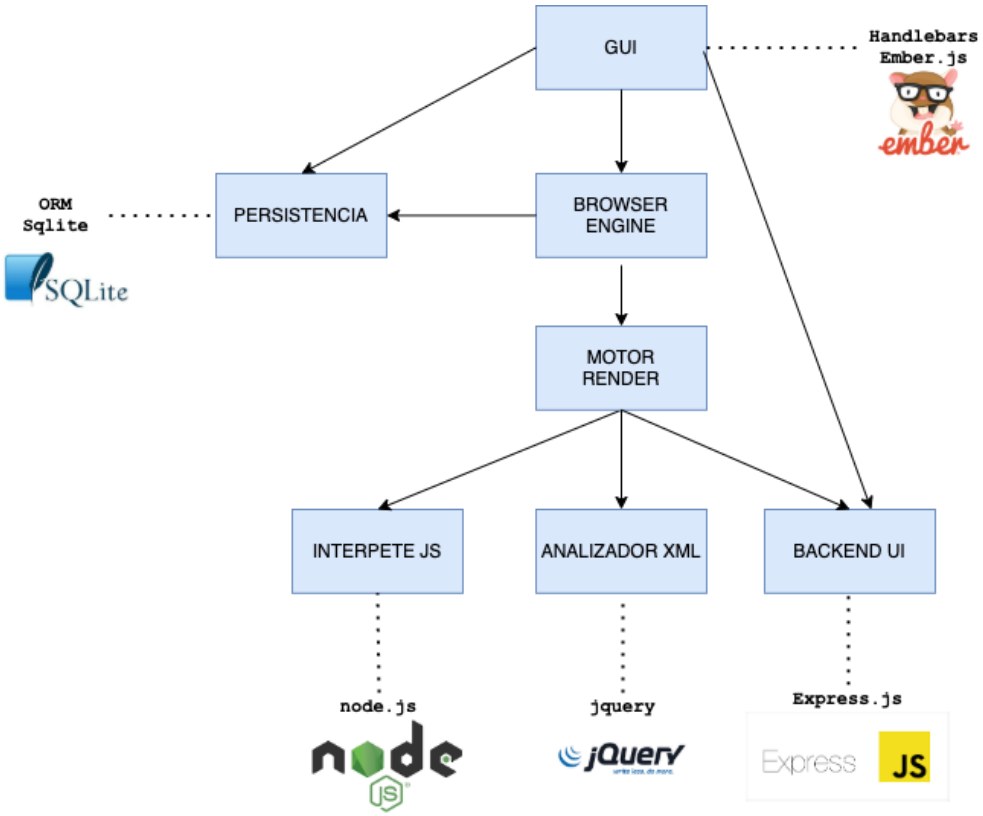


Figura 1 – Estructura a nivel de capas web

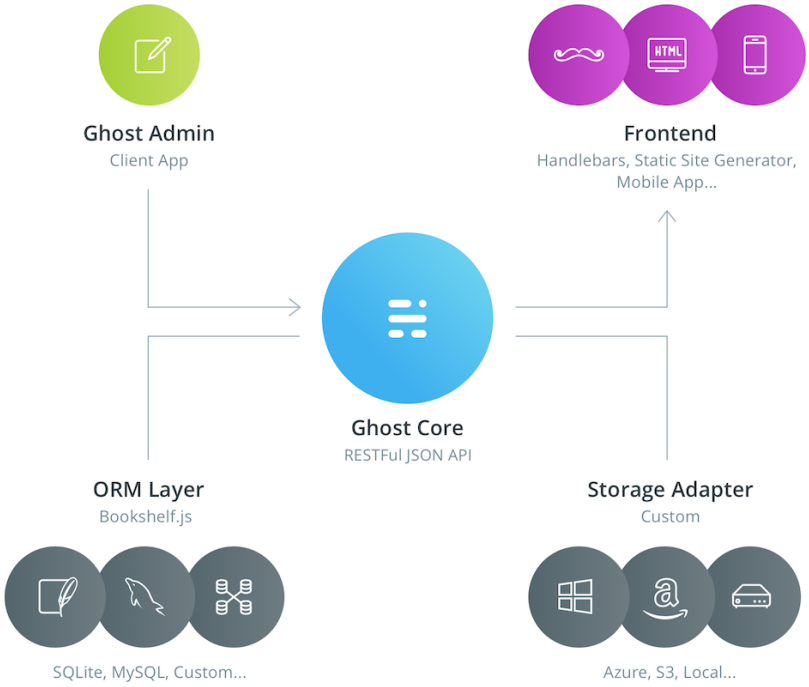


Figura 2 – Estructura simplificada Ghost (tomado de https://ghost.org/docs/architecture).

El núcleo o base de Ghost es un Una API de transferencia de estado representacional (REST) y que usa Javascript Object Notation, En cuanto a la arquitectura del Core se compone de clientes, servidores y recursos, con la gestión de solicitudes a través de HTTP. Dicho API permite desacoplamiento entre los componentes de la arquitectura, de tal forma que se pueden usar varias soluciones, ya sea para ORM, Cloud (adaptador de almacenamiento).
El componente Ghost Admin es la aplicación cliente desde la que se solicitan las operaciones hacia los demás componentes y donde finalmente terminan las respuestas de estos. Cualquier cambio en el Ghost admin tiene un minimo impacto en desempeño dado que el desacoplamiento con el Core así lo permite.
De acuerdo con el diagrama de arquitectura, el ORM que se usa es BookShelf que es un ORM basado en Javascript para Node.js y que soporta relaciones de tipo uno a uno, uno a muchos y muchos a muchos. De este componente se deriva la persistencia que no esta precisamente indicada, sin embargo, debería tener conexiones con el adapatador de almacenamiento (AWS u otros).
El frontend provee unas herramientas de GUI que están conectadas al Core, dichas herramientas contienen funcionalidades para la creación de ambientes para diversas clases de dispositivos y proveen por supuesto el GUI

1.6. Diagrama de Contexto:

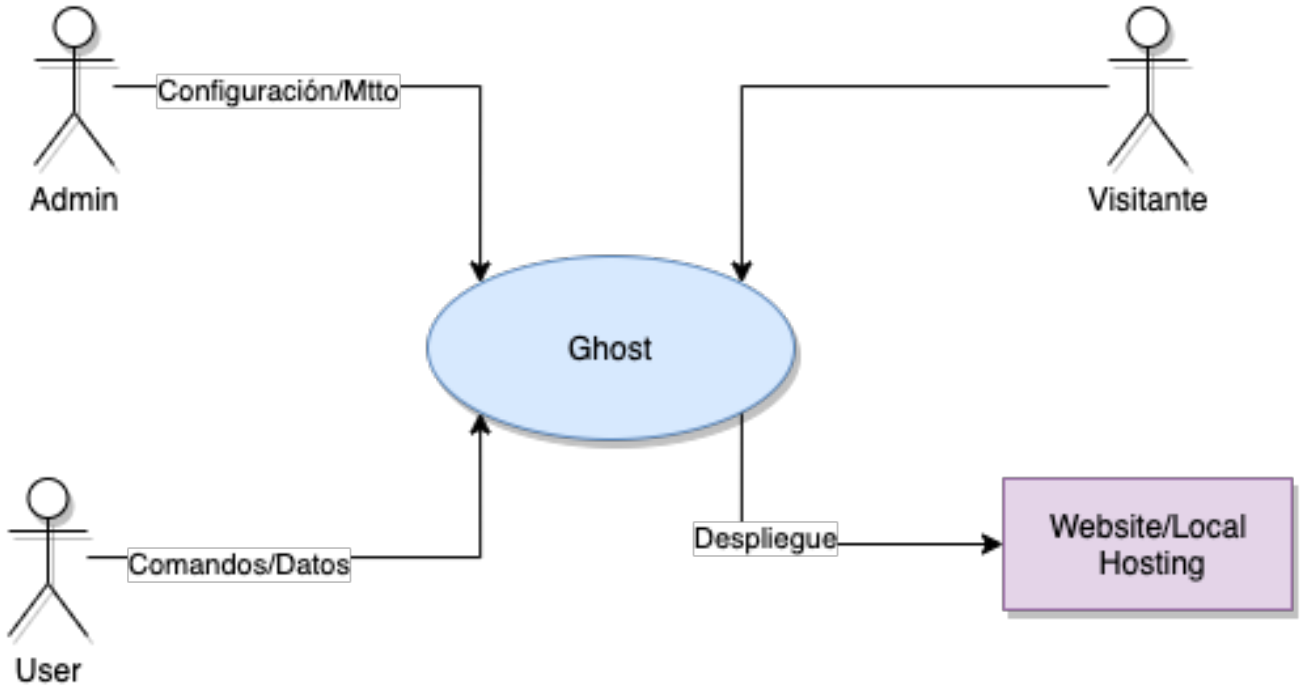


Figura 3 – Diagrama de contexto de aplicativo Ghost.

1.7. Modelo de Datos:

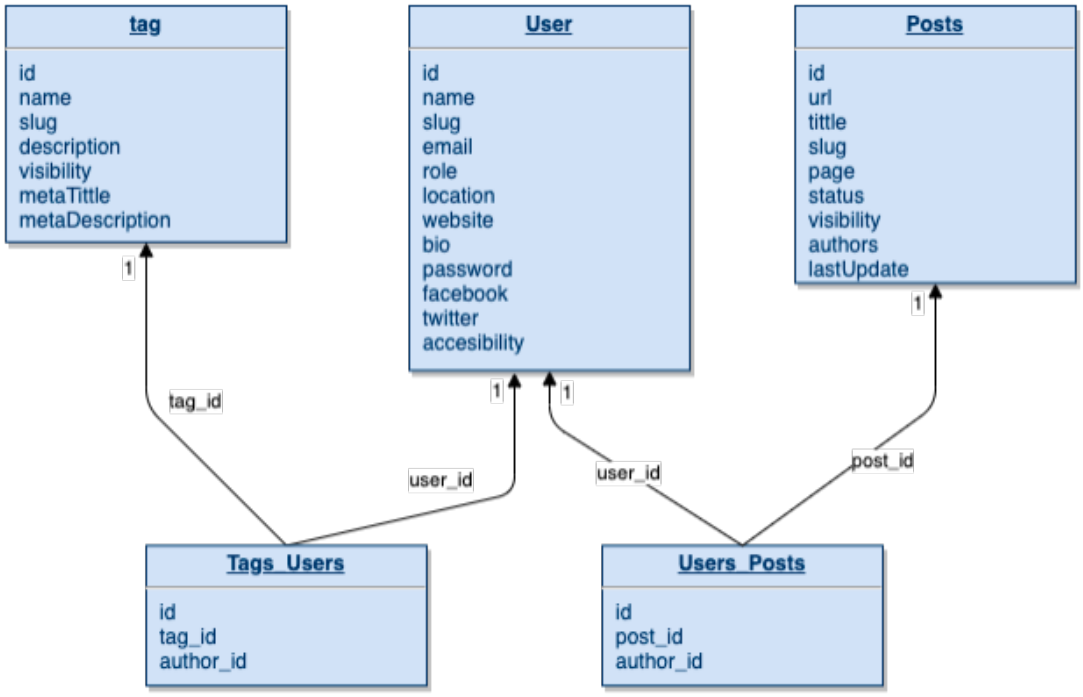
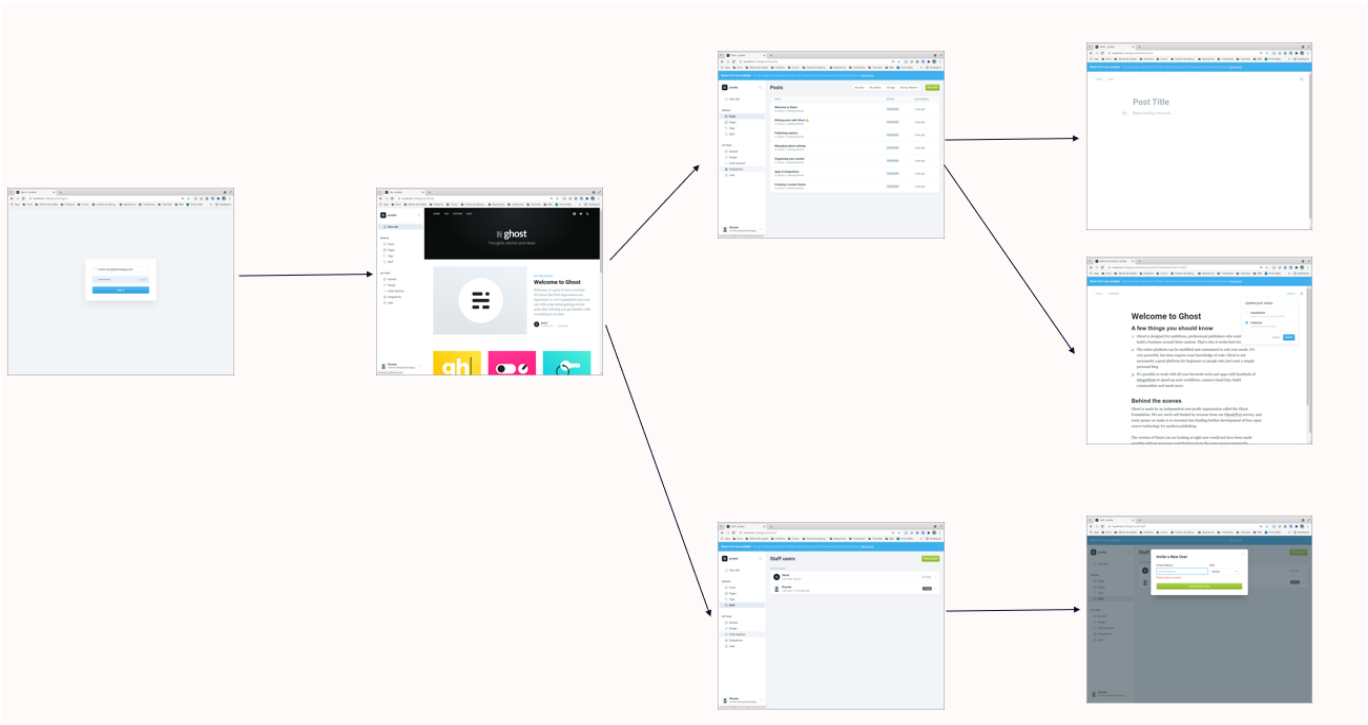


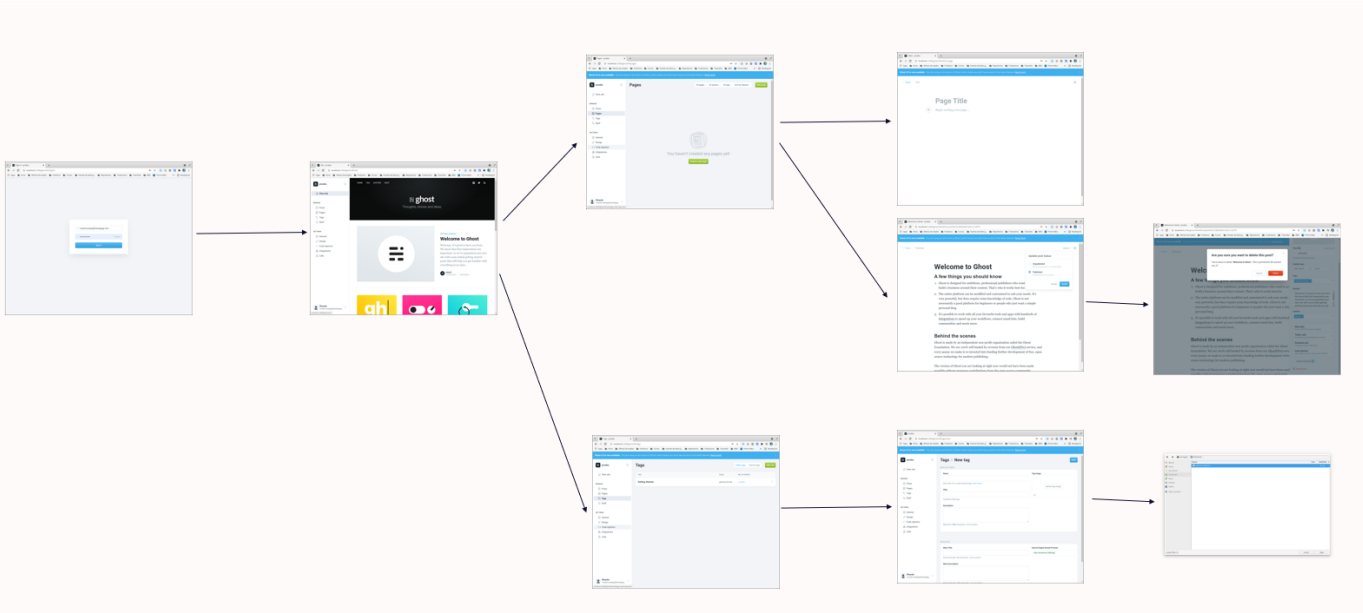
Figura 4 – Modelo de datos del aplicativo Ghost.

1.8. Modelo de GUI:

(Versión anterior - v.3.3.0) https://github.com/mgdarwin/ImagenesRecursos/blob/main/modeloGUI.pdf

(Versión actual - v.3.42.5)





2. Contexto de la estrategia de pruebas

2.1. Objetivos:

La estrategia desglosada en el presente documento tiene como objetivo primario realizar pruebas de operación a la plataforma Ghost las cuales serán abordadas desde dos enfoques como son pruebas manuales y pruebas automatizadas bajo diferentes practicas y plataformas que serán mencionadas en mayor detalle mas adelante en este documento. Dentro del esquema que se tiene preparado se espera encontrar por medio de las pruebas manuales y automatizadas defectos dentro de la aplicación o posibles inconvenientes en la operación no solo desde el punto de vista de casos de prueba sino de escenarios completos como lo son las pruebas e2e; es así como para esta estrategia se plantean los siguientes objetivos.

- Realizar pruebas exploratorias sobre la aplicación Ghost v3.42.5 para evidenciar la funcionalidad general y sus componentes representativos, de la misma obtener resultados iniciales de *issues* y defectos para plantear su corrección.
- Aplicar pruebas funcionales de caja blanca y negra en diferentes niveles como son de unidad sobre las funcionalidades *core*, de integración entre estas y de aceptación con el fin de detectar, registrar y documentar los resultados e *issues* que se presenten dando una buena cobertura funcional sobre el aplicativo.
- Diseñar y ejecutar de manera automatizada pruebas
    - Unitarias
    - De integración
    - Regresión visual
    - Extremo a extremo o e2e
    - De validación de datos

Sobre las funcionalidades core de Post, Tag, Page y Staff con el fin de ejecutar un monto considerable de pruebas sobre la interfaz y detectar con resultados cuantitativos posibles desviaciones de la operación.

2.2. Duración de la iteración de pruebas:

Las diferentes pruebas trazadas serán ejecutadas durante un periodo de ocho (8) semanas de manera consecutiva con una asignación de treinta y dos horas hombre por semana. La distribución de las HH y las estrategias de pruebas se muestran de manera resumida a continuación.

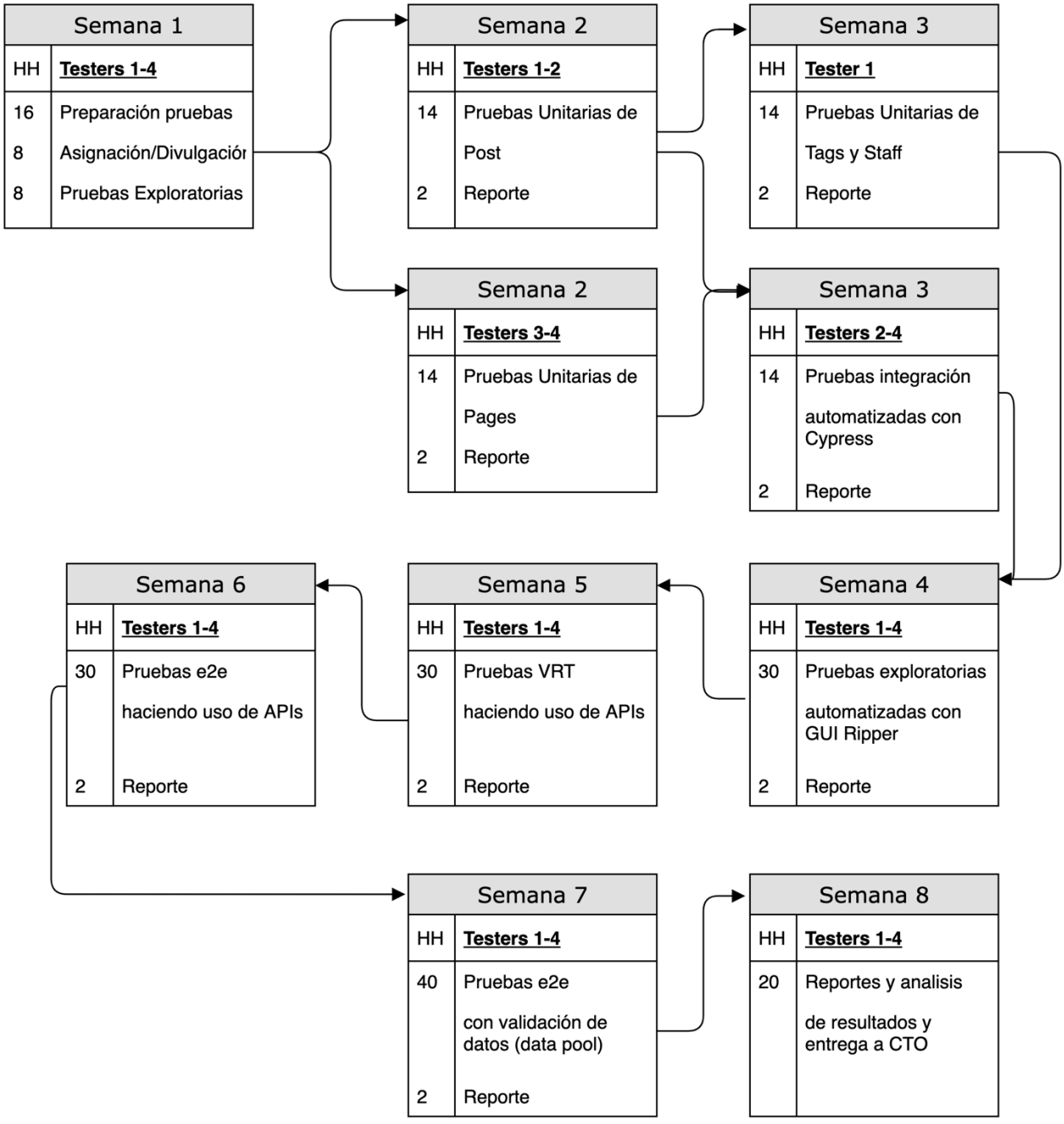


Figura 5 – Resumen de asignación y flujo de actividades de la estrategia de pruebas propuesta..

para completar un total de doscientas cincuenta y seis (256) horas dividido entre pruebas manuales y automatizadas.

2.3. Presupuesto de pruebas:

El presupuesto está dividido en dos aspectos principales como son recursos humanos y computacionales, los rubros están basados en las condiciones establecidas para las pruebas de este plan que se ejecutará en ocho (8) semanas de labores. La tabla siguiente resume el presupuesto requerido.

ÍTEM	DESCRIPCIÓN	UNIDAD	CANTIDAD	VALOR UNITARIO	VALOR TOTAL
1	Ingeniero senior de automatización de pruebas	Hora/Hombre	256	$30.468	$7.800.000
2	Servidor de pruebas AWS (*Opcional)	Hora/Maquina	200	$7.400	$1.480.000
TOTAL PRESUPUESTO					**$9.280.000**

El presupuesto para el ítem 1 es basado enteramente en el salario medio-superior para Automatizador De Pruebas en Colombia 2021 dado en la página https://co.talent.com/salary?job=Automatizador+De+Pruebas cuyo valor se muestra en la siguiente figura.

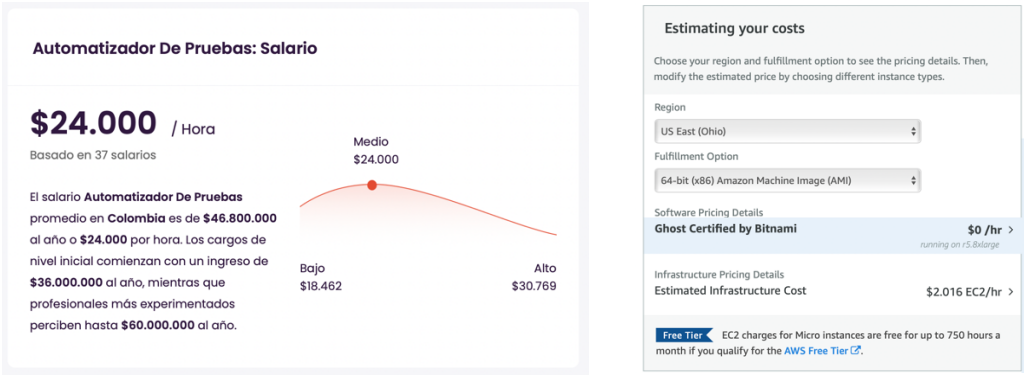


Figura 6 – (a) Fuente para costo base de ingeniero automatizador. (b) Costo instancia AWS.

Para el cálculo del ítem 2 dos se ha hecho uso de la calculadora de precios para las diferentes instancias ofrecidas en AWS, el detalle de la instancia se resume en la figura 2, este valor es opcional pero requerido dado que por el monto de pruebas debe establecerse un equipamiento adicional para su ejecución de manera paralela.

2.3.1. Recursos Humanos

Se considera entonces:
Cuatro testers senior, durante 8 semanas, en una dedicación de 8 horas por persona semanalmente, para un total de doscientas cincuenta y seis (256) horas hombre.

Dentro de las horas descritas arriba se requiere que el personal conozca las herramientas de pruebas disponibles para el proyecto, la preparación de los escenarios de pruebas y el alistamiento y planeación de los componentes de pruebas y del proyecto en general.

2.3.2. Recursos Computacionales

Cada ingeniero del grupo tiene asignado un computador i7 con 16GB de ram y 120GB de almacenamiento para realizar las pruebas. Se cuenta con herramientas de grabación web como Loom o Screencastify que pueden ser instalados como complementos del navegador web.

Se tienen 200 horas/máquina en Amazon AWS como recursos disponibles en la nube para realizar pruebas automatizadas.

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

La estrategia no contempla el uso de servicios Outsourcing. Se contempla contratación directa de los cuatro (4) testers senior

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

Se ratifica el TNT de las pruebas de esta estrategia, dado que el costo de hacerlas es relativamente comparado con el costo de tener mas Horas Hombre haciendo pruebas manuales. De acuerdo con la experiencia de pruebas usando GUI Ripper, son mucho mas efectivas y tienen un alcance mayor, para encontrar errores en GUI, por lo que la distribución y carga de estas pruebas es menor que las pruebas unitarias, por ende, sigue siendo un modelo piramidal. Además, el presupuesto para este caso es menor que la opción 1 que incluye más pruebas manuales.

Se realizarán técnicas de pruebas manuales y automatizadas incluyendo pruebas exploratorias para conocer el estado del sistema. Los ingenieros automatizadores se enfocarán en pruebas del sistema para probar el aplicativo como un todo, pero granulándolo por funcionalidades y escenarios de pruebas. Para construir los escenarios de pruebas se usará la técnica de caja negra es decir que para diferentes escenarios con entradas definidas se obtenga el resultado esperado. Posteriormente los ingenieros usarán recursos en la nube para realizar pruebas automatizadas al sistema y así garantizar la calidad esperada.

PATRON	TIPO	NIVEL	TECNICA	OBJETIVO
Pirámide de pruebas	Pruebas funcionales, de caja blanca, caja negra positivas y negativas	Unidad	API's de pruebas unitarias	Validar el correcto funcionamiento de la app y sus funcionalidades
	Pruebas funcionales de caja negra, positivas y negativas	Sistema	Pruebas manuales exploratorias	A través de estas manuales se realizarán pruebas exploratorias para conseguir obtener errores no detectados por los Monkeys o Rippers
	Pruebas funcionales, de caja negra, positivas y negativas	Sistema	GUI Rippper	Adquirir conocimiento del producto a través de la realización de pruebas exploratorias
	Pruebas funcionales, de caja negra y positivas	Aceptación	Regresión Visual + APIs	Buscar diferencias entre versiones, teniendo como base una versión anterior
	Pruebas funcionales, de caja negra, positivas y negativas	Aceptación	validación de datos (pools y random)	Busca validar las respuestas del sistema de acuerdo con una heurística definida
	Pruebas funcionales, de caja negra, positivas y negativas	Aceptación	API's de automatización E2E	Validar la correcta integración entre las diferentes funcionalidades de la app

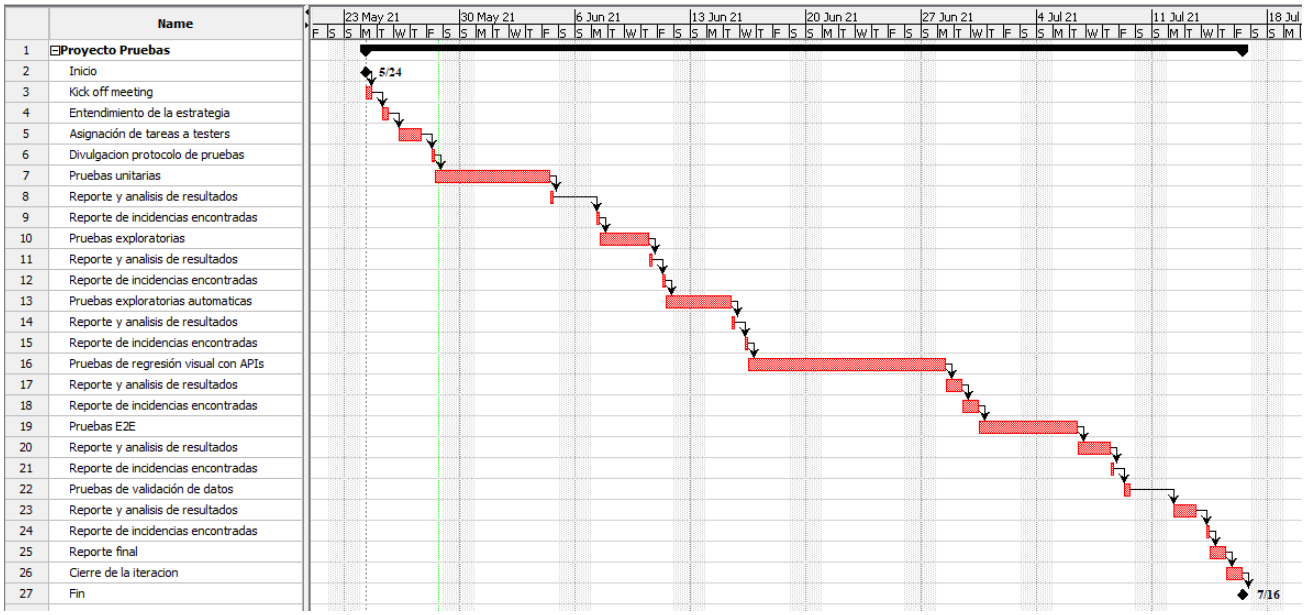
2.5. Distribución de Esfuerzo

Al contar con cuatro (4) personas dedicadas a las pruebas del programa bajo pruebas, la metodología de pruebas sigue los patrones relacionados con pirámide de pruebas, de esta manera, las estrategias pondrán a las pruebas unitarias automatizadas como base y a la que mayor recurso se dará en cuanto a horas hombre y horas máquina en conjunto con las pruebas de integración y pruebas de componentes.
En cuanto a la distribución de esfuerzo para la semana se tiene el siguiente cronograma, este abarca las ocho (8) semanas requeridas para el proyecto.
Cabe destacar que las pruebas con APIs automatizados para E2E son de preferencia en primera instancia Cypress y en segunda, Cucumber webdriver.io por ser las APIs más versátiles a ala hora de realizar este tipo de pruebas, velocidad en la ejecución, simplicidad en el código disponibilidad de generar múltiples escenarios, usar tablas de datos a priori y generación de data aleatoria.

Cronograma detallado de actividades:

Proyecto pruebas	Tipo	Nivel	Tecnica	39.5 days?	5/24/21, 8:00 AM	7/16/21, 1:00 PM
Inicio				0 days	**5/24/21, 8:00 AM**	**5/24/21, 8:00 AM**
Kick off meeting				1 day	5/24/21, 8:00 AM	5/24/21, 5:00 PM
Entendimiento de la estrategia				1 day	5/25/21, 8:00 AM	5/25/21, 5:00 PM

Proyecto pruebas	Tipo	Nivel	Tecnica	39.5 days?	5/24/21, 8:00 AM	7/16/21, 1:00 PM
Asignación de tareas a testers				2 days	5/26/21, 8:00 AM	5/27/21, 5:00 PM
Divulgacion protocolo de pruebas				0.5 days	5/28/21, 8:00 AM	5/28/21, 1:00 PM
Pruebas unitarias	Pruebas funcionales, de caja blanca, caja negra positivas y negativas	Unidad	API's de pruebas unitarias	5 days	5/28/21, 1:00 PM	6/4/21, 1:00 PM
Reporte y analisis de resultados				0.5 days	6/4/21, 1:00 PM	6/4/21, 5:00 PM
Reporte de incidencias encontradas				0.5 days	6/7/21, 8:00 AM	6/7/21, 1:00 PM
Pruebas exploratorias	Pruebas funcionales de caja negra positivas y negativas	Sistema	Pruebas manuales exploratorias	3 days	6/7/21, 1:00 PM	6/10/21, 1:00 PM
Reporte y análisis de resultados				0.5 days	6/10/21, 1:00 PM	6/10/21, 5:00 PM
Reporte de incidencias encontradas				0.5 days	6/11/21, 8:00 AM	6/11/21, 1:00 PM
Pruebas exploratorias automáticas	Pruebas funcionales, de caja negra y positivas	aceptación	GUI Ripper	2 days	6/11/21, 1:00 PM	6/15/21, 1:00 PM
Reporte y análisis de resultados				0.5 days	6/15/21, 1:00 PM	6/15/21, 5:00 PM
Reporte de incidencias encontradas				0.5 days	6/16/21, 8:00 AM	6/16/21, 1:00 PM
Pruebas de regresión visual con APIs	Pruebas funcionales, de caja negra, positivas	aceptación	Regresión Visual + APIs	8 days	6/16/21, 1:00 PM	6/28/21, 1:00 PM
Reporte y análisis de resultados				1 day	6/28/21, 1:00 PM	6/29/21, 1:00 PM
Reporte de incidencias encontradas				1 day	6/29/21, 1:00 PM	6/30/21, 1:00 PM
Pruebas E2E	Pruebas funcionales de caja negra	aceptación	Pruebas E2E con APIs	4 days	6/30/21, 1:00 PM	7/6/21, 1:00 PM
Reporte y análisis de resultados				2 days	7/6/21, 1:00 PM	7/8/21, 1:00 PM
Reporte de incidencias encontradas				0.5 days	7/8/21, 1:00 PM	7/8/21, 5:00 PM
Pruebas de validación de datos	Pruebas funcionales, de caja negra, positivas y negativas	aceptación	validación de datos E2E (pools y random)	1 day	7/9/21, 8:00 AM	7/9/21, 5:00 PM
Reporte y analisis de resultados				2 days	7/12/21, 8:00 AM	7/13/21, 5:00 PM
Reporte de incidencias encontradas				0.5 days	7/14/21, 8:00 AM	7/14/21, 1:00 PM
Reporte final				1 day	7/14/21, 1:00 PM	7/15/21, 1:00 PM
Cierre de la iteracion				1 day	7/15/21, 1:00 PM	7/16/21, 1:00 PM
Fin				0 days	7/16/21, 1:00 PM	7/16/21, 1:00 PM


Name
1 Proyecto Pruebas
2 Inicio
3 Kick off meeting
4 Entendimiento de la estrategia
5 Asignación de tareas a testers
6 Divulgacion protocolo de pruebas
7 Pruebas unitarias
8 Reporte y analisis de resultados
9 Reporte de incidencias encontradas
10 Pruebas exploratorias
11 Reporte y analisis de resultados
12 Reporte de incidencias encontradas
13 Pruebas exploratorias automaticas
14 Reporte y analisis de resultados
15 Reporte de incidencias encontradas
16 Pruebas de regresión visual con APIs
17 Reporte y analisis de resultados
18 Reporte de incidencias encontradas
19 Pruebas E2E
20 Reporte y analisis de resultados
21 Reporte de incidencias encontradas
22 Pruebas de validación de datos
23 Reporte y analisis de resultados
24 Reporte de incidencias encontradas
25 Reporte final
26 Cierre de la iteracion
27 Fin
23 May 21
30 May 21
6 Jun 21
13 Jun 21
20 Jun 21
27 Jun 21
4 Jul 21
11 Jul 21
18 Jul
5/24
7/16