Estrategia de Pruebas

1. Aplicación Bajo Pruebas

1.1. **Nombre Aplicación:** Ghost

1.2. **Versión:** 3.42.5

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.

1.4. **Funcionalidades Core**:

- **01. Búsqueda:** Permite buscar recursos como posts, users y tags dentro del dashboard administrativo
- **02. Gestionar post:** Permite utilizar la funcionalidad core del sistema crear, editar, eliminar post
- **03. Gestionar staff:** Permite editar la información básica de los perfiles del sistema e invitar a usuarios a gestionar el sitio, con diferentes roles.
- **04. Gestionar página:** Permite al creador de contenido y administrador crear, editar o eliminar páginas en el sitio
- **05. Gestionar tag:** Permite crear, editar, eliminar etiquetas para posteriormente añadirlos al post
- **06. Crear integración:** Permite crear integraciones con software de terceros a nuestro sitio
- **07. Gestionar diseño:** Permite personalizar mediante esta funcionalidad el diseño del sitio

1.5. Diagrama de Arquitectura:

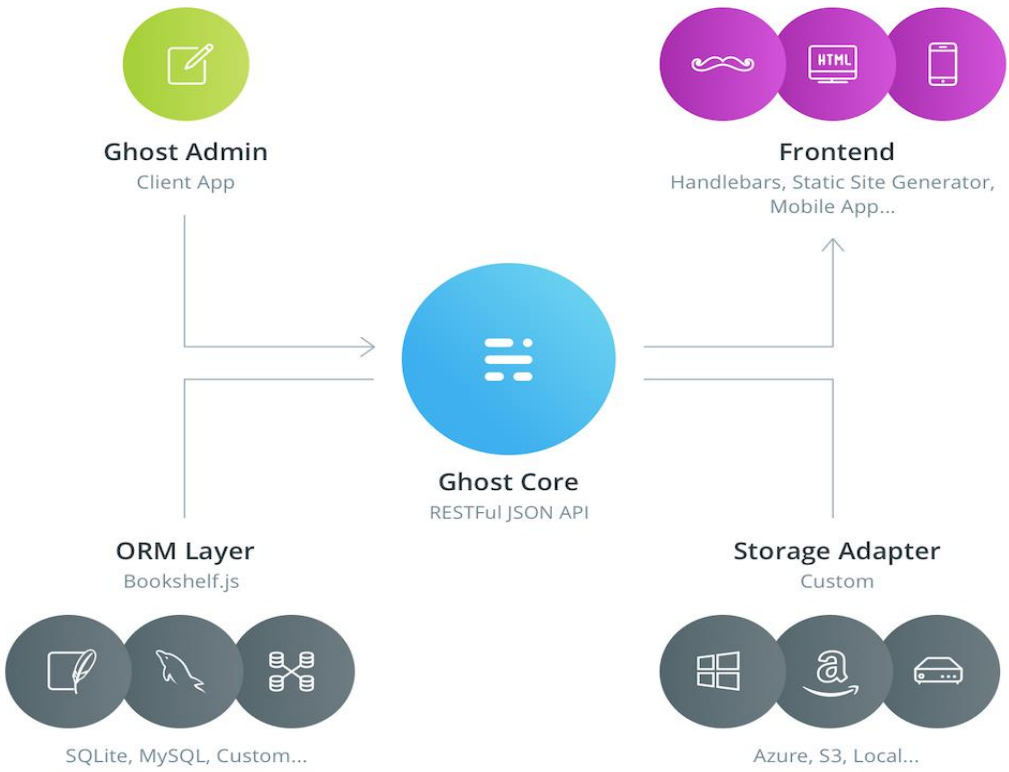


Figura 1. Arquitectura Ghost
Tomado de https://ghost.org/docs/architecture/#ghost-core

La arquitectura general de ghost se encuentra en la Figura 1, desacoplada y basada en una arquitectura orientada a servicios, acompañada de los siguientes componentes que se describen a continuación:

- **Ghost Core**
  - Este módulo es el corazón de Ghost, expuesto como un RESTful JSON API, diseñado para crear, gestionar y obtener el contenido con facilidad.
  - Ghost API: Se encuentra dividido por dos partes, contenido y administración, ambos con sus métodos correspondientes de autenticación.

- **ORM Layer:**
  - Esta capa permite persistir y exponer las configuraciones que usan los demás servicios.
  - Ghost usa un ORM para Node.js llamado bookshelf.js. Este se encuentra diseñado para trabajar con diferentes motores de bases de datos como PostgreSQL, MySQL, SQLite3 etc. Por defecto Ghost soporta SQLite3 en su entorno de desarrollo, mientras que recomienda MySQL para los entornos de producción.

- **Storage Adapter**
  - Ghost permite la integración personaliza con alternativas de almacenamiento, por ejemplo, extender el sistema de archivos a servicios externos como Azure, S3 o incluso local-storage

- **Ghost Admin – Admin Api**
  - Esta capa se encarga de gestionar el contenido y roles, hechos en el Admin API de ghost, es quien se encarga de leer y escribir contenido del sitio.

- **Front end**

  - Ghost se define como un full-headless CMS, por lo cual lo hace agnóstico ante cualquier framework front end. Extendiéndose así a infraestructuras móviles o aplicaciones nativas.

1.6. Diagrama de Contexto:

Ghost al ser un sistema manejador de contenidos, permite el manejo de roles de acuerdo con las propiedades de administración, edición, contribución de los posts en el sitio, o un simplemente de lectura de estos para un usuario externo.

Los rores de los usuarios en Ghost se clasifican en Administrador, Autor, Editor y Contribuidor. En la figura 2 se muestra el diagrama de contexto de la aplicación. Se puede observar como a través de la plataforma se interactúa con la base de datos que pueda ser local o remota.

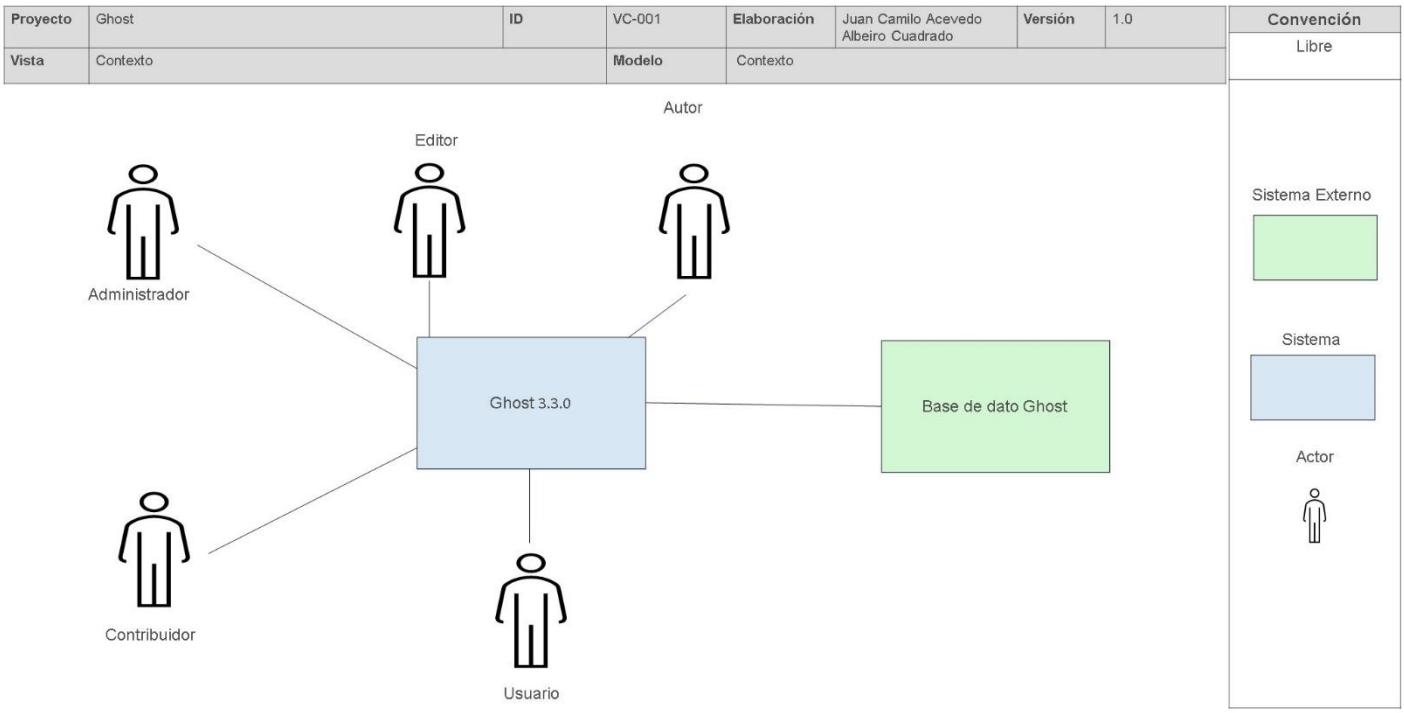
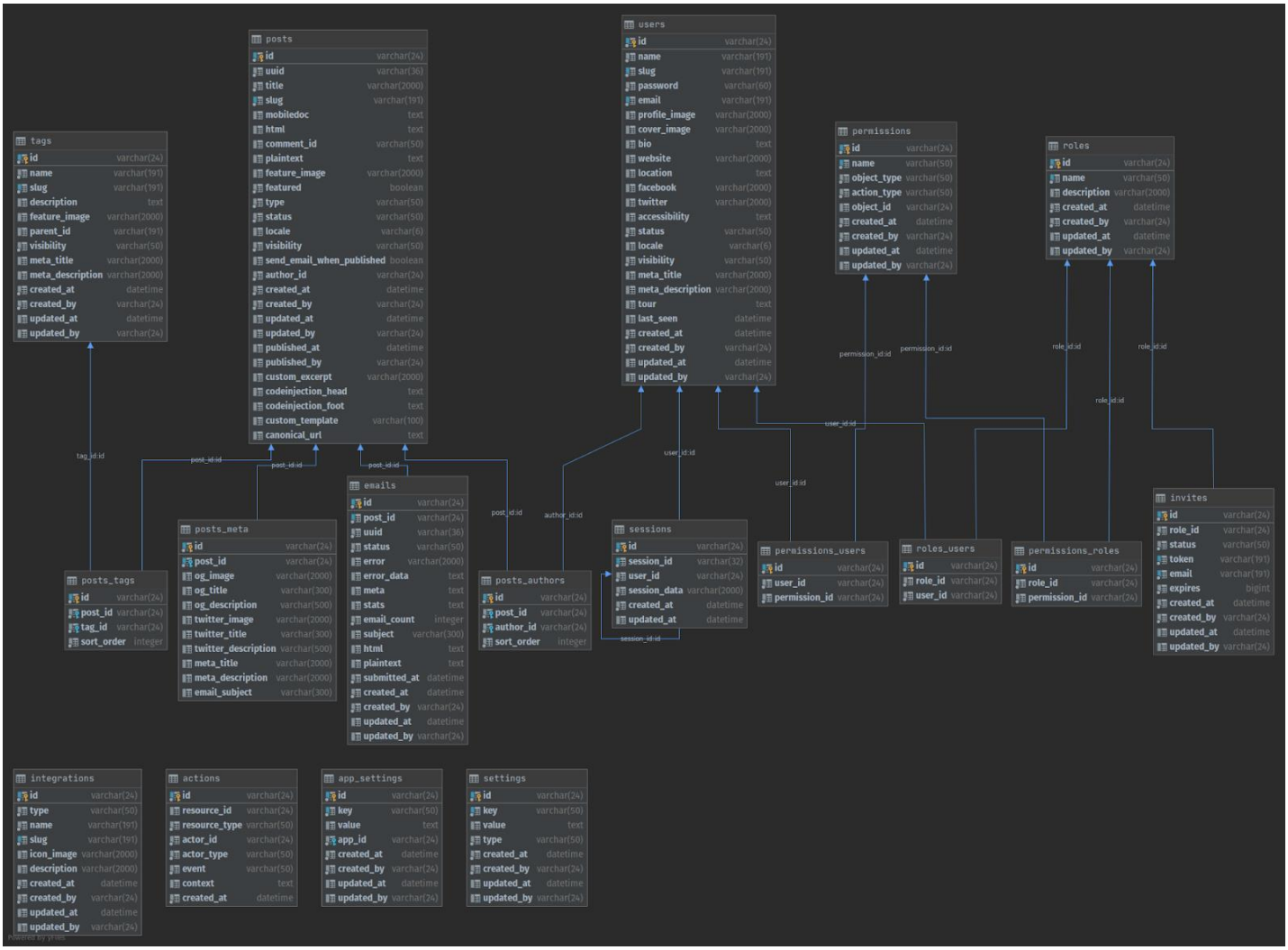


Figura 2. Diagrama de contexto Ghost

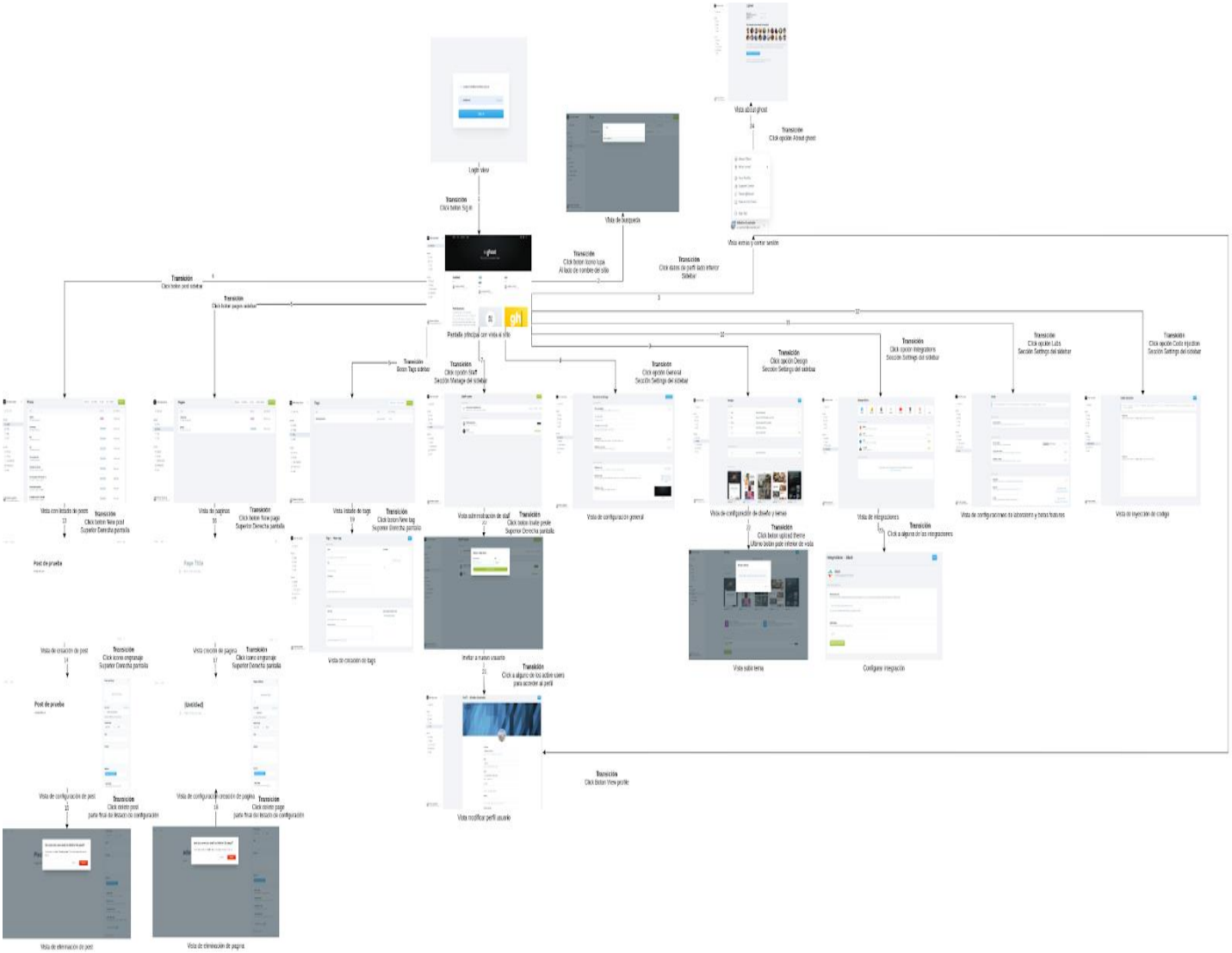
1.7. Modelo de Datos:

El modelo mostrado a continuación, se puede encontrar en una mejor resolución en el archivo de nombre "**Modelo_de_Datos.png"** que se encuentra en la raíz del comprimido en el que se encuentra este documento.



1.8. Modelo de GUI:

El modelo mostrado a continuación, se puede encontrar en una mejor resolución en el archivo de nombre “**Modelo_de_GUI.png”** que se encuentra en la raíz del comprimido en el que se encuentra este documento.



2. Contexto de la estrategia de pruebas

2.1. Objetivos:

2.1.1. Objetivo General

Desarrollar e implementar una estrategia de pruebas para determinar si el software Ghost, cumple con los criterios de calidad establecidos por el equipo de desarrollo y el cliente.

2.1.2. Objetivos Específicos

- Diseñar estrategias de pruebas que involucren pruebas manuales, de reconocimiento y de "extremo a extremo" (E2E), en el contexto dado por una aplicación bajo pruebas.
- Construir pruebas de reconocimiento y E2E usando *frameworks* de automatización existentes.
- Ejecutar pruebas de reconocimiento y E2E usando *frameworks* de automatización existentes.

2.2. Duración de la iteración de pruebas:

Basado en el personal y las horas disponibles para la ejecución de las pruebas, se muestra en la tabla 1 como será el plan de trabajo.

Tabla 1. Plan de trabajo

Semana	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Ingeniero 1								
Ingeniero 2								
Ingeniero 3								
Ingeniero 4								
Total, horas	**32**	**32**	**32**	**32**	**32**	**32**	**32**	**32**
	256 horas							

0. Preparación de pruebas
1. Pruebas exploratorias manuales
2. Pruebas de reconocimiento usando Monkey y rippers
3. Pruebas End To End (E2E)
4. Pruebas de regresión visual (VRT)
5. Pruebas de validación de datos, usando datos aleatorios
6. Informe de cierre con el reporte de incidencias y resultados

Se tiene un estimado que, para el cumplimiento de las pruebas, cada ingeniero tome 8 horas por semana, para un total de 64 horas por ingeniero.

2.3. Presupuesto de pruebas:

- Cuatro testers senior (64 horas/persona)

2.3.1. Recursos Humanos

El perfil de “Testers Senior” para la realización de las pruebas, deberá cumplir con el perfil que se describe a continuación:

Profesional con estudios en ingeniería de automatización, sistemas, electrónica, telecomunicaciones, software, mecatrónica o carreras afines con conocimientos en automatización de pruebas, manejo de pruebas funcionales y no funcionales, bases de datos y programación.

Experiencia en Javascript, Nodejs, Ruby, bases datos relacionales y herramientas para reporte de incidencias. Debe tener experiencia en el área superior a 6 meses.

El tiempo estipulado para el periodo de prueba es de 64 horas por cada ingeniero, distribuidas en jornadas laborales de lunes a viernes en horario de 8:00 am – 12:00 pm y 2:00 pm – 6:00 pm, durante un periodo de 8 semanas. El personal seleccionado debe cumplir con una jornada de 8 horas semanales para cumplir la meta programada.

2.3.2. Recursos Computacionales

Los equipos para la realización de las pruebas serán entregados por la empresa. Para la gestión de las pruebas se usarán equipos de cómputo que tengan por lo menos las siguientes características:

- Procesador Core I3 o equivalentes
- Disco de estado sólido con una capacidad de 240 gigas o superior
- Capacidad en memoria RAM de 8 gigas o superior
- Sistema operativo Windows 10 Pro o distribuciones Linux con Kernel 5 en adelante
- Conexión a Internet a través de conexión GigaEthernet

Para el tráfico de datos se deberá contar con una conexión a Internet estable, por medio de canales cableados y con un ancho de banda mínimo de 50 megas por equipo de cómputo usado.

SE RECOMIENDA A FUTURO SI SE CUENTA CON EL PRESUPUESTO Adquirir los siguientes servicios para extender y escalar la estrategia de manera eficiente

- **AWS Device Farm**
- **AWS Amazon Inspector**
- **AWS CodePipeline**
- **AWS EC2**

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

El personal que se contratará estará trabajando por cumplimiento de objetivos, es decir que al finalizar cada semana se deberá reportar las metas logradas. Se espera que cada ingeniero trabaje 8 horas semanal, durante un periodo de 8 semanas.

TestMail (https://testmail.app**)**

Presupuesto tomado: $ 0 USD (https://testmail.app/pricing)

Se usará el plan *Free* del proveedor testmail.app, el cual nos permitirá tener múltiples direcciones y bandejas de correo para automatizar las pruebas end to end de los módulos que requieran interacción por medio de correos, por ejemplo, funcionalidades como, *recuperar contraseña, invitar al sitio, registro, notificaciones, etc.*

Este plan nos permitirá ejecutar:

- 100 emails por mes
- Número ilimitado de bandejas, direcciones de correos y suite de spam tests

BrowserStack (https://browserstack.com)

Presupuesto tomado: $ 0 USD (https://www.browserstack.com/open-source?ref=pricing)

Al ser ghost un proyecto open source, podemos acceder al servicio de BrowserStack de manera gratuita, este nos permitirá:

- Minutos de testing ilimitados
- Todos los navegadores de escritorio incluyendo versiones de Windows y MacOS
- Soporte para la automatización con Cypress
- Más de 80 dispositivos para probar
- Screenshots
- Integración con otros servicios como slack etc.
- Panel de gestión
- Pruebas de geolocalización

Coveralls (https://coveralls.io/**)**

Presupuesto tomado: $ 0 USD

Al ser ghost un proyecto open source, podemos usar el servicio coveralls de manera gratuita, esto nos permitirá, mantener estadísticas del coverage del repositorio en donde se aloja el proyecto, cobertura del código línea por línea, reporte general, log de coverage, etc.

Lo que brindará mejor visibilidad, de las pruebas implementadas sobre las líneas escritas del proyecto.

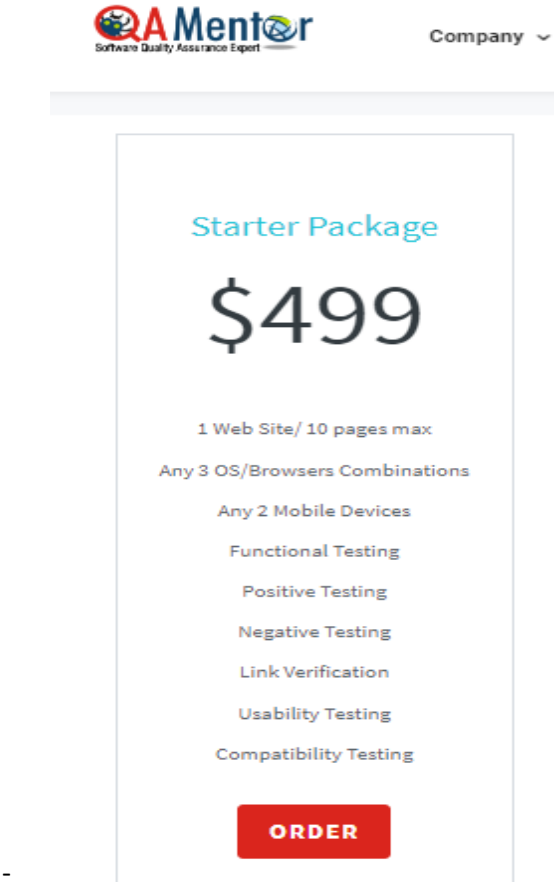
SonarLint (https://www.sonarlint.org/**)**

Presupuesto tomado: $ 0 USD)

Se usará la extensión de IDE SonarLint, como analizador estático de código con el fin de detectar y corregir error de calidad. Esta extensión no tiene ningún costo.

SE RECOMIENDA A FUTURO SI SE CUENTA CON EL PRESUPUESTO Adquirir los siguientes servicios para extender y escalar la estrategia de manera eficiente

- **QAMentor (https://qamentor.com)**
- **Presupuesto:** $ 499 USD
- Se recomendaria usar el *Starter package* del proveedor qamentor.com, el cual ayudará con la ejecución del siguiente plan de prueba que incluye:



-

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

Para el cumplimiento de los objetivos se hace necesario aplicar diferentes tipos y técnicas de pruebas en los niveles asociados a la aplicación.

De acuerdo con el plan de trabajo que se presentó en la tabla 1, las técnicas, niveles y tipos de las pruebas aplicadas al proyecto se muestran en la tabla 2.

Tabla 2. TNT

Nivel	Tipo	Técnica	Objetivo
Unidad	Funcionales No funcionales Caja negra Caja blanca	APIs de automatización Record and replay	1. Pruebas exploratorias manuales 2. Pruebas de reconocimiento usando Monkey y rippers
Integración	Positivas Negativas	Pruebas aleatorias	1. Pruebas exploratorias manuales 2. Pruebas de reconocimiento usando Monkey y rippers 3. Pruebas End To End (E2E)
Sistema	Caja negra	APIs de automatización Record and replay Pruebas de reconocimiento Pruebas aleatorias	2. Pruebas de reconocimiento usando Monkey y rippers 4. Pruebas de regresión visual (VRT) 3. Pruebas End To End (E2E) 5. Pruebas de validación de datos, usando datos aleatorios
Aceptación	Funcionales	Exploración sistemática	4. Pruebas de regresión visual (VRT) 3. Pruebas End To End (E2E) 5. Pruebas de validación de datos, usando datos aleatorios

2.5. Distribución de Esfuerzo

Para alcanzar los objetivos planteados en el proyecto se hará uso de la primera semana en la elaboración del plan de trabajo definición de las herramientas y equipos a usar. Para esta semana se usarán 32 horas que equivalen al 12.5% del total de las horas. La idea es que con esta etapa se realice el setup técnico de la herramienta, generación de data pools, limitaciones, alcances, definición de escenarios, preparación de ambiente.

A partir de la información en la Tabla 1. Plan de trabajo y la Tabla 2. TNT; se puede aplicar el patrón de distribución de esfuerzo de la pirámide **de automatización**, en el cual, el 12.57% del tiempo se usará en la implementación de pruebas exploratorias manuales. Este tiempo equivale a 32 horas de los ingenieros, las cuales se dedicarán a la etapa de conocimiento de la aplicación. Los ingenieros tomarán como base el inventario existente desarrollado en las semanas anteriores el cual se encuentra en el repositorio con el nombre de "**Inventario.xlsx"**, allí extenderán los escenarios y utilizarán los formatos propuestos.

Las horas restantes se usarán en las etapas pruebas de reconocimiento usando Monkey y rippers, Para llevar a cabo las pruebas de reconocimiento y el cumplimiento de los objetivos anteriores, se hará uso de monkey-cypres y RIPuppet. Basándonos en la sugerencia de los asesores de The software Design Lab. Iniciando así la prueba de concepto y ejecución de la misma, como quedó plasmado en el documento anexo ***"Pruebas de reconocimiento monkey-cypress y RIPuppet.pdf"***. Donde se pueden apreciar resultados de la ejecucion de la estrategia, pros, contras y conclusiones.

Para las pruebas de regresión visual (VRT) se usarán las herramientas Resemble y backstop, pueden consultar los pros y contras de usar esta herramienta aquí (https://github.com/albcm-uniandes/PA_week5/wiki/Pros-y-contras-de-las-herramientas-(-Resemble-&-Backstop)) además dentro del repositorio encontrarán el directorio resemble en donde podrán encontrar el código soporte para ejecutar la comparación de versiones.

Para las pruebas de validación de datos, se guiarán del readme del repositorio, en donde encontraran el inventario de escenarios y la metodologia de generación aleatoria seguida https://github.com/albcm-uniandes/PA_week5

Las pruebas E2E son las que más tiempo necesitan de acuerdo a la planeación. Se realizarán durante dos semanas con un total de 64 horas de los ingenieros, lo que equivale al 25% del total del tiempo disponible.

La última semana será usada para el análisis de resultados y generación de informes.