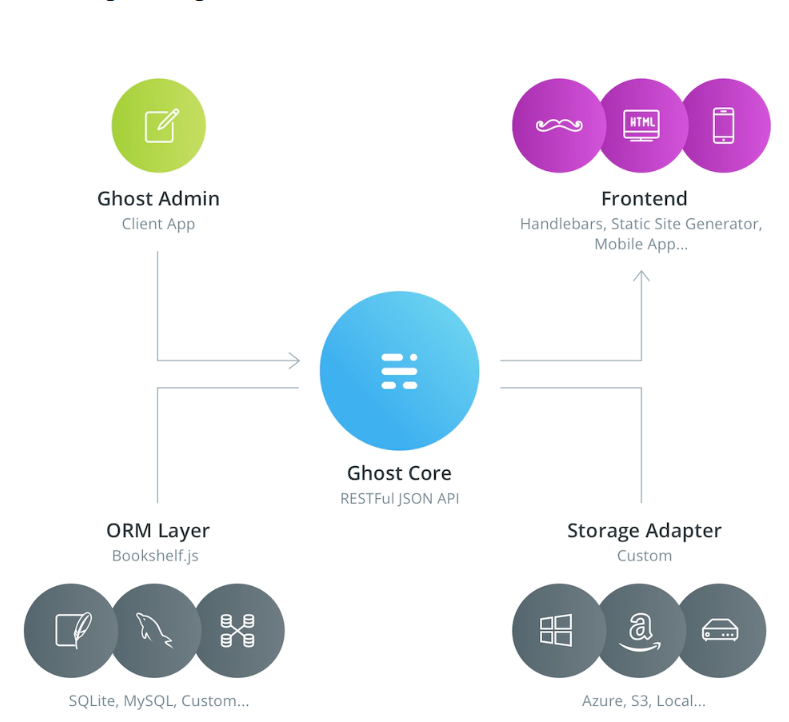
Estrategia de Pruebas

Cuatro testers senior, durante 8 semanas, en una dedicación de 8 horas por persona semanalmente.

1. **Aplicación Bajo Pruebas**
   1. **Nombre Aplicación:** Ghost
   2. **Versión:** 4.41.3
   3. **Descripción**:

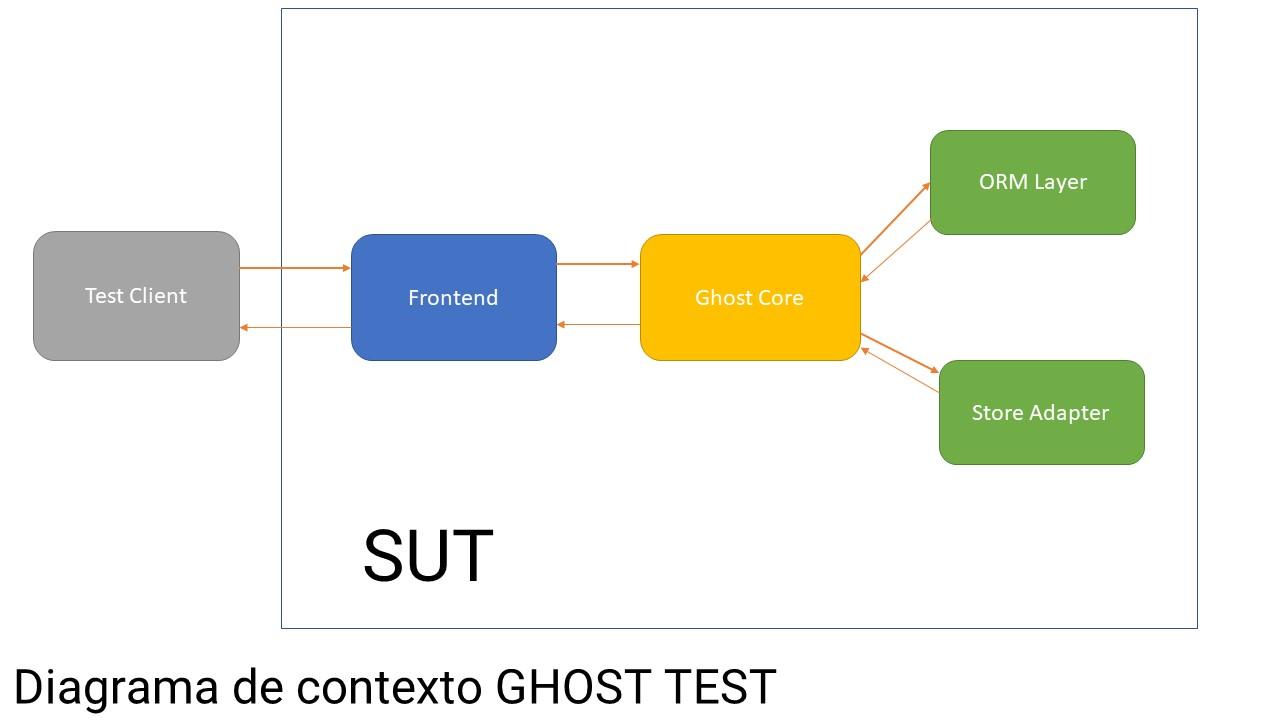
Ghost es una aplicación poderosa para que los creadores de nuevos medios publiquen, compartan y hagan crecer un negocio en torno a su contenido. Viene con herramientas modernas para crear un sitio web, publicar contenido, enviar boletines y ofrecer suscripciones pagas a los miembros.

* 1. **Funcionalidades Core**:
* Agregar Post.
* Actualizar Post
* Listar Post
* Publicar post.
* Guardar vista Post
* Agregar página
* Actualizar página
* Publicar página
* Asignar tags a página
* Listar página
  1. **Diagrama de Arquitectura:**
* A core JSON API
* A admin client app
* A front-end theme layer



* 1. **Diagrama de Contexto:**

Se plantea utilizar las pruebas en un ambiente de Servidor hermético para la aplicación con el fin de encapsular la aplicación en una máquina virtual para inicializar la aplicación y ejecutar las pruebas accediendo directamente al Front



* 1. **Modelo de Datos:**

El modelo de dominio fue construido tomando como referencia la base de datos de la aplicación de esta forma se presenta en [este enlace](https://github.com/j-albarracin-uniandes/pruebas-automatizadas/wiki/Modelo-de-dominio)  como un modelo entidad de relación (MER). También está disponible para descargar como imagen o en formato .erd el cual se puede abrir desde el programa DBeaber (Versión recomendada 21.3.3).

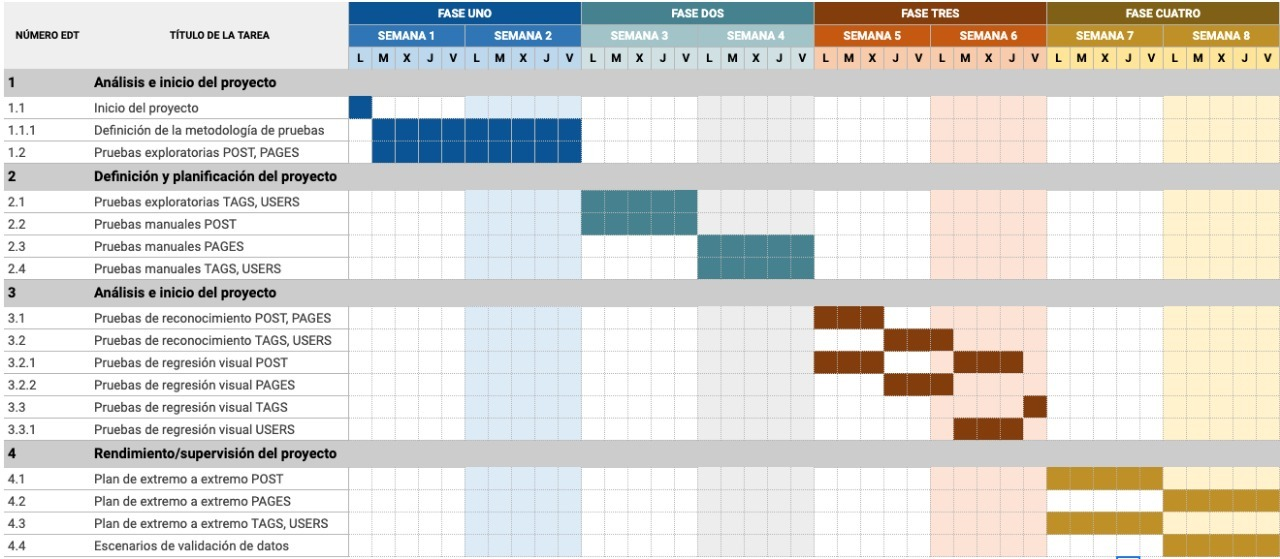
**1.8. Modelo de GUI:**

El modelo de dominio fue construido tomando como referencia la interfaz de usuario de la aplicación, dividida por vistas numeradas con un identificador único. Este modelo se encuentra en [este enlace](https://uniandes-my.sharepoint.com/:p:/r/personal/j_albarracinp_uniandes_edu_co/_layouts/15/doc.aspx?sourcedoc=%7Bbf4673f0-1553-4942-b4e1-e5b65b9bcc6a%7D&action=edit)  las imágenes que componen este se encuentran en el repositorio en el [siguiente enlace](https://github.com/j-albarracin-uniandes/pruebas-automatizadas/tree/master/modeloGUI) diferenciadas cada una con su correspondiente identificador.

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

El enfoque que se dará a las pruebas está dividido entre pruebas manuales, pruebas de reconocimiento, pruebas de extremo a extremo, pruebas de regresión visual y escenarios de validación de datos, divididos de la siguiente manera:

* Pruebas Manuales POST, PAGES, TAGS y USERS.
* Pruebas de reconocimiento POST, PAGES, TAGS y USERS.
* Evaluar módulos de funcionalidades de Post y Página solo en la interfaz gráfica
* Verificación lo funcional
* Validar lo funcional
* Enfocarnos en pruebas automatizadas
* Enfocarnos en realizar pruebas de aceptación
  1. **Duración de la iteración de pruebas :**

****

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

En el grupo de pruebas estará conformado por:

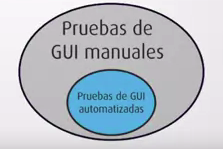
* Un ingeniero de pruebas Senior, que se encargará de automatizar todas las tareas de usuario final o de cliente sobre la aplicación, (60 horas).
* Un ingeniero de pruebas Senior, contratado a través de outsourcing para automatizar las tareas de administrador (60 horas).
* Un ingeniero de pruebas Senior, para realizar pruebas exploratorias sobre la aplicación (68 horas).
* Un ingeniero de pruebas Senior, para realizar las pruebas de caja negra sobre la aplicación (68 horas).
  + 1. **Recursos Computacionales**

Cada ingeniero cuenta con un equipo dedicado para realizar sus respectivas pruebas, adicionalmente se dispone de 400 horas de procesamiento en un servidor AWS para la ejecución de las pruebas automatizadas una vez estén terminadas.

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

Se cuenta con 500 USD, distribuidos de la siguiente manera:

* $200 para contratar 40 horas de un ingeniero automatizador para realizar automatización de tareas del módulo administrativo.
* $200 para contratar 40 horas de un ingeniero de pruebas para realizar pruebas de caja negra.
* $100 para contratar 30 horas de un ingeniero Junior para realizar pruebas exploratorias.
  1. **TNT (Técnicas, Niveles y Tipos) de pruebas:**
* Técnicas
  + Aunque vamos a explorar inicialmente las componentes vamos a dejar las pruebas automatizadas que permite dejar un grupo de pruebas durante todo el sistema



* Niveles
  + *Estamos probando dado el presupuesto el probar componentes de forma aislada y pruebas funcionales exploratorias del sistema.*
  + Pruebas de Unidad
  + Pruebas de Sistema (Exploratorias)
* Tipos
  + Pruebas funcionales positivas de caja negra
  1. **Distribución de Esfuerzo**

Para esta iteración se hará uso de la distribución de esfuerzos tipo Pirámide de Alister Scott, sin embargo, por el tiempo necesario para poder cumplir a cabalidad con dicha distribución se acota esta, enfocándonos en las pruebas automatizadas de API y las pruebas de componentes para poder validar que las funcionalidades cumplen con lo requerido, así como identificar posibles fallos evaluando escenarios negativos y poco comunes donde dicha funcionalidad puede verse afectada.

Dado que el trabajo se asignó a nivel de componentes funcionales para cada desarrollador y considerando que los componentes a evaluar tienen una funcionalidad similar y no requieren recursos extra a la hora de ser probados se plantea dividir los recursos en 3, dado partes equitativas a que cada desarrollador realice las mismas tareas en el mismo margen de tiempo, pero con diferentes componentes.

Comenzando con las pruebas automatizadas de componentes y finalizando con las pruebas automatizadas de API de cada funcionalidad listada en el cronograma. Primero validando y verificando que cada funcionalidad del componente hace lo que debe hacer para después comenzar con la validación de los casos atípicos que pueden llegar a generar fallos en el sistema a probar.