Tecnologías y características del Portal Gatsby

Todavía hace falta asignar prioridades y cómo aproximarse a la arquitectura del sitio, esquemas y construcción, pero esto es básicamente la (probablemente incompleta) lista de características clave y tecnologías a emplear

---

- Frontend/Generador de sitio estático:
  - Gatsby

- Backend/CMS Truncado:
  - Strapi CMS

- Preprocesador CSS:
  - Stylus

- Estructurador de Frontend:
  - Bulma/Bulma-stylus

- CI/CD:
  - Webhooks (Para compilación estática y cambios realizados por usuarios)
  - Drone (Línea para compilación completa del sitio, incluyendo cambios a estructuras de GraphQL, esquemas y modificaciones globales)

- Integraciones:
  - Git/GitHub (Disponer de un repositorio para los cambios accesibles a los usuarios; archivos markdown, cambios en base de datos, etc)
  - Prometheus/Grafana (Para tener monitoreo adecuado de los cambios realziados y un pundo de checkeo de salud del sitio)

- Consideraciones para el Backend:
  - Posiblemente habrá que escribir un servicio de conversión entre la base de datos que use Strapi y el markdown o lo que sea que convenga parasincronizar con el repositorio de Git. Hay que ver si un sistema de archivos plano es viable o funcional con Strapi o si es mejor escribir la capad e conversión adicional
  - Escribir integraciones con Mattermost y Kanboard o software de administrac ión de proyectos que usemos, para informar de cambios, actualizaciones y actividad a todo el equipo
  - La estructura de datos del sitio viejo es un desastre, hay que borrarla por completo y diseñar de entrada una mejor bien ordenada
  - Hay que ver cómo integrar D3/Chart.js a Strapi de manera que los datos puedan ser presentados adecuadamente de manera nativa, sin necesitar depender de elementos externos, o para agregar interactividad
  - Diseñar los roles de usuarios en Strapi de manera de restringir el acceso a los editores a algunos cambios (por ejemplo permitir que cambien esquemas de GraphQL puede ser demasiado y propiciar que algo se rompa si lo camibian)

    - Si es posible, habilitar un precompilador de MathJax --> SVG para poder incluir fórmulas y datos en el sitio de manera adecuada. Esto puede ser importante para los futuros análisis, encuestas o información técnica a compartir
    - Junto con MathJax, agregar una forma adecuada de mostrar tablas de datos y procesar información cruda automáticamente desde accesos personalizados usando D3 y quizás Chart.js, de manera tal que todos los datos de origen de los análisis que se realicen estén disponibles de manera accesible, fácil de visualizar y aprovechar.

- Consideraciones para el Frontend:
    - La integración de D3/Chart.js requerirá algo de trabajo en el frontend para asegurarse que React/Gatsby compile y muestre correctamente todo
    - El (poco) js personalizado para el sitio tiene que ser eliminado y agregado si hace falta como un gancho o llamada a función en la estructura frontend de React
    - Hay un montón de cosas que incluir, así que hay que ver qué paquetes son superfluos, y qué podemos usar para minificar y reducir el tamaño de los archivos y la cantidad de llamadas realizadas al servidor (que probablemente sean pocas porque va a ser servido de una sola fuente)

- Consideraciones generales:
    - Ver si podemos disponer el sition en una instancia de S3 o DO Spaces para servirlo desde ahí, con los elementos dinámicos siendo tomados desde el server principal
    - Usando S3 o equivalente, hay que ver de activar el caché de extremos de cloudflare y otras cosas que pueden mejorar el sitio