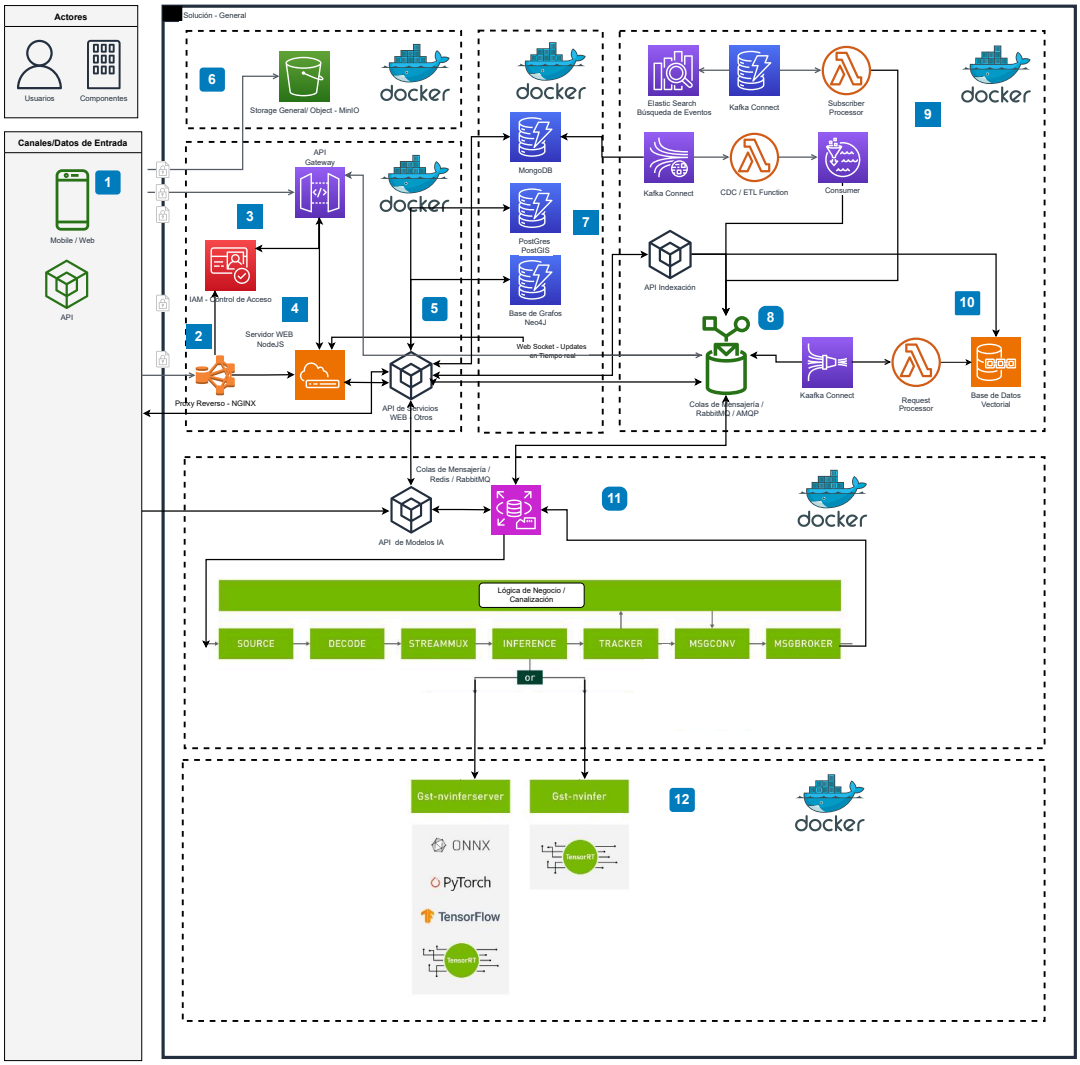
# Arquitectura Reto Datos Con Historia

Descripción de Alto Nivel de la Solución Propuesta




Descripcioón de Componenetes

1. Acceso por un servidor web local que tiene un proxy reverso basado en Apache Nginx. Todos los otros dispositivos se comunican por medio de una API y/o un componente del contenedor del servidor web y/o del contenedor que tiene el servicio del Colas de mensajería abre una conexión remota.
2. Proxy reverso que permite escalar varios de los componentes de la solución planteada utilizando Apache Nginx. Control de Acceso a las diferentes aplicaciones / OAuth2 - JWT. Los datos de autenticación estarán en la base de datos Mongo y en caso de necesidad se utilizará KeyCloak.
3. Componente de API gateway para lograr balanceo de carga, control de throughput y diferentes de temas de seguridad. Este API gateway puede funcionar como un pequeño bus de integración para diferentes protocolos: por ejemplo, websocket y/o basados en REST.
4. Servicio WEB que tiene una aplicación basada en node.js que tendrá todos los casos de uso como autenticación, un CRUD básico de procesos de carga y transformación de archivos basado en formato PDF y/o EXCEL. También se implementarán todos los otros casos de uso necesarios para las 3 áreas de la Secretaría Distrital de Planeación y todos sus diferentes roles.
5. API de servicios implementada usando SpringBoot y java para atender todos los servicios de capa media necesarios por el servidor web, extraer los datos desde algún otro componente de la solución en caso de necesidad.
6. Servicios de almacenamiento de objetos basado en un protocolo compatible con S3 y/o similar. Se propone la herramienta de código abierto MinIO. En caso de que en el futuro se desee migrar a una solución de nube, se puede cambiar este servicio sin problema.
7. Servicios de base de datos. La solución contempla una base de datos no estructurada basada en MongoDB. También se contempla una base de datos postgres para persistir información de tipo geo-espacial (extensión postig) y cualquier otro tipo de información estructurada y relacional necesaria. También se contempla una base de datos basada en grafos como Neo4J o similar.
8. Servidor de colas de mensajería para poder implementar toda la lógica de comunicación asíncrona propuesta en la solución. Se propone utilizar RabbitMQ y/o un servidor basado en el protocolo AMQP. Este protocolo permite escalar todo el proceso de indexación con uno o mas nodos respectivamente para realizar todos los trabajos de indexación de los diferentes documentos y/o casos de uso en batch y/o streaming necesarios. Este servicio es fundamental para recibir todos los eventos generados por los pipelines y/o canalizaciones de procesamiento de modelos de Inteligencia Artificial.
9. Componentes para poder hacer procesamiento en streaming de eventos. Se utiliza para implementar el patrón de captura (Change Data Capture) y así poder realizar la persistencia de los textos de los diferentes documentos y persistirlos en un servidor de ElasticSearch. Este componente permitirá realizar búsquedas hibridas. Se fundamenta en el protocolo de Kafka, Kafka Connect y todos los conectores necesarios. Este componente permitirá generar alarmas y eventos de interés frente a reglas de negocio como geocercas, objetos en una región de interés, entre otros.
10. En esta sección se tiene el servicio de la base de datos vectorial basado en una tecnología como Milvus y/o similar. Este motor se encargará de persistir los sumergimientos (embeddings - por su nombre en inglés). Esto le permitirá a la solución realizar búsquedas semánticas, implementar varios temas de similitud semántica, entre otros. Este servicio está disponible por medio de la API visualizada en el contenedor y/o conectado directamente a las colas de mensajería.
11. Contenedor Docker que implementa una solución basada en la tecnología de nvidia, python y otros servicios para exponer uno o mas modelos de Inteligencia Artificial de una manera sincrónica y/o asincrónica. Este módulo cuanta con un servicio de cola de mensajería para hacer replicación y/o atender solicitudes en batch de manera asincrona. Servicios de canalización y flujos de trabajo para encadenar uno o mas modelos en un flujo de trabajo con capacidad para escalar cada componente del módulo ode manera independiente. Integración con mensajería basadas en MQTT y/o KAFKA en caso de necesidad.
12. Contenedor Docker dedicado a correr triton-server y/o tensro_rt para exponer modelos de inteligencia artificial, procesamiento de lenguaje natural en múltiples formatos como ONNX y pytorch.