Visualización de datos en tiempo real con tecnologías Web

RESUMEN

Se desea tener una plataforma web que reciba información de sensores y que ésta los despliegue en una gráfica en tiempo real.

OBJETIVOS

1. Comprobar conocimiento en desarrollo web FrontEnd/Angular9+.
2. Comprobar el conocimiento en desarrollo web en BackEnd/NodeJS.
3. Comprobar conocimiento en el flujo completo de información con tecnologías web.
4. Comprobar capacidad de aprendizaje e iniciativa para obtener capacidades requeridas.

ESPECIFICACIONES

Se desea simular el flujo completo de la información en un sistema genérico de telemetría el cual consiste en enviar la información recabada por los sensores a la nube, que estos datos sean almacenados en una base de datos, que los usuarios finales mediante una plataforma web puedan tener acceso a la información histórica de estos sensores mediante gráficas y a su vez se actualicen en tiempo real conforme los datos de los sensores son enviados a la nube.

Para lograr esta tarea se ha pensado usar las siguientes tecnologías:

Frontend

- Angular 9+ como framework de la webapp.
- Angular Material 9+ como framework UI para la webapp.
- Chartjs como librería para generar gráficas.
- Socket io (frontend library) como librería de comunicación bidireccional con la nube.

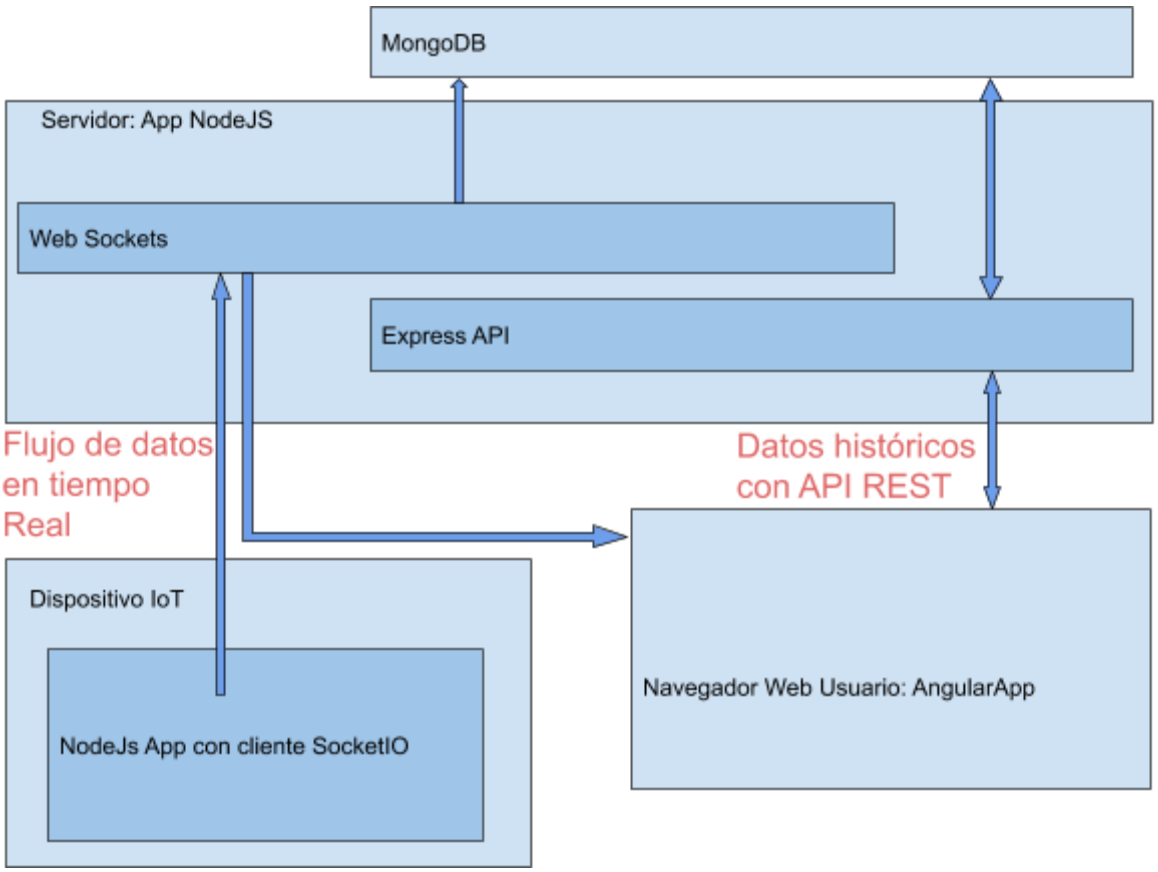
Backend

- Nodejs como motor de los servicios web.

- Socket io (Nodejs library) como librería de comunicación bidireccional en el backend con los clientes web.
- Express para los servicios REST que se requieran.
- MongoDB o alguna otra Base de Datos SQL o NoSQL para almacenar los datos que llegan del dispositivo y para su posterior consulta mediante la api rest en Express.

Simulación de Dispositivo

- Nodejs como motor de procesamiento.
- Socket io client (Node JS library for clients) como librería de comunicación bidireccional para comunicarse con el servidor socket.io del Backend.
- Puedes descargar el app de simulación de dispositivo desde el siguiente enlace: https://github.com/Mecabot/iotDeviceSimulator.git
- Sólo se requiere configurar el código del simulador para alcanzar la ip/url con el puerto apropiado.



Funcionamiento

- El código de simulación de dispositivo envía datos de dos sensores diferentes, uno de temperatura (rango [10,25]) y uno de humedad (rango [30,70]) cada 10 segundos.

- El servidor debe recibir estos datos y guardarlos en una base de datos para su posterior consulta.
- Los datos que llegan al servidor deben llegar de igual forma a los clientes web que estén conectados en ese momento al servidor socket io.
- Cuando se carga la página web se debe ver la vista de abajo con un default de 15 minutos de datos de los sensores.

- Es necesario poder realizar consultas de los valores históricos de diferentes tiempos, 15 minutos es el default, las otras opciones deben ser 30, 60, 120 y 300 minutos.
- Las gráficas se actualizan automáticamente conforme van recibiendo datos del dispositivo simulado.
- Para los componentes y layout de la UI se debe usar el framework de Angular Material.
- Cada gráfica debe ser una instancia de un Componente Angular de gráfica al que se le debe pasar un Objeto JSON “Sensor” con los atributos “nombre” y “unidades”, los cuales serán visualizados como se muestra en la imagen anterior. Además se deben incluir atributos que sean necesarios.
- Los datos de los sensores no son necesarios almacenarlos en una DB, pueden ser “hardcodeados” en el código del Front de acuerdo a los datos del script de sensores.
- La suscripción a los topics de datos de cada Componente gráfica puede realizarse ya sea por componente o a nivel servicio, dependiendo cómo se considere mejor.

Notas Finales

- Es deseable que la solución se monte en alguna nube y se pueda acceder a ella desde cualquier sitio con una ip o dominio, sin embargo lo necesario es solo que el flujo se pueda comprobar incluso ejecutandolo en una máquina local.
- Lo que se califica en esta prueba es el conocimiento y capacidad del desarrollador en soluciones FullStack JS.
- Para esta prueba no es necesario considerar puntos de seguridad como credenciales para la api rest o el broker socket.io sin embargo es un plus mencionar en los entregables qué problemas de seguridad deben ser considerados y cómo se pueden solventar.

Entregables

- Código fuente para análisis.
- Accesos para la plataforma si se ha deployado en la nube y son requeridos.
- Instrucciones para ejecutar el proyecto si no se ha deployado en la nube.
- Cualquier material que se considere necesario para que el proyecto sea ejecutado y de esta forma comprobar su funcionamiento.
- Documento con información extra si es necesaria.

Contacto

José de Jesús Dzib, jdzibs@elektra.com.mx