RESUMEN

Se busca desarrollar una plataforma web que permita la recepción de datos de sensores y su visualización en gráficos dinámicos en tiempo real.

OBJETIVOS

1. Evaluar competencias en desarrollo web Front-End, específicamente con Angular20.
2. Evaluar el dominio en desarrollo web del lado del servidor empleando Node.js y manejo de diferentes librerias.
3. Evaluar conocimientos en el flujo integral de información en aplicaciones web, incluyendo Front-End, Back-End y comunicación entre ambos.
4. Evaluar la disposición para el aprendizaje continuo y la proactividad en el desarrollo de las competencias necesarias.

ESPECIFICACIONES

Se busca simular el flujo completo de información en un sistema genérico de monitoreo. El sistema debe incluir un backend desarrollado con Node.js que genere datos simulados de temperatura y humedad, y los envíe en tiempo real mediante WebSockets. Estos datos deben almacenarse en una base de datos local, y además deben estar disponibles a través de endpoints para su consulta. Finalmente, los datos deben visualizarse en una plataforma web mediante gráficas en tiempo real y con acceso al historial de mediciones.

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

Frontend

- Angular 20 como framework de la aplicación.
    - Agregar los modulos
- Angular Material como gramework UI para la aplicación.
- Chartjs como librería para generar graficas
    - Para visualizar la gráfica en tiempo real, puedes apoyarte en las funciones proporcionadas por la API de Chart.js, disponibles en la documentación oficial: https://www.chartjs.org/docs/latest/developers/api.html. En particular, se recomienda utilizar métodos como chart.update() o manipular directamente los datasets con chart.data.datasets para agregar nuevos datos dinámicamente.

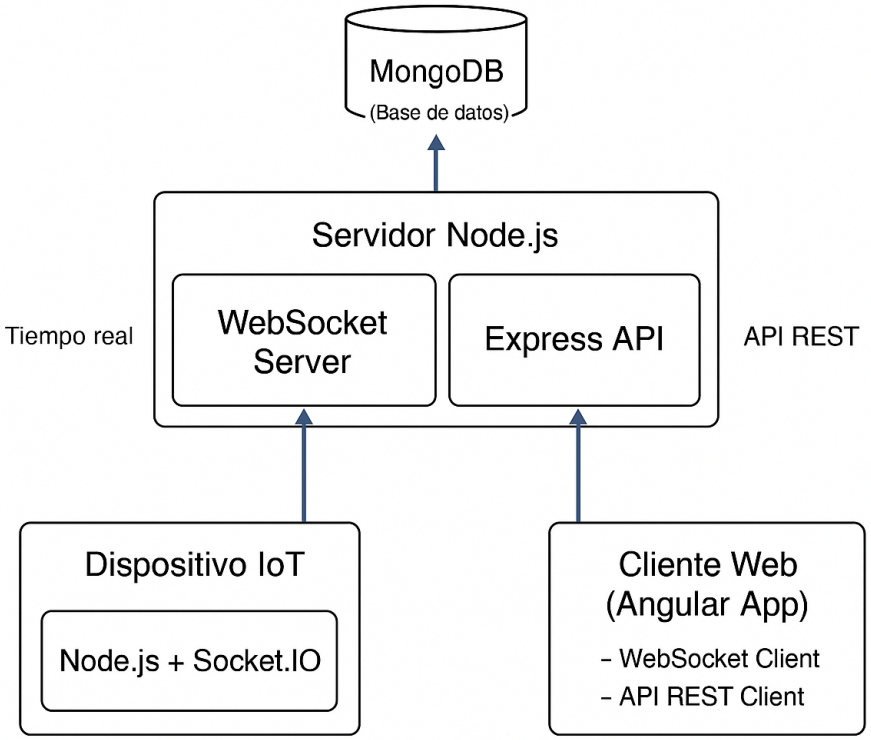
- Socket io (frontend library) como librería de comunicación bidireccional con la nube
    - Recuerda consultar la documentación oficial para configurar correctamente Socket.IO en el cliente: https://socket.io/docs/v4/client-initialization.

Backend

- Nodejs como motor de los servicios web.
- Socket io (Nodejs library) como librería de comunicación bidireccional en el backend.
    - Para implementar esta parte, es necesario que revises la documentación oficial de Socket.IO para la configuración del servidor: https://socket.io/docs/v4/server-initialization.
- 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

- Node.js se empleará como la plataforma de ejecución encargada del procesamiento y manejo de la lógica del sistema.
- La librería Socket.IO Client de Node.js será utilizada para habilitar una conexión bidireccional en tiempo real con el servidor Socket.IO implementado en el backend.
- 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.
- 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.

Ejemplo de visualizacion

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.
- Instrucciones para ejecutar el proyecto.
- 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.
- (Opcional) Video demo de 2-3 min explicando flujos clave.

CONTACTO

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

Edgar Roberto Martinez Mendoza, edgar.martinezm@elektra.com.mx