Esercizio: Dashboard Sensori di Temperatura

Obiettivo

Creare una dashboard per la visualizzazione dei dati di temperatura. In allegato

1. temperature-dashboard-example.png - esempio UI (non deve essere uguale, utilizzando l'ulitma versione di angular material può variare qualcosa).
2. generate-sensor-data.js.txt Script generazione automatica dei dati necessari (ATTENZIONE, per eseguire lo script, è necessario eseguirlo come js).

Descrizione

Il candidato dovrà implementare una dashboard che permetta di:

1. Filtrare i dati per intervallo temporale
2. Scegliere uno specifico sensore dalla lista disponibile
3. Visualizzare i dati in un grafico interattivo (i dati relativi al sensore e al range selezionati)
4. Gestire lo stato di caricamento durante il recupero dei dati

Layout Generale

La dashboard deve essere organizzata con:

- **Pannello principale (destra)**: Visualizzazione del grafico
- **Pannello filtri (sinistra)**: Form con controlli per data e sensore
- Layout responsive che si adatta ai dispositivi mobile

Tecnologie Richieste

- **Angular 20** (standalone components)
- **Angular Material** per i componenti UI
- **Plotly.js** per la visualizzazione dei grafici
- **RxJS** per la gestione reattiva dei dati
- **Tailwind CSS** (opzionale, a discrezione del candidato)

Struttura dei Dati

Interfacce TypeScript da implementare:

```
export interface SensorDataPoint {
  value: number;
  timestamp: string;
}

export interface SensorDataset {
  sensor: string;
  data: SensorDataPoint[];
}

export interface ApiResponse {
  sensors: SensorDataset[];
```

```
  timeRange: {
    earliest: string;
    latest: string;
  };
}
```

Requisiti Funzionali

1. Form di Filtro (Pannello Destro)

- **Selezione Periodo**: Range picker per data inizio e fine
  - **Validazione**: Entrambi i campi data devono essere compilati per rendere il form valido
- **Impostazione default: un mese da oggi (da oggi a 30 giorni fa)**:
  - Se il timerange ricevuto dall'api supera un mese, preselezionare automaticamente l'ultimo mese (dal dato più recente indietro di 30 giorni) in fase di inizializzazione
  - L'utente potrà successivamente modificare il range per visualizzare periodi più ampi
- **Selezione Sensore**: Dropdown con i sensori disponibili
  - Lista popolata dai dati ricevuti dall'API
  - **Auto-selezione del primo sensore disponibile al caricamento**

2. Grafico (Pannello Principale)

- Visualizzazione con Plotly.js
- Punti collegati da linee (mode: 'lines+markers')
- Asse X: timestamp (formato data/ora)
- Asse Y: temperatura in °C
- Responsive e ridimensionabile
- **Aggiornamento automatico quando tutti i campi del form sono compilati e validi**
- Il grafico si aggiorna sia durante l'inizializzazione che quando l'utente modifica i filtri

3. Gestione Stati

- **Loading**: Spinner durante il caricamento dati dall'API
- **Empty State**: Messaggio quando non ci sono dati per i filtri selezionati (snackbar)
- **Error Handling**: Gestione degli errori derivanti dall'API che non risponde o restituisce errori (visualizzazione tramite snackbar)

4. Comportamento Reattivo

- **Chiamata API** all'inizializzazione per recuperare tutti i dati disponibili
- **Filtro locale**: Dopo il caricamento iniziale, i filtri per sensore e range temporale funzionano sui dati già in memoria (senza nuove chiamate API)
- **Validazione form**: Il filtro viene eseguito solo se il form è valido (entrambe le date compilate e sensore selezionato)
- Se il form è invalido, il grafico non viene aggiornato

Generazione Dati Mock

Script di Generazione Dati

È fornito uno script JavaScript separato (`generate-sensor-data.js`) che permette di generare un file JSON con tutti i dati necessari per l'esercizio.

Utilizzo dello script:

```
node generate-sensor-data.js [numero_mesi] <--- input dinamico per test funzionalità
mesi
```

Lo script genera:

- Dati per 4 sensori: `Sensor_1` , `Sensor_2` , `Sensor_3` , `Sensor_7`
- Frequenza: Un punto dati ogni 2-4 ore
- Temperatura: Range 10-25°C con picchi occasionali fino a 30°C
- Pattern realistici: Variazioni giornaliere e per sensore
- Output: File `sensor-data.json` da utilizzare nell'applicazione

Implementazione nel Componente

Il candidato dovrà:

1. Eseguire lo script per generare `sensor-data.json`
2. Caricare il JSON nell'applicazione Angular (o in alternativa utilizzare un json-server)
3. Implementare un servizio che simuli una chiamata API ritornando i dati dal JSON
4. Filtrare localmente i dati in base ai filtri selezionati

CSS/Styling:

- Utilizzare Flexbox per il layout
- Responsive design per mobile
- Angular material

Criteri di Valutazione

Funzionalità (40%)

- [ ] Form di filtro completamente funzionante
- [ ] Grafico interattivo con Plotly
- [ ] Gestione stati (loading, empty, error)
- [ ] Comportamento reattivo corretto
- [ ] Preselezione automatica e caricamento iniziale
- [ ] Filtro locale per selezione sensore

Codice (30%)

- [ ] Architettura Angular moderna (standalone components)
- [ ] Utilizzo corretto di RxJS
- [ ] Tipizzazione TypeScript completa

UI/UX (20%)

- [ ] Interfaccia simile all'esempio fornito
- [ ] Responsive design
- [ ] Feedback utente appropriato

Bonus (10%)

- [ ] Utilizzo di Tailwind CSS
- [ ] Ottimizzazioni performance
- [ ] Utilizzo Signals
- [ ] Customizzazione tema Angular Material

Tempo Stimato

2-3 ore per implementazione base 1-2 ore aggiuntive per ottimizzazioni e bonus

---

Buona fortuna!

Per domande o chiarimenti durante l'esercizio, non esitare a chiedere.