PROGETTO: APP METEO CON MAPPA E PAGINA DI DETTAGLIO

Classe 4ª – Sistemi e Reti
Anno scolastico: 2025–2026

Obiettivo del progetto

Realizzare un'applicazione web lato client che permetta di:

1. selezionare una regione italiana da una lista;
2. selezionare una provincia appartenente alla regione scelta;
3. visualizzare su una mappa interattiva tutti i comuni della provincia selezionata, con un marker per ciascun comune;
4. per ogni comune, visualizzare informazioni meteo di base fornite dall'API Open-Meteo;
5. aprire una pagina di dettaglio per un singolo comune, contenente le previsioni meteo per più giorni.

Struttura dei file

La cartella del progetto deve contenere almeno:

```
/app-meteo
    index.html          (pagina principale con select e mappa)
    dettaglio.html      (pagina di dettaglio per un singolo comune)
    main.js             (logica della pagina principale)
    dettaglio.js        (logica della pagina di dettaglio)
    style.css           (stili grafici)
```

È possibile organizzare i file in sottocartelle (es. `/js`, `/css`), purché la struttura rimanga chiara.

Dati e API da utilizzare

1. Elenco dei comuni italiani

Si utilizza il file JSON pubblico mantenuto da Matteo Contrini.

Repository GitHub:
https://github.com/matteocontrini/comuni-json

File JSON con tutti i comuni (versione raw):
https://raw.githubusercontent.com/matteocontrini/comuni-json/master/comuni.json

Il file contiene un array di oggetti. Ogni oggetto rappresenta un comune, con campi simili ai seguenti:

```
{
  "nome": "Padova",
  "codiceCatastale": "G224",
  "regione": { "nome": "Veneto", "codice": "05" },
  "provincia": { "nome": "Padova", "sigla": "PD" }
}
```

2. API Open-Meteo – geocoding

Nota importante: il file non include i campi latitudine e longitudine. Questi sono reperibili dal seguente endpoint di open-meteo: https://geocoding-api.open-meteo.com/v1/search

Esempio di utilizzo:

```
https://geocoding-api.open-meteo.com/v1/search?name=Padova&count=1&language=it&format=json
```

Risposta:


```
[{"id":3171728,"name":"Padova","latitude":45.40797,"longitude":11.88586,"elevation":12.0,"feature_code":"PPLA2","country_code":"IT","admin1_id":3164604,"admin2_id":3171727,"admin3_id":6542281,"timezone":"Europe/Rome","population":203725,"country_id":3175395,"country":"Italia","admin1":"Veneto","admin2":"Padova","admin3":"Padova"}],"generationtime_ms":0.41544437}
```


Per il formato della richiesta e per il significato dei parametri utilizzabili, fate riferimento alla guida ufficiale:

https://open-meteo.com/en/docs/geocoding-api

Notate come sono disponibili provincia e regione del comune passato come parametro, questo vi permetterà di gestire casi di omonimia tra comuni.

---

3. API Open-Meteo – Forecast

Documentazione ufficiale:
https://open-meteo.com/en/docs

Endpoint principale per le previsioni:

```
https://api.open-meteo.com/v1/forecast
```

Parametri principali:

- latitude (obbligatorio)
- longitude (obbligatorio)
- current (per dati meteo attuali, ad esempio: `temperature_2m,wind_speed_10m`)
- daily (per previsioni giornaliere, ad esempio: `temperature_2m_max,temperature_2m_min,precipitation_sum`)
- timezone (si consiglia `timezone=auto`)

Esempio di chiamata per meteo attuale su Padova:

```
https://api.open-meteo.com/v1/forecast?latitude=45.4064&longitude=11.8768&current=temperature_2m,wind_speed_10m&timezone=auto
```

---

4. Leaflet e OpenStreetMap

Per la mappa interattiva si utilizza la libreria Leaflet.js con le tile di OpenStreetMap.

Documentazione Leaflet, Quick Start:
https://leafletjs.com/examples/quick-start/

Tile Usage Policy di OpenStreetMap (da leggere):
https://operations.osmfoundation.org/policies/tiles/

La mappa dovrà utilizzare il tile layer standard:

```
L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution: '&copy; OpenStreetMap contributors'
}).addTo(mappa);
```

Funzionalità richieste – Pagina principale (index.html + main.js)

1. Caricamento del file dei comuni

All'evento `DOMContentLoaded` la pagina deve:

1. eseguire una chiamata `fetch` al file `comuni.json` remoto;
2. convertire la risposta JSON in un array JavaScript (ad esempio `tuttiIComuni`);
3. conservare questo array in una variabile globale (o in una struttura dati) per l'uso successivo.

In caso di errore (file non raggiungibile, JSON non valido) la pagina deve mostrare un messaggio di errore all'utente.

---

2. Select Regione

Nella pagina principale deve essere presente una select per la scelta della regione:

```
<select id="selectRegione">
  <option value="">Seleziona una regione...</option>
</select>
```

Requisiti:

1. Dopo aver caricato `tuttiIComuni`, il codice deve:
   - estrarre l'elenco delle regioni dal campo `regione.nome`;
   - rimuovere i duplicati;
   - ordinare l'elenco in ordine alfabetico;
   - creare una `<option>` per ogni regione e aggiungerla alla select.
2. Quando l'utente cambia regione (evento `change`), deve essere aggiornata la select delle province (vedi punto seguente).

---

3. Select Provincia (dipendente dalla regione)

Seconda select:

```
<select id="selectProvincia">
  <option value="">Seleziona una provincia...</option>
```

```
</select>
```

Requisiti:

1. Quando l'utente seleziona una regione:
   - filtrare l'array `tuttiIComuni` per mantenere solo i comuni appartenenti a quella regione;
   - estrarre l'elenco delle province (si può usare ad esempio `comune.provincia.sigla` o `comune.provincia.nome`);
   - rimuovere i duplicati;
   - ordinare l'elenco;
   - popolare la select delle province con queste informazioni.
2. Quando l'utente seleziona una provincia:
   - ottenere l'elenco di tutti i comuni appartenenti a quella provincia (array `comuniProvincia`);
   - aggiornare di conseguenza la mappa.

---

4. Mappa con marker per i comuni della provincia selezionata

In `index.html` deve essere presente un contenitore per la mappa, ad esempio:

```
<div id="map" style="height: 500px;"></div>
```

Requisiti:

1. In `main.js` all'avvio inizializzare la mappa con Leaflet:
   - definire un oggetto mappa:
     `const mappa = L.map('map').setView([latIniziale, lonIniziale], zoomIniziale);`
     ad esempio centrando sull'Italia con zoom 6 o 7;
   - aggiungere il tile layer di OpenStreetMap come indicato al punto 3.3.
2. Quando viene selezionata una provincia:
   - rimuovere i marker precedenti (ad esempio utilizzando un `L.layerGroup` o mantenendo un array di marker e rimuovendoli);
   - per ogni comune in `comuniProvincia`:
     - leggere i campi `lat` e `lng` dal JSON;
     - creare un marker:
       `const marker = L.marker([comune.lat, comune.lng]);`
     - aggiungerlo alla mappa tramite il layer di marker;
     - associare un popup con almeno:
       - nome del comune,
       - sigla della provincia,
       - un link alla pagina di dettaglio (vedi oltre).
3. La mappa deve adattare la vista complessiva ai comuni della provincia, ad esempio utilizzando `mappa.fitBounds(bounds)` dove `bounds` è il riquadro minimo che contiene tutti i marker.

Esempio di link nel popup verso la pagina di dettaglio:

```
<a href="dettaglio.html?nome=Padova&lat=45.406435&lon=11.876761">
  Vai al dettaglio meteo
</a>
```

(Il link deve essere generato dinamicamente in JavaScript utilizzando i dati del comune.)

---

5. Dati meteo nei popup dei comuni (versione minima)

Requisito minimo:

- Il popup di ogni comune deve contenere almeno il nome del comune e la sigla della provincia.
- In aggiunta, è richiesto che per ciascun comune sia possibile visualizzare almeno la temperatura attuale attraverso l'API Open-Meteo.

Sono possibili due varianti:

1. Variante A (più semplice):
    - quando l'utente clicca sul marker, viene effettuata una `fetch` all'API Open-Meteo per quel comune (utilizzando `lat` e `lng`);
    - il popup viene aggiornato con i dati meteo ricevuti (es. temperatura, vento).
2. Variante B (più avanzata):
    - dopo la selezione della provincia, il codice effettua una serie di chiamate Open-Meteo (una per ciascun comune) e inserisce subito nel popup i dati meteo;
    - soluzione più completa ma più pesante in termini di richieste.

Funzionalità richieste – Pagina di dettaglio (dettaglio.html + dettaglio.js)

La pagina `dettaglio.html` viene aperta tramite un link di questo tipo:

```
dettaglio.html?nome=Padova&lat=45.406435&lon=11.876761
```

Requisiti:

1. All'avvio, `dettaglio.js` deve:
    - leggere i parametri della query string tramite `URLSearchParams`;
    - estrarre almeno: `nome`, `lat`, `lon`.
2. La pagina deve:
    - impostare il titolo, ad esempio:
      "Dettaglio meteo: Padova";
    - chiamare l'API Open-Meteo (`https://api.open-meteo.com/v1/forecast`) utilizzando i parametri `latitude` e `longitude` letti dalla query;
    - richiedere:
        - dati correnti (`current=temperature_2m,wind_speed_10m`);
        - dati giornalieri (`daily=temperature_2m_max,temperature_2m_min,precipitation_sum`);
        - `timezone=auto`.
3. Visualizzazione richiesta:

    a) Meteo attuale:

    - temperatura attuale in °C;
    - vento in km/h;
    - orario di rilevazione.

    b) Previsioni giornaliere:

    - tabella HTML o altra struttura con almeno 5 giorni;
    - colonne: data, temperatura massima, temperatura minima, precipitazioni totali del giorno.

Requisiti sul codice JavaScript

Dovrete mostrare di saper utilizzare:

- `fetch()` e la gestione delle Promises (`then` e/o `async/await`);
- `response.json()` per convertire le risposte in oggetti JavaScript;
- array e metodi di array (`filter`, `map`, `sort`, `forEach`);
- oggetti con proprietà annidate (es. `comune.regione.nome`, `comune.provincia.sigla`);
- manipolazione del DOM:
  - creazione dinamica di `<option>` per le select;
  - aggiornamento di contenuti testuali con `textContent/innerHTML`;
- eventi:
  - `DOMContentLoaded`;
  - `change` sulle select;
  - eventuali eventi associati ai marker.

Dovete inoltre utilizzare:

- `URLSearchParams` per la gestione dei parametri della query string;
- le principali funzioni di Leaflet (`L.map`, `L.tileLayer`, `L.marker`, `bindPopup`, `fitBounds`).

Il codice deve essere ragionevolmente commentato, spiegando a grandi linee il ruolo delle funzioni principali.

Relazione

Al termine del progetto, redigete una breve relazione i cui destinatari saranno i vostri compagni. In questa relazione dovrete spiegare, con estratti di codice e screenshot:

1. I metodi JS che avete utilizzato per gli array;
2. I metodi principali della libreria leaflet/openStreetMaps
3. La logica delle chiamate a servizi HTTP esterni (file comuni italiani o Open Meteo)

Riferimenti

- Open-Meteo – Weather Forecast API
  https://open-meteo.com/en/docs
- Open-Meteo - Geocoding API
- https://open-meteo.com/en/docs/geocoding-api
- Leaflet – Quick Start Guide
  https://leafletjs.com/examples/quick-start/
- OpenStreetMap – Tile Usage Policy
  https://operations.osmfoundation.org/policies/tiles/
- Comuni italiani JSON – Matteo Contrini
  https://github.com/matteocontrini/comuni-json
- Fetch() in javascript:
  https://www.html.it/pag/66525/fetch-api/

- Async/Await: cos'è e quando utilizzarlo
  https://cyberalchimista.it/async-await-javascript/

Valutazione

La valutazione terrà conto di:

1. Funzionalità
   - caricamento corretto del file `comuni.json`;
   - popolamento dinamico delle select Regione e Provincia;
   - visualizzazione dei comuni selezionati sulla mappa;
   - funzionamento dei marker;
   - apertura corretta della pagina di dettaglio con i parametri corretti;
   - visualizzazione di meteo attuale e previsioni giornaliere.
2. Qualità del codice
   - leggibilità e organizzazione;
   - uso appropriato di funzioni;
   - utilizzo corretto di fetch, array, DOM;
   - presenza di commenti essenziali.
3. Interfaccia utente
   - chiarezza e pulizia;
   - leggibilità dei dati;
   - eventuale cura grafica aggiuntiva.
4. Estensioni facoltative
   - icone meteo;
   - uso di framework css (es: Bootstrap, Tailwind)
   - più parametri meteo (umidità, copertura nuvolosa, ecc.);
   - selezione diretta del comune da una terza select;
   - Altro a vostro piacimento
5. Autovalutazione
   - Nel caso di lavoro svolto in gruppo, potrà essere somministrato un questionario individuale in cui vi verrà chiesto di valutare il grado in cui i vostri compagni sono stati partecipi all'attività.
6. Osservazione dei docenti durante lo svolgimento del progetto in laboratorio