科目名	年度	レポート番号	クラス	学籍番号	名前
API 実習	2024	5	A	20123020	五味帆翔

ページ数や文字数よりも、読んでわかりやすく書けているかどうかが、点数アップの分かれ目です。

API を使ったアプリやゲームが作ったけど「動きませんでした、完成しませんでした」は評価に値しません。 単位取得は、きちんと動くものが評価対象です。 API を使うこと、そしてプログラミングは 1 年生からの講義で学ぶことをすべて活用すれば実現できるはずです。

設問(1)

API とは、アプリケーション・プログラミング・インターフェースの略で、ソフトウェアと Web サイトをつなぐ役割を果たしている。つまり、異なるソフトウェア間で情報をやり取りする手段である。身近なもので例えると、レストランに行って注文するとき、メニューを見て注文することがある。ここで「メニュー」が API であり、注文内容（リクエスト）を伝えると、キッチン（サーバー）が料理を作り、注文したものを渡してくれる（レスポンス）。

API は、身近なものにも活用されている。例えば、Google マップがその一例である。地図や位置情報を表示する Google マップは、Google の API を使用している。例えば、レストランの住所を検索したり、現在地から目的地までの道順を確認する際に、API を通じてデータを取得し、それを地図に表示する仕組みである。

設問(2)

レポート(4)をもとに、API 連携作成または API を用いたサービス開発結果を書いてください。何かしら動くものが出来ている前提です。

名称

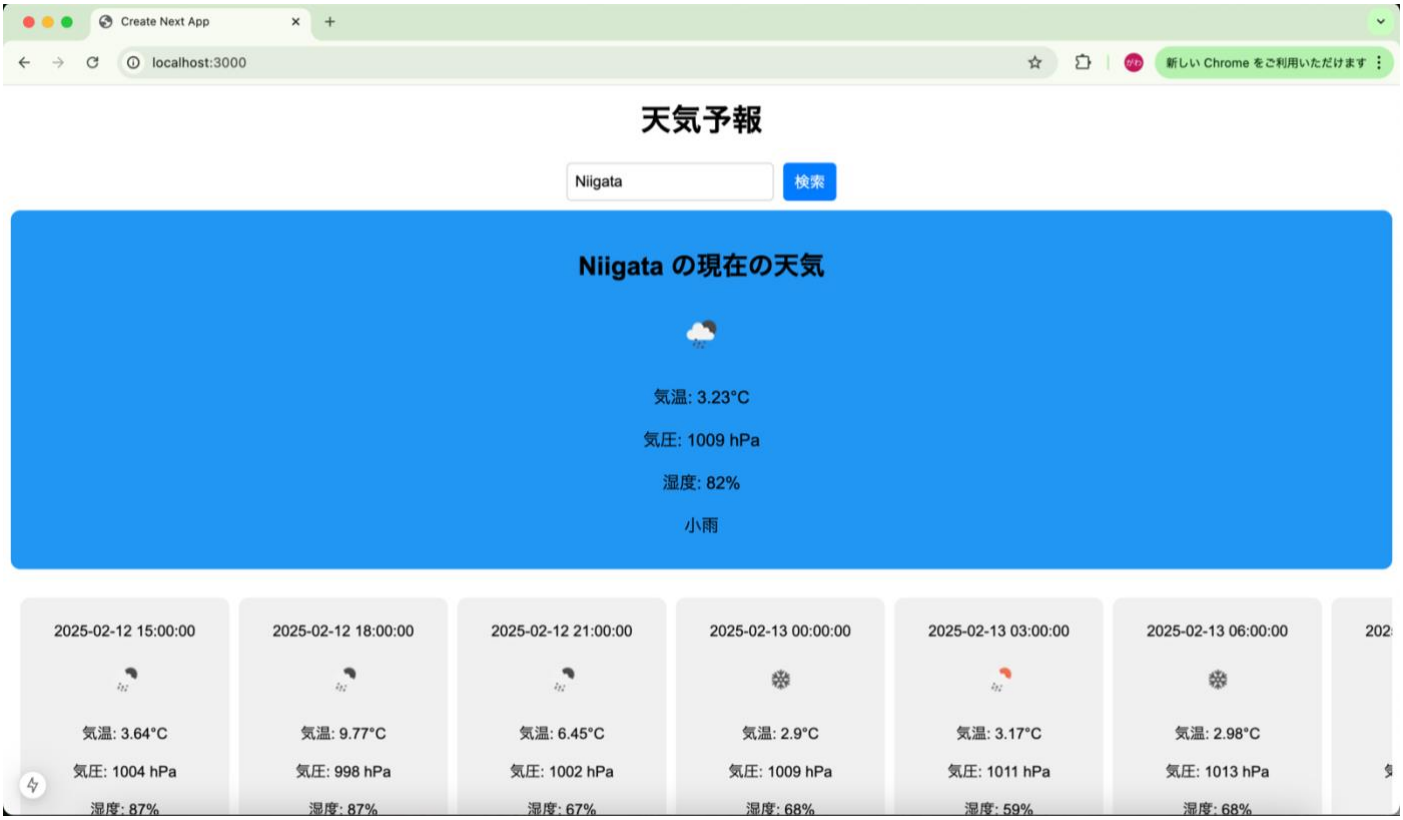
自己流天気予報アプリ

概要(作ったものの説明)

今回私が製作したアプリは天気予報アプリであり、従来の天気予報アプリとは異なる特徴があります。それは、気圧や湿度など、天気以外の詳細なデータも取得できる点です。私は自分が低気圧に弱いため、気圧の情報を取得できるようにし、また服の乾き具合を把握するために湿度の情報も表示できるようにしました。

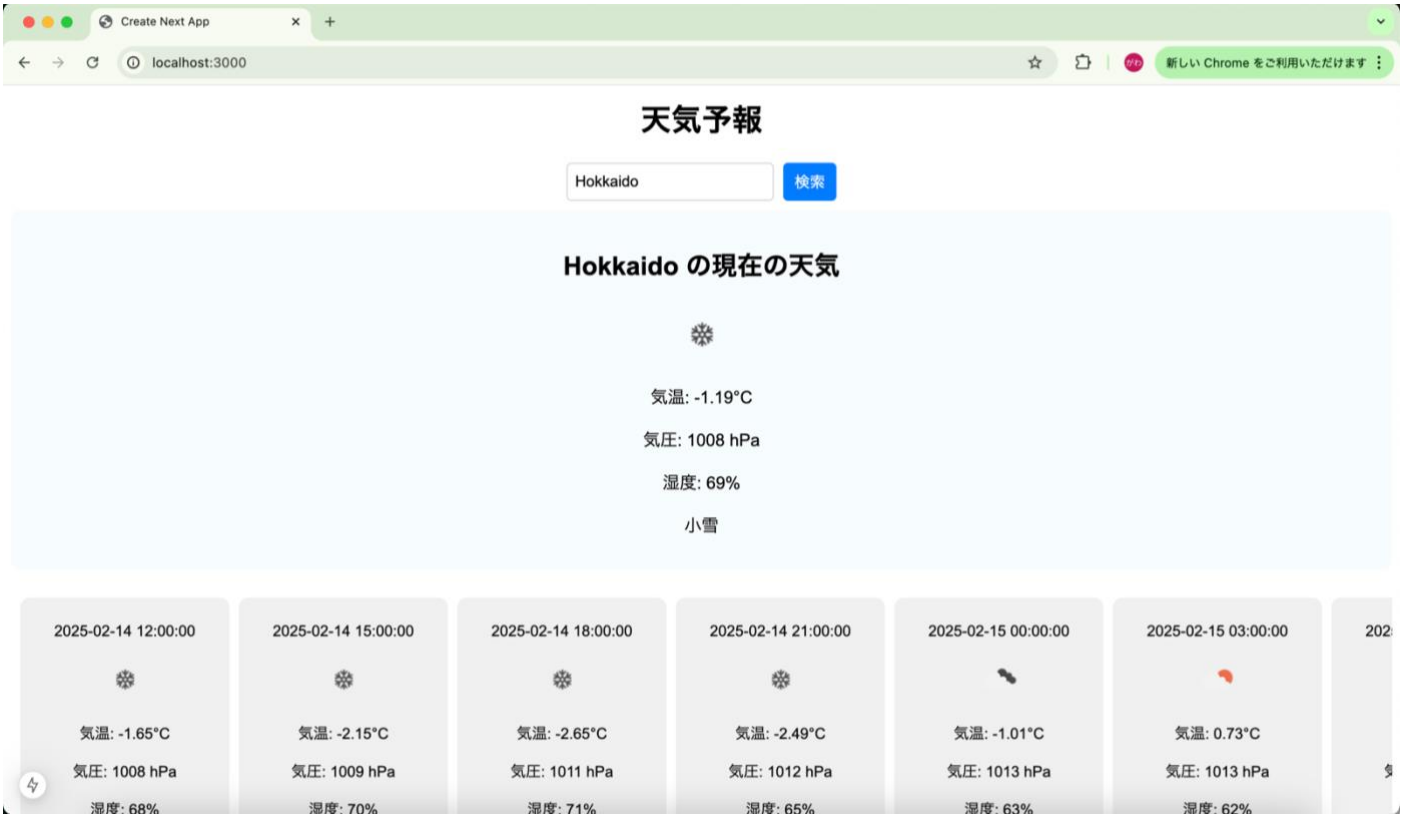
さらに、拡張性を考慮して、風速や UV などのデータも追加可能な設計にしています。このように、ユーザーのニーズに合わせて多角的な気象データを提供できる点が、他の天気予報アプリと差別化されるポイントです。

サービス説明(動作がわかるように画面を交えて説明すること)



上の画面はホーム画面である。

上の検索バーに知りたい都市の名前を入れることでその年の天気、気温、気圧、湿度、天気の説明を知ることができる。



上の画面は北海道を検索した時の画面である。

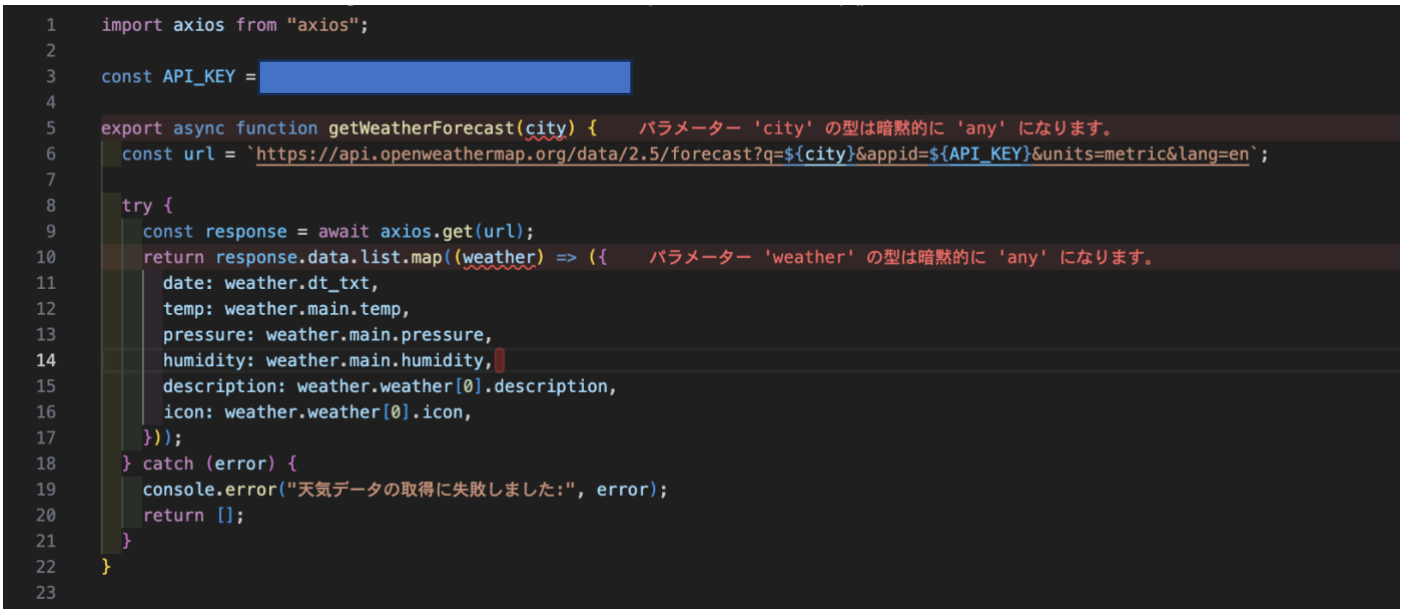
src > app > page.tsx > Home > fetchWeatherData

```tsx
"use client";
import { useEffect, useState } from "react";
import { getWeatherForecast } from "../lib/weather";
import "./globals.css";

export default function Home() {
  const [city, setCity] = useState("Tokyo");
  const [forecast, setForecast] = useState([]);
  const [currentWeather, setCurrentWeather] = useState(null);
  const [inputCity, setInputCity] = useState("");

  useEffect(() => {
    fetchWeatherData(city);
  }, [city]);

  async function fetchWeatherData(cityName: string) {
    const data = await getWeatherForecast(cityName);
    if (data.length > 0) {
      setCurrentWeather(data[0]);
      setForecast(data.slice(1, 9));
    }
  }

  const handleSearch = () => {
    if (inputCity.trim() !== "") {
      setCity(inputCity.trim());
    }
  };

  return (
    <div className="container">
      <h1>天気予報</h1>

      {/*  検索 */}
      <div className="search-bar">
        <input
          type="text"
          value={inputCity}
          onChange={(e) => setInputCity(e.target.value)}
          placeholder="都市名を入力 (例: Tokyo, Osaka)"
        />
        <button onClick={handleSearch}>検索</button>
      </div>

      {/*  現在の天気 */}
      {currentWeather && (
        <div className="current-weather">
          <h2>{city} の現在の天気</h2>
          <img src={`https://openweathermap.org/img/wn/${currentWeather.icon}.png`} alt={currentWeather.description} />
          <p>気温: {currentWeather.temp}°C</p>    プロパティ 'temp' は型 'never' に存在しません。
          <p>気圧: {currentWeather.pressure} hPa</p>    プロパティ 'pressure' は型 'never' に存在しません。
          <p>湿度: {currentWeather.humidity}%</p>    プロパティ 'humidity' は型 'never' に存在しません。
          <p>{currentWeather.description}</p>    プロパティ 'description' は型 'never' に存在しません。
        </div>
      )}

      {/*  未来の天気 */}
      <div className="weather-list">
        {forecast.map((weather, index) => (
          <div key={index} className="weather-card">
            <p>{weather.date}</p>    プロパティ 'date' は型 'never' に存在しません。
            <img src={`https://openweathermap.org/img/wn/${weather.icon}.png`} alt={weather.description} />    プロパティ
            <p>気温: {weather.temp}°C</p>    プロパティ 'temp' は型 'never' に存在しません。
            <p>気圧: {weather.pressure} hPa</p>    プロパティ 'pressure' は型 'never' に存在しません。
            <p>湿度: {weather.humidity}%</p>    プロパティ 'humidity' は型 'never' に存在しません。
            <p>{weather.description}</p>    プロパティ 'description' は型 'never' に存在しません。
          </div>
        ))}
      </div>
    </div>
  );
}
```

フロントエンドのコード

```css
.container {
  text-align: center;
  font-family: Arial, sans-serif;
}

.current-weather {
  background-color: #ffeb3b;
  padding: 15px;
  border-radius: 10px;
  margin-bottom: 20px;
  font-size: 18px;
}

.current-weather img {
  width: 60px;
  height: 60px;
}

.weather-list {
  display: flex;
  flex-wrap: nowrap;
  overflow-x: auto;
  padding: 10px;
  gap: 10px;
}

.weather-card {
  flex: 0 0 auto;
  width: 200px;
  padding: 10px;
  border-radius: 10px;
  background-color: #f0f0f0;
  text-align: center;
}

.weather-card img {
  width: 50px;
  height: 50px;
}

.search-bar {
  display: flex;
  justify-content: center;
  margin-bottom: 10px;
}

.search-bar input {
  padding: 8px;
  font-size: 16px;
```

CSS



```
import axios from "axios";

const API_KEY = [redacted]

export async function getWeatherForecast(city) {    パラメーター 'city' の型は暗黙的に 'any' になります。
  const url = `https://api.openweathermap.org/data/2.5/forecast?q=${city}&appid=${API_KEY}&units=metric&lang=en`;

  try {
    const response = await axios.get(url);
    return response.data.list.map((weather) => ({    パラメーター 'weather' の型は暗黙的に 'any' になります。
      date: weather.dt_txt,
      temp: weather.main.temp,
      pressure: weather.main.pressure,
      humidity: weather.main.humidity,
      description: weather.weather[0].description,
      icon: weather.weather[0].icon,
    }));
  } catch (error) {
    console.error("天気データの取得に失敗しました:", error);
    return [];
  }
}

```

バックエンドのコード

レポート(4)の記載内容の実現状況（原則 100%となること）

すべて実装済みであり、動作確認済みである

実際に動いていることがわかる動画の URL

https://kaishipu-my.sharepoint.com/:v:/g/personal/20123020_kaishi-pu_ac_jp/EcHPMrRl24tIvAMBUVb5IUkB0-fjaC8yXVMdpfQB6PtMFw?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&e=XCTJYI