@mostajs/weather — fiche LLM

> Module météo pour @mostajs : OpenWeatherMap, critères de volabilité (flyability), alertes, mode démo.

- Version: 0.1.1 · Licence: AGPL-3.0-or-later · Auteur: Dr Hamid MADANI drmdh@msn.com
- Chemin: mostajs/mosta-weather · Statut audit: complet (dist/)

RÔLE

Récupère la météo courante et les prévisions 5 jours via OpenWeatherMap, puis évalue si les conditions permettent un vol (drone/parapente) selon des critères configurables (vent max, visibilité min, conditions interdites). Sans clé API, bascule en mode démo avec données factices. Fournit aussi un composant React `WeatherPage` prêt à l'emploi.

INSTALLATION

npm i @mostajs/weather

EXPORTS

- fetchWeather(config) : function — météo courante + prévisions (ou mock si pas de clé)
- fetchWeatherByPlace(query, config?) : function — géocode via @mostajs/geo puis météo du 1er résultat
- resolveLabel(location, geo?) : function — complète le label par reverse-geocoding (@mostajs/geo)
- isFlyable(windKmh, condition, criteria?) : function — décide si le vol est autorisé
- msToKmh(ms) : function — conversion m/s → km/h
- getConditionIcon(condition) : function — emoji associé à une condition
- getMockWeather(config) : function — génère des données météo factices (démo)
- CONDITION_ICONS : Record<string,string> — table condition → emoji
- DEFAULT_CRITERIA : FlyabilityCriteria — critères de volabilité par défaut
- WeatherPage : composant React
- weatherModuleRegistration : objet de métadonnées d'enregistrement de module
- Types : WeatherConfig, FlyabilityCriteria, CurrentWeather, Forecast, WeatherAlert, WeatherData

API — SIGNATURES

- function fetchWeather(config: WeatherConfig): Promise

- function fetchWeatherByPlace(query: string, config?: PlaceWeatherConfig): Promise // PlaceWeatherConfig = Omit<WeatherConfig,'location'> & { geo?: GeoConfig }
- function resolveLabel(location: {lat;lon;label?}, geo?: GeoConfig): Promise
- function isFlyable(windKmh: number, condition: string, criteria?: FlyabilityCriteria): boolean
- function msToKmh(ms: number): number
- function getConditionIcon(condition: string): string
- function getMockWeather(config: WeatherConfig): WeatherData
- function WeatherPage(props: { config: WeatherConfig; onBack?: () => void; title?: string }): JSX.Element
- const weatherModuleRegistration: { name; label; description; version: string; priority: number }

TYPES CLÉS

- WeatherConfig { apiKey?: string; provider?: 'openweathermap'; location: { lat: number; lon: number; label?: string }; flyabilityCriteria?: FlyabilityCriteria; lang?: string; units?: 'metric'|'imperial' }
- FlyabilityCriteria { maxWindKmh: number; minVisibilityKm: number; forbiddenConditions: string[] }
- CurrentWeather { temp; humidity; wind; visibility: number; condition: string; icon: string }
- Forecast { day: string; temp; wind: number; condition: string; icon: string; flyable: boolean }
- WeatherAlert { type: string; message: string; severity: 'warn'|'danger' }
- WeatherData { location: string; lat; lon: number; current: CurrentWeather; forecast: Forecast[]; alerts: WeatherAlert[]; isRealData: boolean }

PATTERN

```
import { fetchWeather, isFlyable, msToKmh } from '@mostajs/weather';

const data = await fetchWeather({
  apiKey: process.env.OPENWEATHER_KEY,           // omis → mode démo
  location: { lat: 36.7538, lon: 3.0588, label: 'Alger' },
});
data.current.temp;          // 22
data.forecast[0].flyable;   // true
data.isRealData;            // false si mock

isFlyable(msToKmh(7.2), 'Clear'); // false (≈26 km/h > 25)
isFlyable(15, 'Rain');            // false (Rain interdit par défaut)
```

Composant : `<WeatherPage config={...} onBack={() => router.back()} />`

DÉPEND DE

- @mostajs/geo (^0.1.0) — géocodage pour fetchWeatherByPlace/resolveLabel (compose, ne réimplémente pas — DEVRULES §0).

- Peer dependency optionnelle : react (>=18) pour WeatherPage.
- fetchWeather classique (par {lat,lon}) reste autonome, sans appel à geo.

PIÈGES

- Sans `apiKey`, `fetchWeather` ne renvoie PAS d'erreur : il bascule silencieusement sur `getMockWeather` (vérifier `WeatherData.isRealData` pour distinguer).
- `provider` n'accepte actuellement que `'openweathermap'`.
- Les vitesses de vent sont exprimées en km/h dans toute l'API ; l'API OWM renvoyant des m/s, utiliser `msToKmh` avant `isFlyable`.
- `forbiddenConditions` compare les chaînes brutes OWM ('Rain', 'Snow', 'Thunderstorm'…).
- Le README mentionne « MIT » en pied de page, mais package.json fait foi : AGPL-3.0-or-later.

RÉFÉRENCES

- README.md
- CHANGELOG.md : absent
- docs/ (présent dans le module)