**Contrato de integração**

**1) Geocoding → cidade → coordenadas**

**Endpoint: https://geocoding-api.open-meteo.com/v1/search?name=<CIDADE>&count=1&language=pt&format=json**

**Entrada (servidor → API):**

* **name: string (ex.: "Goiânia")**
* **count: 1 (pegaremos o primeiro resultado)**
* **language: "pt"**
* **format: "json"**

**Saída (API → servidor) – campos que usaremos do results[0]:**

* **name: string normalizada da cidade**
* **country: string (país)**
* **latitude: number**
* **longitude: number**
* **timezone: string (ex.: "America/Sao\_Paulo")**

**Objeto interno que vamos guardar após o geocoding:**

**{**

**"city": {**

**"query": "<texto digitado>",**

**"name": "<nome normalizado>",**

**"country": "<país>",**

**"latitude": <num>,**

**"longitude": <num>,**

**"timezone": "<tz>"**

**}**

**}**

**Caso results venha vazio, retornaremos erro de “cidade não encontrada” (ver seção de erros).**

**2) Forecast diário → probabilidade de chuva**

**Endpoint: https://api.open-meteo.com/v1/forecast**

**Entrada (servidor → API):**

* **latitude: <num>**
* **longitude: <num>**
* **daily: precipitation\_probability\_max *(podemos acrescentar outras depois, mas por ora só essa)***
* **forecast\_days: 2 *(hoje + amanhã)***
* **timezone: "auto" *(para alinhar os dias ao fuso da cidade)***

**Saída (API → servidor) – campos que vamos ler:**

* **daily.time: string[] em ISO YYYY-MM-DD (ex.: ["2025-09-02","2025-09-03"])**
* **daily.precipitation\_probability\_max: number[] em % (alinhado pelo índice com daily.time)**
* **daily\_units.precipitation\_probability\_max: "%"**
* **(opcional, para metadados) timezone, elevation, etc.**

**Derivações que faremos:**

* **tomorrowISO: daily.time[1] *(como pedimos forecast\_days=2, o índice 1 é “amanhã” no fuso local)***
* **probTomorrow: daily.precipitation\_probability\_max[1] (number 0–100)**

**Objeto interno após o forecast:**

**{**

**"forecast": {**

**"date": "<YYYY-MM-DD>", // tomorrowISO**

**"precipitation\_probability\_max": <num> // probTomorrow (0-100)**

**}**

**}**

**Regra de decisão “vai chover amanhã?”**

**Entrada: probTomorrow (0–100)  
Parâmetro: THRESHOLD = 50 (%)**

**Decisão:**

* **willRain = (probTomorrow >= THRESHOLD)**

**Saída consolidada (o que enviaremos ao EJS):**

**{**

**"city": {**

**"query": "<texto digitado>",**

**"name": "<nome normalizado>",**

**"country": "<país>",**

**"latitude": <num>,**

**"longitude": <num>,**

**"timezone": "<tz>"**

**},**

**"forecast": {**

**"date": "<YYYY-MM-DD>",**

**"precipitation\_probability\_max": <num>,**

**"threshold": 50,**

**"willRain": true**

**},**

**"meta": {**

**"generatedAtISO": "<timestamp ISO do servidor>",**

**"source": "open-meteo"**

**}**

**}**

**Contrato de erros (para exibir algo amigável no EJS)**

**Quando falhar, retornaremos um único objeto error com:**

**{**

**"error": {**

**"code": "CITY\_NOT\_FOUND" | "FORECAST\_UNAVAILABLE" | "NETWORK\_ERROR" | "NO\_DATA\_TOMORROW",**

**"message": "<mensagem curta e clara>",**

**"stage": "geocoding" | "forecast"**

**}**

**}**

**Casos típicos:**

* **CITY\_NOT\_FOUND (geocoding sem resultados)**
* **FORECAST\_UNAVAILABLE (resposta sem daily ou sem precipitation\_probability\_max)**
* **NO\_DATA\_TOMORROW (array veio mas faltou índice 1)**
* **NETWORK\_ERROR (timeout/conexão/etc.)**

**Se estiver tudo de acordo, no próximo passo partimos para a configuração do projeto (etapa 3) e montamos os esboços das rotas/serviços respeitando esse contrato — só o esqueleto, sem lógica ainda, como você prefere.  
  
-------------------------------------------------------------------------------------------------------------------------------**