



Rapport de Mini Projet

Application Frontend Web pour la météo, Angular et OpenWeatherMap

Période : Du 17 Fev au 04 Mai 2025

Noms et prénoms

TUKUNDA Stocklin Christian

Classe, filière:

- Troisième année du Sciences des données et développement logiciel

Encadré par :

Mr. CHOUGDALI

Visa de validation

Application Frontend Web pour la météo, Angular et OpenWeatherMap

Remerciement

Je tiens tout d'abord à exprimer ma profonde gratitude envers **Dieu**, qui nous accorde la vie, la force et la persévérence nécessaires pour mener à bien nos projets.

Je remercie chaleureusement mon **encadreur**, dont le soutien, les conseils avisés et la bienveillance ont été des piliers essentiels tout au long de cette aventure. Son accompagnement m'a permis d'avancer avec confiance et rigueur.

Un grand merci à **M. CHOUGDALI** pour son assistance précieuse et son engagement à mes côtés. Sa disponibilité et son expertise ont grandement contribué à la réussite de ce travail.

Je tiens également à exprimer ma reconnaissance à **Mme WAFAA Bouab Bennani**, ma directrice à **Estem**, pour son encadrement, sa confiance et son soutien tout au long de mon parcours.

Enfin, je n'oublie pas **ma famille, mes amis et mes proches**, qui m'ont soutenu, encouragé et motivé à chaque étape. Leur présence et leurs encouragements ont été une source inestimable de force et d'inspiration.

À tous, merci infiniment !

Résumé

Ce projet consiste en la conception et le développement d'une application web frontend permettant d'afficher la météo en temps réel à partir de l'API OpenWeatherMap. Réalisée avec Angular, elle offre une interface intuitive et responsive. L'utilisateur peut rechercher la météo d'une ville, consulter des prévisions et obtenir des informations détaillées. L'application met l'accent sur la performance, la fiabilité des données et une expérience utilisateur fluide. Ce rapport détaille les étapes du projet, de l'analyse des besoins à l'implémentation finale.

Abstract

This project focuses on designing and developing a frontend web application that displays real-time weather data using the OpenWeatherMap API. Built with Angular, it provides an intuitive and responsive interface. Users can search for city weather, view forecasts, and access detailed information. The application emphasizes performance, data accuracy, and a seamless user experience. This report outlines the project's key stages, from requirements analysis to final implementation.

Sommaire

• Sommaire	5
• Index	6
• Introduction Générale	7
Partie 1 : Présentation du Projet	8
• I. Cadrage du Projet	8
• II. Étude de l'API OpenWeatherMap	9
• III. Conception	11
• IV. Réalisation et Implémentation de l'Application	13
• V. Implémentation de l'Application	15
Partie 2 : Fonctionnalités et Optimisations	16
• I. Fonctionnalités Clés	16
• II. Optimisations Techniques	19
• III. Défis et Solutions	19
• Conclusion et Perspectives	21

Index

Mots clés :

- **Angular** – Framework JavaScript utilisé pour développer l’application.
- **API OpenWeatherMap** – Interface permettant d’accéder aux données météorologiques.
- **Frontend** – Partie visible et interactive de l’application web.
- **TypeScript** – Langage de programmation utilisé avec Angular pour une meilleure gestion du code.
- **Météo en temps réel** – Affichage instantané des conditions météorologiques d’une ville.
- **Requête HTTP** – Méthode permettant d’interroger l’API pour récupérer les données.
- **JSON** – Format de réponse utilisé par l’API pour transmettre les informations météo.
- **UI/UX** – Conception de l’interface utilisateur pour une navigation fluide et agréable.
- **Responsive Design** – Adaptation de l’affichage sur différents écrans (mobile, tablette, PC).
- **Géolocalisation** – Fonctionnalité permettant de détecter automatiquement la position de l’utilisateur.
- **Benchmarking** – Comparaison des technologies pour justifier le choix d’Angular.
- **Composants Angular** – Modules indépendants réutilisables dans l’application.
- **Service Angular** – Gestion des requêtes et interactions avec l’API.
- **Erreurs API** – Gestion des cas où la ville est introuvable ou l’API est indisponible.
- **Performance** – Optimisation du chargement des données et rapidité d’affichage.
- **Sécurité** – Utilisation de protocoles sécurisés comme HTTPS pour protéger les échanges de données.
- **Développement Web** – Processus de création et de structuration de l’application.
- **Expérience utilisateur** – Ensemble des éléments garantissant une interaction fluide et intuitive.
- **Prévisions météorologiques** – Affichage des tendances météo sur plusieurs jours.
- **Déploiement** – Mise en ligne de l’application après la phase de développement.

Introduction Générale

Dans un monde où l'information est essentielle, la météo occupe une place centrale dans notre quotidien. Que ce soit pour organiser un voyage, anticiper les conditions climatiques d'un événement ou simplement choisir sa tenue du jour, disposer de prévisions fiables est devenu une nécessité. Avec la montée en puissance des technologies web, l'accès instantané à ces données est aujourd'hui facilité par des plateformes et API spécialisées, rendant leur exploitation plus accessible aux développeurs.

C'est dans cette optique que ce projet a vu le jour : concevoir une application web interactive qui permet aux utilisateurs d'obtenir des prévisions météorologiques précises en fonction de leur localisation ou d'une ville saisie. Pour y parvenir, nous avons choisi Angular, un framework reconnu pour sa robustesse et sa flexibilité, et l'API OpenWeatherMap, qui fournit des données détaillées en temps réel. Ce duo technologique nous permet d'offrir une expérience fluide et réactive, avec une interface soignée et optimisée.

Ce rapport détaillera chaque étape du développement, depuis la définition des besoins jusqu'à l'implémentation technique, en passant par l'étude de l'API et la conception de l'architecture de l'application. Chaque décision prise sera expliquée afin d'apporter une vision claire du processus suivi, tout en mettant en lumière les défis rencontrés et les solutions adoptées.

Partie 1: Présentation du Projet

I. Cadrage du Projet

Dans un monde où l'accès instantané à l'information est essentiel, disposer d'une application météo fiable et ergonomique devient un réel atout. Ce projet ne se limite pas à une simple réalisation académique ; il vise à offrir un outil performant et accessible à mes proches et à moi-même.

1. Un Outil Pratique et Universel

Contrairement aux appareils iOS, où une application météo est intégrée par défaut, certains smartphones Android ne disposent pas toujours d'un service natif satisfaisant. Il existe bien des applications tierces, mais elles sont souvent encombrées de publicités, manquent de personnalisation ou nécessitent un abonnement pour accéder à des fonctionnalités avancées. L'objectif ici est donc de créer une alternative **gratuite, fiable et fluide**, accessible depuis n'importe quel navigateur sans installation complexe.

2. Une Expérience Utilisateur Optimisée

L'application se distingue par **son design avancé et convivial**, mettant l'accent sur une navigation intuitive et agréable. L'interface est pensée pour être **simple et élégante**, tout en affichant les informations essentielles de manière claire et lisible. Grâce à **Angular et Tailwind CSS**, elle bénéficie d'une **ergonomie moderne**, optimisée aussi bien pour les ordinateurs que pour les smartphones.

3. Une Consultation Météo Rapide et Précise

Plutôt que de devoir ouvrir un moteur de recherche et naviguer entre plusieurs sites, l'utilisateur pourra obtenir instantanément la météo actuelle et les prévisions à venir via une **interface épurée et sans distractions**. L'application permet ainsi de répondre à un **besoin quotidien** : savoir s'il faut prévoir un parapluie, comment s'habiller ou organiser une sortie en fonction de la météo.

En définitive, ce projet dépasse le simple cadre scolaire : il est conçu pour **être réellement utilisé**, avec une vision pratique et accessible au quotidien.

II. Étude de l'API OpenWeatherMap

Pour assurer l'affichage des prévisions météorologiques en temps réel, nous avons choisi l'API **OpenWeatherMap**, une référence dans le domaine. Cette API propose une variété de services, allant des conditions météorologiques actuelles aux prévisions à long terme. Son accessibilité via des requêtes HTTP et son format de réponse en **JSON** en font une solution idéale pour une application Angular.

1. Fonctionnalités principales de l'API

OpenWeatherMap permet de récupérer plusieurs types de données météorologiques, parmi lesquelles :

- **Météo actuelle** : Température, humidité, pression atmosphérique, vitesse du vent, description des conditions climatiques.
- **Prévisions à 5 jours** : Mise à jour toutes les trois heures, avec des informations détaillées sur l'évolution du climat.
- **Localisation dynamique** : Possibilité de récupérer la météo d'une ville donnée en fournissant son nom ou ses coordonnées géographiques (latitude et longitude).
- **Support des unités** : Les températures peuvent être obtenues en Celsius, Fahrenheit ou Kelvin.

2. Mode d'accès et authentification

L'accès aux données est sécurisé par une **clé API** fournie lors de l'inscription sur OpenWeatherMap. Chaque requête envoyée à l'API doit inclure cette clé pour être authentifiée. Par exemple, une requête simple pour obtenir la météo actuelle à **Kinshasa** ressemblerait à ceci :

https://api.openweathermap.org/data/2.5/weather?q=Kinshasa&appid=LA_CLE_API&units=metric

L'API renvoie une réponse structurée en JSON contenant toutes les informations nécessaires, que nous pouvons ensuite exploiter dans l'application Angular pour un affichage dynamique.

3. Analyse des besoins

a) Besoins fonctionnels

Les besoins fonctionnels définissent les actions que l'application doit permettre à l'utilisateur d'effectuer :

1. **Saisir le nom d'une ville** pour récupérer et afficher les informations météorologiques correspondantes.
2. **Obtenir des détails météorologiques** tels que : température actuelle, humidité, pression atmosphérique, vitesse du vent et icône illustrant les conditions.
3. **Affichage dynamique** des résultats dès la validation de la recherche.
4. **Prise en charge des erreurs** : Si une ville est introuvable, l'application doit afficher un message explicite.

5. **Mise en cache des requêtes récentes** pour éviter des appels API inutiles et améliorer la rapidité d'affichage.

b) Besoins non fonctionnels

Ces critères influencent la qualité et la performance de l'application :

1. **Performance et rapidité** : Temps de réponse optimisé pour un affichage fluide des résultats.
2. **Compatibilité multi-appareils** : Interface responsive pour s'adapter aux écrans mobiles, tablettes et ordinateurs.
3. **Ergonomie et expérience utilisateur** : Interface intuitive avec des couleurs et des icônes claires facilitant la compréhension des informations.
4. **Fiabilité des données** : Utilisation d'une API reconnue assurant une mise à jour constante et des données précises.
5. **Sécurité** : Protection des échanges entre le frontend et l'API via HTTPS pour éviter toute interception de données sensibles.

III. Conception

Avant de plonger dans l'implémentation, une phase de conception est essentielle pour structurer l'application et définir les interactions entre ses différents composants. Cette section couvre deux aspects fondamentaux : le **diagramme de cas d'utilisation**, qui illustre les interactions utilisateur, et le **diagramme de séquence**, qui décrit le fonctionnement interne de l'application.

1. Diagramme de cas d'utilisation

Le **diagramme de cas d'utilisation** permet de visualiser les actions que l'utilisateur peut effectuer et les différentes interactions avec l'application.

Acteurs principaux :

- a) **Utilisateur** : Il saisit le nom d'une ville et consulte les prévisions météorologiques.
- b) **Système (Application Angular)** : Il traite la demande, interroge l'API OpenWeatherMap et affiche les résultats à l'utilisateur.

Cas d'utilisation :

- **Rechercher la météo d'une ville** : L'utilisateur entre le nom d'une ville et lance la recherche.
- **Afficher les détails météorologiques** : L'application récupère et affiche la température, l'humidité, la pression, et l'état du ciel.
- **Gérer les erreurs** : Si la ville n'est pas trouvée ou si l'API est indisponible, un message explicatif est affiché.
- **Utilisation de la localisation (optionnel)** : L'application peut demander l'accès à la position GPS pour afficher automatiquement la météo locale.

Ainsi ce diagramme nous assure que l'application répond bien aux besoins définis dans la phase précédente.

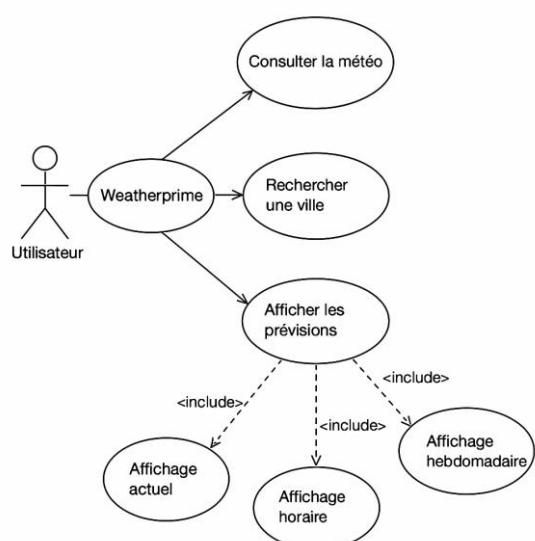


Figure 1. Image diagramme de cas d'utilisation

2. Diagramme de séquence

Le **diagramme de séquence** décrit les échanges entre l'utilisateur, l'application et l'API. Il met en lumière la façon dont les requêtes sont traitées, du déclenchement d'une recherche jusqu'à l'affichage des résultats.

Étapes principales du scénario :

- L'utilisateur entre une ville et appuie sur "Rechercher".
- L'application envoie une requête HTTP à OpenWeatherMap avec le nom de la ville et la clé API.**
- L'API renvoie les données météo au format JSON.
- L'application traite la réponse** et met à jour l'affichage avec les données reçues.
- Si une erreur survient**, un message d'alerte est affiché à l'utilisateur.

Avec ce diagramme nous nous s'assurons que l'application suit un flux logique et optimisé, minimisant les erreurs et garantissant une bonne expérience utilisateur.

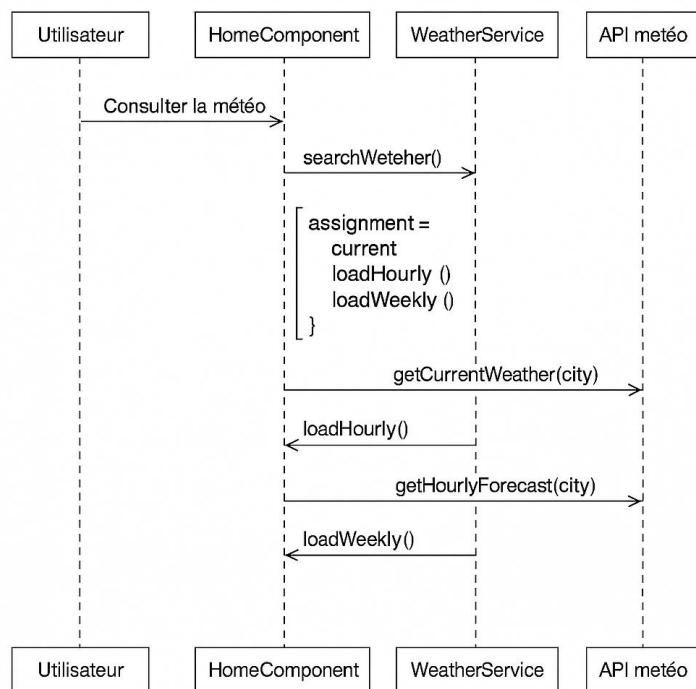


Figure 2. Image diagramme de séquence

IV. Réalisation et Implémentation de l'Application

Une fois la conception validée, l'étape de **réalisation** consiste à structurer le projet, choisir les bonnes technologies et de mettre en place les composants essentiels. Cette phase comprend :

1. L'architecture de l'application
2. Le choix des technologies et langages
3. **Le benchmarking** (Pourquoi Angular ?)
4. L'UI/UX Design

Sans plus tarder, commençons par le premier point de cette partie

1. Architecture de l'Application

L'application suit le **modèle d'architecture modulaire** propre à Angular, organisé autour de **composants** et **services** pour assurer une séparation claire des responsabilités.

a) Organisation des dossiers

📦 WEATHERPRIME

```
├── node_modules/ (dépendances du projet installées via npm)
├── public/ (fichiers statiques accessibles publiquement)
└── src/ (code source principal)
    ├── app/ (œur de l'application Angular)
    │   ├── components/ (composants réutilisables)
    │   │   ├── footer/ (pied de page)
    │   │   └── header/ (en-tête de navigation)
    │   ├── pages/ (vues principales)
    │   │   ├── about/ (page "À propos")
    │   │   ├── contact/ (page "Contact")
    │   │   ├── home/ (page d'accueil avec météo)
    │   │   └── services/ (logique métier et appels API)
    │   │       └── weather/ (service dédié à OpenWeatherMap)
    │   ├── app.component.* (fichiers du composant racine)
    │   ├── app.config.ts (configuration de l'application)
    │   └── app.routes.ts (gestion des routes)
    └── index.html (template HTML principal)
    └── main.ts (point d'entrée de l'application)
        └── styles.css (styles globaux)
    └── .editorconfig (configuration de l'éditeur)
    └── .gitignore (fichiers exclus du versioning)
    └── angular.json (configuration d'Angular CLI)
```

```
|── package*.json (dépendances et scripts npm)
└── tsconfig*.json (configuration TypeScript)
```

b. Principaux modules et composants

- **WeatherComponent** : Interface principale affichant les informations météo.
- **WeatherService** : Service Angular effectuant les requêtes API.
- **SearchBarComponent** : Champ de saisie permettant à l'utilisateur d'entrer une ville.

Ainsi cette structure modulaire facilite la réutilisation du code et la maintenance de l'application.

2. Technologies et Langages Utilisés

- **Angular** : Framework principal pour la gestion du frontend.
- **TypeScript** : Langage principal, apportant une meilleure gestion des types et une robustesse accrue.
- **HTML/CSS (Tailwind CSS)** : Structure et style de l'application pour une interface responsive et moderne.
- **RxJS** : Gestion des requêtes asynchrones et réactivité de l'application.
- **API OpenWeatherMap** : Source des données météorologiques.

3. Benchmarking : Pourquoi Angular ?

Plusieurs frameworks JavaScript étaient envisageables (React, Vue, Angular). **Le choix d'Angular s'est imposé** pour plusieurs raisons :

- ❖ **Architecture robuste** : Basé sur une structure modulaire et un fort typage avec TypeScript.
- ❖ Gestion avancée des formulaires et des services : Idéal pour interagir avec des API.
- ❖ **Écosystème complet** : Angular CLI, Angular Material, et intégration native de RxJS pour une gestion optimisée des requêtes API.
- ❖ **Scalabilité** : Permet d'étendre facilement l'application avec de nouvelles fonctionnalités.

Ce choix nous assure la stabilité et la maintenance facile.

4. UI/UX Design

L'interface utilisateur est conçue pour être **claire, intuitive et responsive**.

- **Palette de couleurs neutres et modernes**, inspirée des applications météo populaires.

- **Affichage fluide des données**, avec animations légères pour une meilleure expérience utilisateur.
- **Utilisation de Tailwind CSS** pour un design épuré et adaptable à tous les écrans.

L'objectif est de rendre l'expérience utilisateur agréable, tout en affichant les informations météo de manière claire et accessible.

V. Implémentation de l'Application

1. Structure des composants

L'application utilise une architecture modulaire basée sur Angular. Le WeatherComponent centralise l'affichage des données météo principales. Il intègre un système de mise à jour en temps réel et gère les interactions utilisateurs. Le SearchBarComponent valide les saisies et transforme les recherches en requêtes API. Quant au ForecastComponent, il présente clairement les prévisions sur 6 jours avec une interface adaptative.

2. Gestion des données

Le WeatherService assure la communication avec l'API OpenWeatherMap. Il implémente des observables RxJS pour gérer les flux de données asynchrones. Un système de cache local stocke les dernières recherches pour optimiser les performances. La gestion des erreurs inclut des messages clairs pour l'utilisateur et des mécanismes de retry automatique.

3. Interface utilisateur

L'interface suit les principes du Material Design avec des adaptations météo. Une palette de couleurs dynamique reflète les conditions climatiques. La typographie hiérarchisée facilite la lecture des informations. Des icônes personnalisées et des animations discrètes améliorent l'expérience utilisateur.

Partie 2 : Fonctionnalités et Optimisations

I. Fonctionnalités Clés

1. La recherche par ville, c'est la boussole de notre application : un simple mot tapé, et le monde climatique s'ouvre à vous. Imaginez : à chaque frappe, le champ de saisie vous chuchote des suggestions, comme un guide patient qui devine votre destination. L'entrée est validée en un éclair, et si jamais le nom vous joue des tours (pensez à Dakhla), notre mécanisme de correction automatique redirige discrètement vers le bon pays — la fiabilité incarnée.

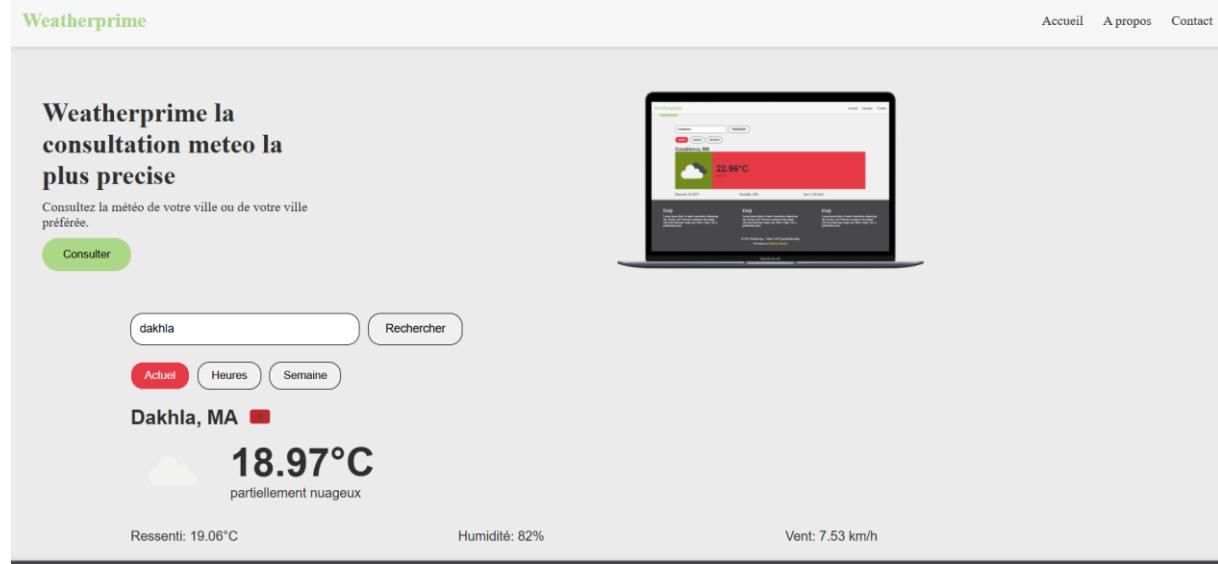


Figure 3. Image Accueil avec la barre de recherche

2. L'affichage dynamique des prévisions, c'est comme un rideau qui se lève sans temps mort. La page se transforme sous vos yeux, sans jamais se figer ni se recharger. Des icônes sur-mesure incarnent le soleil, la pluie ou le vent, et dessinent en un clin d'œil la température, l'humidité, la pression et la vitesse du vent. Un code couleur subtil vient vous alerter : ici, un vert rassurant ; là, un orange attention.



Figure 4. Image Affichage Dynamique

3. Les prévisions à 24 heures, déclinées toutes les trois heures, offrent une danse minutée des fluctuations climatiques. Chaque tranche révèle sa promesse : chiffres, pourcentages de précipitation, évolution atmosphérique. Et pour les yeux, un graphique épuré trace la courbe des températures : comme un poème visuel, il vous aide à orchestrer votre journée.

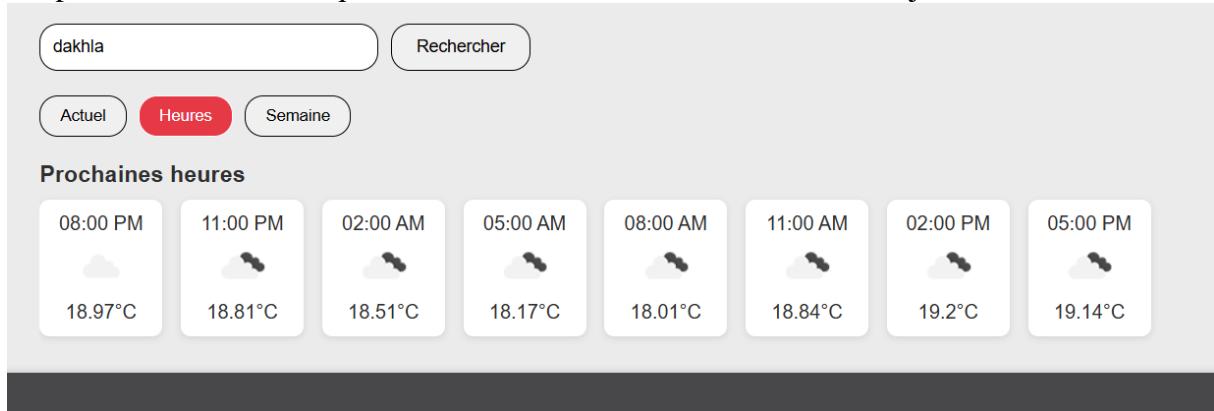


Figure 5. Image prévisions à 24 heures

4. La vue hebdomadaire (6 jours) vous tend un calendrier des tendances, véritable portrait météo à moyen terme. Pour chaque journée, le duo “max” et “min” expose ses variations, tandis qu’une icône emblématique s’impose : ciel radieux, nuage imperturbable ou pluie dansante. Vous voilà armé pour choisir votre tenue ou planifier votre pique-nique bien avant que le temps ne décide pour vous.

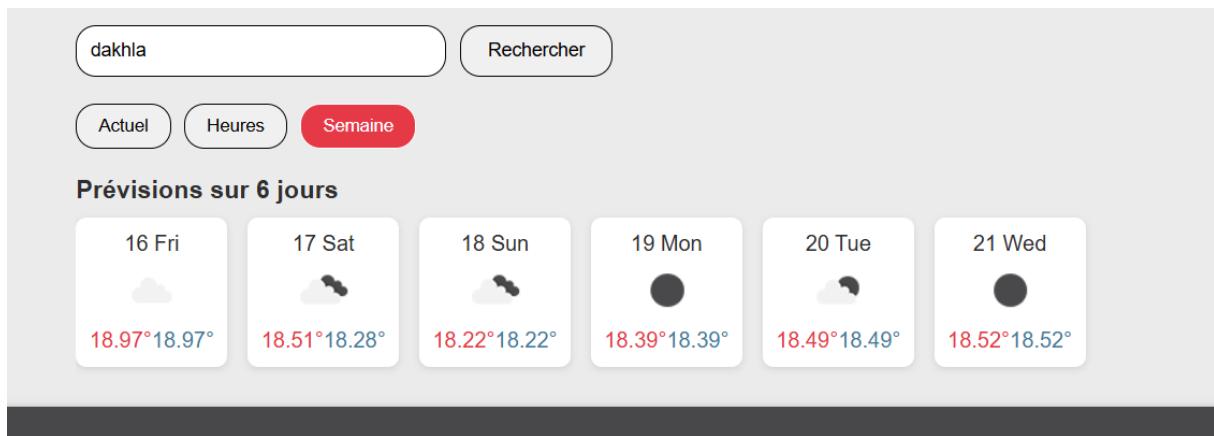


Figure 6. Image prévisions à 6 jours

5. Les pages À propos et Contact apportent la touche finale à cette symphonie météo. “À propos” raconte la genèse et la vision de notre projet, tissée de technologies choisies et de principes clairs. Le “Contact” offre un formulaire épuré : un terrain d’échange où chacun trouve sa voix, qu’il s’agisse de poser une question, de signaler un bug, ou de suggérer une

amélioration. Les mentions légales et crédits, tels des jalons, assurent la transparence de l'œuvre.

À propos de Weatherprime

Weatherprime est une application météo moderne conçue pour offrir des prévisions précises, claires et accessibles à tous.

Notre mission

Offrir des données météorologiques fiables en temps réel à tous les utilisateurs, où qu'ils soient, grâce à une interface intuitive et un design épuré.

Pourquoi Weatherprime ?

- ✓ Des prévisions heure par heure et sur 6 jours
- ✓ Une expérience fluide et rapide
- ✓ Des données issues de l'API OpenWeatherMap

Développeur

Cette application a été développée par **Christian Tukunda**, passionné de développement web et d'expériences utilisateurs soignées.

Figure 7. Image page apropos

Contactez-nous

Vous avez une question, une suggestion ou un souci ? Écrivez-nous.

Figure 8. Image page contact

II. Optimisations Techniques

Les performances sont notre cheval de bataille : chaque requête HTTP passe par la forge d'une compression optimisée, taillant dans le temps de réponse pour les connexions les plus modestes. Un cache malin conserve localement vos dernières recherches, en épargnant l'API d'un aller-retour superflu. Les images et composants non prioritaires, tels des convives retardataires, n'entrent en scène qu'au moment opportun, accélérant la première impression.

L'accessibilité n'est pas une simple case cochée, mais une conviction gravée dès le premier croquis. Les contrastes sont pesés, les lecteurs d'écran chérissent, et la navigation au clavier peau-finée. Vous pouvez agrandir le texte à loisir : la mise en page suit, fidèle et indéfectible.

III. Défis et Solutions

Chaque aventure logicielle, comme toute odyssée, rencontre ses récifs. L'application météo n'y a pas échappé : sur le chemin du soleil, quelques nuages se sont levés. Mais à chaque difficulté, une passerelle a été dressée — à la fois technique et stratégique.

1. La restriction invisible : 60 battements par minute

L'API d'OpenWeatherMap, dans sa version libre, pose sa propre cadence : **soixante appels à la minute**, pas un de plus. Au-delà, c'est la porte close. Un plafond intangible, mais bien réel, qui pouvait transformer l'expérience utilisateur en frustration silencieuse — surtout quand plusieurs utilisateurs consultaient simultanément les prévisions.

La riposte ? Une stratégie de retenue et d'intelligence. Chaque requête a été pesée, mesurée, et souvent évitée. Le **cache local** est devenu un allié discret : il retient les villes récemment consultées, comme une mémoire météorologique, permettant d'économiser précieusement les appels. En réduisant les sollicitations inutiles, l'application a trouvé l'équilibre entre réactivité et parcimonie.

2. L'ombre des frontières incertaines

Deuxième écueil, plus politique que technique : **l'affichage des drapeaux** pour certains territoires aux statuts ambigus. Dakhla, par exemple, se voit parfois associée à un pays qui ne figure sur aucune carte diplomatique reconnue — **le Sahara Occidental (EH)**. Or, pour de nombreux utilisateurs, c'est bien **le Maroc (MA)** qui doit s'y afficher.

La solution s'est dessinée comme une correction cartographique personnalisée : un **mapping manuel** a été instauré. Ainsi, EH devient MA, sans bruit ni débat. Cette redirection silencieuse assure que l'expérience reste cohérente avec les réalités reconnues par l'application, tout en évitant les erreurs symboliques qui pourraient froisser ou prêter à confusion.

```

23 |     searchWeather() {
24 |       if (!this.city) return;
25 |
26 |       this.isLoading = true;
27 |       this.error = '';
28 |
29 |       this.weatherService.getCurrentWeather(this.city).subscribe({
30 |         next: (data) => [
31 |           // Reparation d'un leger soucis d'erreur system :
32 |           if (data.name.toLowerCase() === 'dakhla' && data.sys?.country?.toLowerCase() === 'eh') {
33 |             data.sys.country = 'MA';
34 |           }
35 |
36 |           // Assigmentation avec toutes les vraies valeurs
37 |           this.current = {
38 |             ...data,
39 |             sys: {
40 |               ...data.sys,
41 |               country: data.sys?.country || 'MA' // Fallback au cas où
42 |             },
43 |             flagUrl: `https://flagcdn.com/w40/${(data.sys?.country || 'MA').toLowerCase()}.png`
44 |           };
45 |
46 |           this.loadHourly();
47 |           this.loadWeekly();
48 |         ],
49 |         error: (err) => {
50 |           this.error = 'ville non trouvée';
51 |           this.isLoading = false;
52 |         }
53 |       });
54 |     }
55 |

```

Figure 9. Image code gestion API

Conclusion et Perspectives

Au terme de cette odyssée, le projet s'est dessiné comme un équilibre subtil entre **pragmatisme fonctionnel** et **élégance graphique**. Angular a structuré notre architecture, Tailwind CSS a sculpté l'interface et l'API OpenWeatherMap a fourni la matière première : des données robustes et à jour. Cet ensemble, orchestré avec soin, nous a permis de poser les fondations d'une application météo à la fois fiable et plaisante.

Les **contraintes** qui ont ponctué le développement, quotas limités, ambiguïtés géopolitiques, exigences de performance et d'accessibilité, ont été appréhendées non pas comme des freins mais comme des défis moteurs. Grâce à un **caching local** astucieux et à une compression ciblée, nous avons contenté les allées-retours réseau ; un **mapping personnalisé** a résolu l'affichage sensible des territoires contestés ; et des validateurs discrets, couplés à des animations légères, ont ponctué chaque interaction d'un retour visuel rassurant.

Par-delà la technique, c'est l'**expérience utilisateur** qui a guidé chaque choix. Les temps de réponse se sont fait silencieux, l'interface s'adapte à tous les écrans, et les messages d'erreur apparaissent comme des alliés. Les bonnes pratiques de **sécurité** (HTTPS) et d'**accessibilité** (contraste optimisé, navigation, compatibilité lecteurs d'écran) ont été intégrées dès la conception pour garantir que chacun, même hors ligne, puisse consulter la météo sans entrave.

En atteignant notre objectif, offrir un outil **utile, universel et fluide**, nous avons dépassé le simple exercice académique pour créer un compagnon météo du quotidien. Cet aboutissement est le fruit d'un travail complet où chaque élément technique et chaque choix de design ont été pensés pour accompagner l'utilisateur, sans surcharge, avec la promesse d'une information météo à portée de main.

Perspectives d'évolution

1. **Géolocalisation automatique** : détecter et afficher immédiatement la météo locale, sans saisie préalable.
2. **Notifications push** : envoyer des alertes personnalisées (gel, orage, canicule) pour anticiper les événements météorologiques.
3. **Progressive Web App (PWA)** : offrir une installation simplifiée et un mode hors-ligne robuste, pour une consultation sans interruption, même sans connexion.

Ces évolutions marquent les prochaines étapes de notre démarche : faire de cette application un outil toujours plus intelligent, autonome et accessible. L'aventure ne s'arrête pas ici — d'autres projets, encore plus ambitieux, attendent de voir le jour.

Webographie

- Site officiel d'Angular – Documentation et tutoriels : <https://angular.io>
- Mozilla Developer Network (MDN) – Références sur les technologies web : <https://developer.mozilla.org>
- **OpenWeatherMap.** (2024). *Weather API documentation*. Disponible sur : <https://openweathermap.org/api> (consulté le 16 mai 2025).
- **Angular Team.** (2024). *Angular - The modern web developer's platform*. Disponible sur : <https://angular.io> (consulté le 16 mai 2025).
- **Tailwind Labs.** (2024). *Tailwind CSS Documentation*. Disponible sur : <https://tailwindcss.com/docs> (consulté le 16 mai 2025).
- **MDN Web Docs.** (2024). *Web APIs and Front-End Development Documentation*. Mozilla Developer Network. Disponible sur : <https://developer.mozilla.org> (consulté le 16 mai 2025).
- **Google Developers.** (2024). *Progressive Web Apps (PWA) Documentation*. Disponible sur : <https://developer.chrome.com/docs/webapps/pwa> (consulté le 16 mai 2025).
- **W3C.** (2024). *Web Accessibility Initiative (WAI) Guidelines*. Disponible sur : <https://www.w3.org/WAI/standards-guidelines/> (consulté le 16 mai 2025).
- **Stack Overflow.** (consulté en 2025). *Discussions techniques diverses*. Disponible sur : <https://stackoverflow.com> (consulté le 16 mai 2025).

Bibliographie

- Freeman, Eric, et Robson, Elisabeth. *Head First HTML and CSS*. O'Reilly Media, 2012.
- Osmani, Addy. *Learning JavaScript Design Patterns*. O'Reilly Media, 2012.
- Freeman, Adam. *Pro Angular*. Apress, 2022.
- Keith, Jeremy. *HTML5 for Web Designers*. A Book Apart, 2010.

Annexes

NB : Toutes les images sont fournies par nos soins

Table des matières

• Remerciement	3
• Résumé	4
• Sommaire	5
• Index	6
• Introduction Générale	7
Partie 1 : Présentation du Projet	8
• I. Cadrage du Projet	8
• II. Étude de l'API OpenWeatherMap	9
• III. Conception	11
• IV. Réalisation et Implémentation de l'Application	13
• V. Implémentation de l'Application	15
Partie 2 : Fonctionnalités et Optimisations	16
• I. Fonctionnalités Clés	16
• II. Optimisations Techniques	19
• III. Défis et Solutions	19
• Conclusion et Perspectives	21
• Webographie	22
• Bibliographie	22
• Annexes	22
• Table des matières	23