Projet Fil Rouge

Paul Serrano

Digital Campus

Titre Professionnel Concepteur de Solutions Digitales

Application web de géolocalisation, et d'informations météorologiques

Lien Github

Introduction

Dans un monde de plus en plus connecté, la géolocalisation joue un rôle crucial dans de nombreuses applications.

Mon projet vise à exploiter cette technologie pour créer un système de suivi en temps réel basé sur des données de géolocalisation.

À travers une interface intuitive et responsive, les utilisateurs pourront non seulement visualiser leurs localisation sur une carte interactive, mais également obtenir des informations météorologiques en temps réel, le tout dans le respect des normes de sécurité et de confidentialité des données.

Sommaire

Digital Campus

Sommaire

I. Analyse du Projet

Choix ergonomiques

Dans le cadre du projet de positionnement et d'informations météorologiques basés sur des données de géolocalisation, j'accorde une importance primordiale à l'ergonomie de l'interface utilisateur. Pour cela, j'ai sélectionné Angular comme framework front-end, permettant ainsi de bénéficier de ses fonctionnalités avancées de développement d'applications web dynamiques.

Voici comment ont été envisagé ces choix ergonomiques :

- Composants Angular modulaires

J'utilise les composants Angular pour décomposer l'interface utilisateur en éléments réutilisables. Par exemple, je crée des composants distincts pour la carte interactive, les informations météorologiques et la gestion des données utilisateur. Cela favorisera la modularité du code et simplifiera la maintenance.

I. Analyse du Projet

Choix ergonomiques

- Routing Angular pour une navigation fluide

Angular offre un système de routage puissant qui permet de gérer la navigation entre les différentes vues de manière fluide et intuitive. Je l'utiliserais pour définir des routes claires et cohérentes, facilitant ainsi la navigation des utilisateurs au sein de l'application.

- Intégration de Tailwind CSS pour un design élégant et responsive

J'utilise Tailwind CSS pour styliser notre interface utilisateur en mobile first, de manière efficace. Par exemple, j'applique des classes Tailwind pour définir la mise en page, les couleurs, les polices et d'autres éléments de design. Grâce à son approche basée sur les utilitaires, Tailwind CSS me permettra de personnaliser rapidement et facilement l'apparence de notre application, tout en garantissant une expérience utilisateur cohérente sur différents appareils.

I. Analyse du Projet

Digital Campus

Fonctionnalités

Notre système de localisation offrira une gamme de fonctionnalités robustes, exploitant pleinement les capacités de Python pour le back-end et Angular pour le front-end. Voici quelques exemples concrets de fonctionnalités que j'implémentarai :

- Visualisation des données de géolocalisation avec Leaflet :

Je me sers l'Api Leaflet pour intégrer une carte interactive dans notre interface utilisateur. Les utilisateurs pourront voir en temps réel leur position ainsi que leur historique de déplacements directement sur la carte.

- Intégration des informations météorologiques avec Python et Angular :

Grâce à l'API OpenWeather et à l'intégration côté serveur en Python, je pourrais récupérer les données météorologiques en temps réel et les afficher dans l'interface utilisateur Angular. Par exemple, seront affichées les prévisions météorologiques actuelles en fonction de la position de l'utilisateur sur la carte.

I. Analyse du Projet

Digital Campus

Fonctionnalités

- Enregistrement et consultation de l'historique des déplacements avec Python et MongoDB :

En utilisant Python pour le back-end, je stockerais les données de géolocalisation des utilisateurs dans une base de données NoSQL comme MongoDB. Les utilisateurs pourront ensuite consulter leur historique de déplacements via l'interface Angular, accédant ainsi aux détails de chaque trajet et aux conditions météorologiques associées.

- Sécurité renforcée avec Angular Guards et Python Flask :

Je vais mettre en place des fonctionnalités de sécurité robustes en utilisant les fonctionnalités de sécurité intégrées d'Angular, telles que les guards de routage, pour contrôler l'accès aux différentes parties de l'application. Côté serveur, je me servirai de frameworks comme Flask pour gérer l'authentification et l'autorisation des utilisateurs, en veillant à ce que seuls les utilisateurs autorisés puissent accéder aux données de géolocalisation et aux informations météorologiques.

- Google Sign-In :

Intégration de la fonctionnalité de connexion via Google Sign-In, permettant aux utilisateurs de s'authentifier à l'application à l'aide de leur compte Google.

II. Leviers d'action

Justification de la plateforme de déploiement choisie

Le choix de la plateforme de déploiement est une décision stratégique cruciale pour assurer le bon fonctionnement, la performance et la sécurité de notre système de suivi en temps réel. J'ai sélectionné une combinaison de technologies qui répondent à nos besoins spécifiques tout en offrant une flexibilité et une évolutivité optimales.

Voici les raisons pour lesquelles j'ai opté pour cette plateforme de déploiement :

- Python pour le back-end :

Python est un langage de programmation polyvalent et puissant qui offre une grande flexibilité et une vaste gamme de bibliothèques et de frameworks adaptés au développement web. J'ai choisi Python pour le back-end de notre application en raison de sa simplicité, de sa lisibilité du code et de sa large adoption dans la communauté du développement web.

II. Leviers d'action

Justification de la plateforme de déploiement choisie

- Angular pour le front-end :

Angular est un framework front-end moderne et robuste développé par Google. Il offre une architecture MVC (Modèle-Vue-Contrôleur) qui facilite le développement d'applications web complexes et interactives. J'ai choisi Angular pour sa performance, sa stabilité et sa capacité à gérer efficacement les interactions utilisateur dans notre application de suivi en temps réel.

- Tailwind CSS pour le design :

Tailwind CSS est un framework CSS basé sur des utilitaires qui permet de styliser rapidement et efficacement une application web. J'ai opté pour Tailwind CSS en raison de sa flexibilité, de sa facilité d'utilisation et de sa capacité à créer un design élégant et cohérent pour notre interface utilisateur. En utilisant Tailwind CSS, Je pourrais personnaliser facilement l'apparence de notre application tout en maintenant un code propre et maintenable.

II. Leviers d'action

Justification de la plateforme de déploiement choisie

- Bibliothèques Pythons utilisées :

Flask :
Framework léger, offrant des fonctionnalités de routage, de gestion des requêtes HTTP et de création d'API RESTful.

CORS :
Extension Flask pour gérer les requêtes Cross-Origin Resource Sharing et permettre le partage de ressources entre différentes origines.

Dotenv / Os :
Bibliothèques Python pour charger les variables d'environnement à partir d'un fichier .env, facilitant la gestion des configurations sensibles.

Requests :
Bibliothèque Python pour envoyer des requêtes HTTP, utilisée pour interagir avec des API externes et récupérer des données météorologiques en temps réel.

PyMongo :
Driver Python pour MongoDB, utilisé pour interagir avec la base de données NoSQL et stocker les données de géolocalisation des utilisateurs.

II. Leviers d'action

Justification de la plateforme de déploiement choisie

- Bibliothèques Angular utilisées :

Observer :
Design pattern de programmation utilisé dans Angular pour surveiller les changements dans les données et déclencher des mises à jour automatiques de l'interface utilisateur.

HttpClientModule :
Module Angular pour effectuer des requêtes HTTP vers un serveur distant et récupérer des données à partir d'API externes.

HttpClient :
Service Angular pour effectuer des requêtes HTTP asynchrones et gérer les réponses du serveur.

provideHttpClient :
Méthode Angular pour fournir une instance de HttpClient dans l'application.

withFetch :
Fonction Angular pour utiliser l'API Fetch JavaScript pour effectuer des requêtes HTTP.

II. Leviers d'action

Stratégie de communication

Une stratégie de communication efficace est essentielle pour promouvoir notre système de suivi en temps réel, attirer les utilisateurs potentiels et garantir une adoption réussie de la solution. J'ai élaboré une stratégie de communication complète qui met en valeur les avantages et les fonctionnalités de notre produit tout en ciblant les utilisateurs pertinents.

Voici les principaux éléments de notre stratégie de communication :

- Marketing ciblé :

J'identifierais les segments de marché clés pour notre solution de suivi en temps réel, en mettant l'accent sur les industries et les secteurs où la géolocalisation et la météo sont des facteurs critiques. L'accent sera mis sur des campagnes marketing ciblées pour atteindre ces segments de marché et promouvoir notre produit de manière efficace.

- Présence en ligne :

Les canaux de communication en ligne tels que les réseaux sociaux, les blogs, les forums de discussion et les communautés en ligne ferons la promotion de notre solution et fera le lien avec les utilisateurs potentiels. Je mettrais en place une stratégie de contenu engageante pour générer de l'intérêt et stimuler l'engagement autour de notre produit.

II. Leviers d'action

Stratégie de communication

- Partenariats stratégiques

Il faudra exploiter les opportunités de partenariats stratégiques avec d'autres entreprises et organisations qui complètent notre solution de suivi en temps réel. Par exemple, une collaboration avec des fournisseurs de services de géolocalisation, des entreprises de logistique ou des organismes météorologiques pourra enrichir notre offre et étendre notre portée sur le marché.

- Relations publiques :

Je travaillerai activement avec les médias et les journalistes pour obtenir une couverture médiatique positive et une exposition accrue pour notre solution. Nous organiserons des événements de presse, des conférences de presse et des entrevues avec des experts pour présenter notre produit et partager notre vision avec un public plus large.

En mettant en œuvre cette stratégie de communication intégrée, le but étant de positionner notre solution comme une référence dans son domaine, à susciter l'intérêt des utilisateurs potentiels et à favoriser une adoption réussie sur le marché.

III. Fonctionnalités techniques :

Digital Campus

Récupération des datas sur l’api OpenWeather

Dans cette présentation, nous allons explorer le processus de récupération des données météorologiques depuis l'API OpenWeather en utilisant Python pour le back-end, et comment ces données sont transmises efficacement au front-end Angular pour une intégration fluide dans notre application de suivi en temps réel.

- Utilisation de l'API OpenWeather :

L'API OpenWeather fournit des données météorologiques précises et en temps réel pour des emplacements spécifiques à travers le monde.

- Interaction avec l'API en Python :

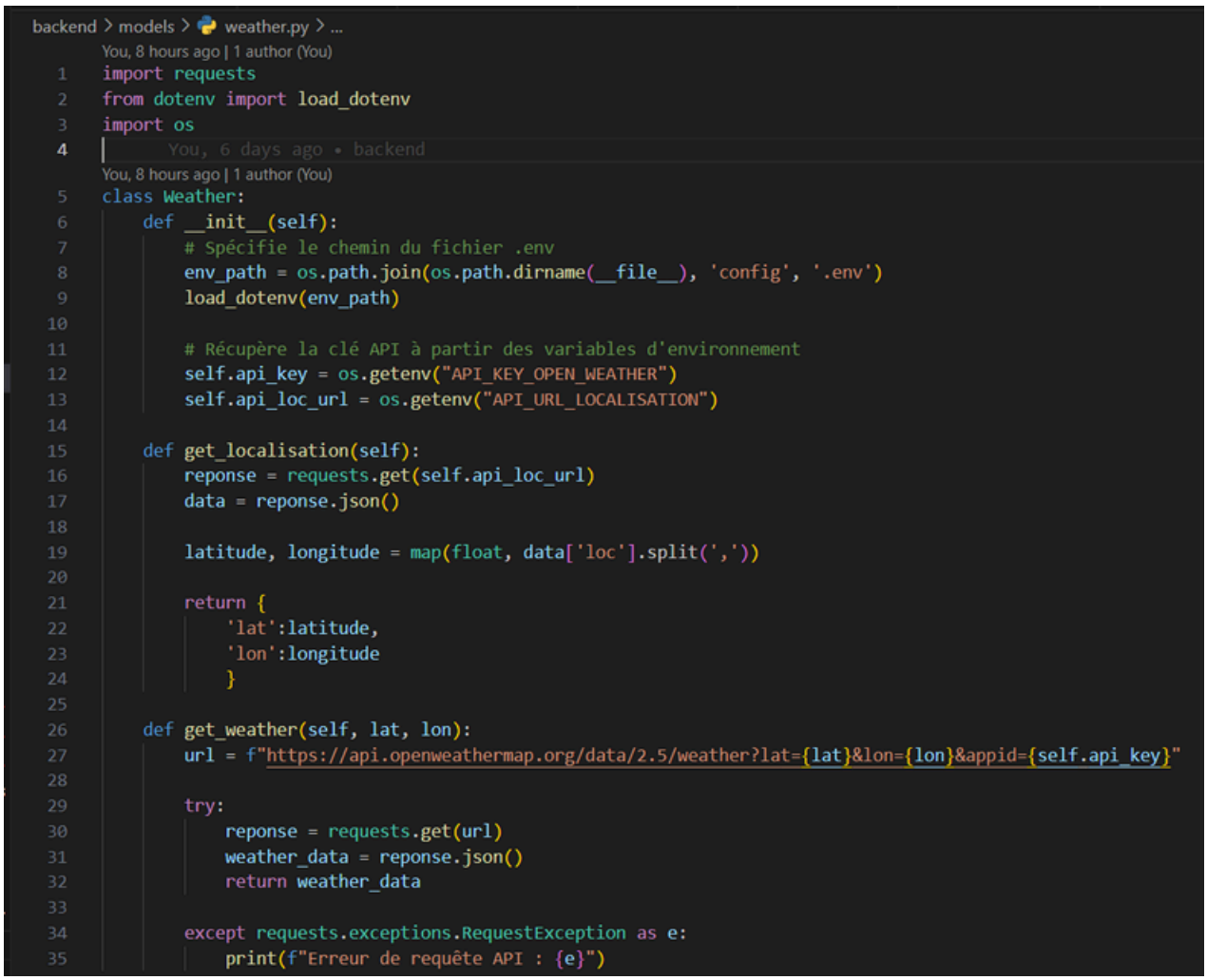
Utilisation de la bibliothèque requests pour envoyer des requêtes HTTP à l'API OpenWeather.
Configuration des paramètres de requête tels que l'emplacement géographique et les unités de mesure.

III. Fonctionnalités techniques :

Digital Campus

Récupération des datas sur l'api OpenWeather

- Exemple de code Python pour récupérer les données météorologiques :



backend > models > weather.py > ...

You, 8 hours ago | 1 author (You)

```python
import requests
from dotenv import load_dotenv
import os
```

You, 6 days ago • backend

You, 8 hours ago | 1 author (You)

```python
class Weather:
    def __init__(self):
        # Spécifie le chemin du fichier .env
        env_path = os.path.join(os.path.dirname(__file__), 'config', '.env')
        load_dotenv(env_path)

        # Récupère la clé API à partir des variables d'environnement
        self.api_key = os.getenv("API_KEY_OPEN_WEATHER")
        self.api_loc_url = os.getenv("API_URL_LOCALISATION")

    def get_localisation(self):
        reponse = requests.get(self.api_loc_url)
        data = reponse.json()

        latitude, longitude = map(float, data['loc'].split(','))

        return {
            'lat':latitude,
            'lon':longitude
            }

    def get_weather(self, lat, lon):
        url = f"https://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={self.api_key}"

        try:
            reponse = requests.get(url)
            weather_data = reponse.json()
            return weather_data

        except requests.exceptions.RequestException as e:
            print(f"Erreur de requête API : {e}")
```

III. Fonctionnalités techniques :

Transmission des données au front-end Angular :

- Utilisation de Flask pour créer une API RESTful :

Flask est utilisé comme framework back-end pour créer une API RESTful qui expose les données météorologiques récupérées.

- Endpoint d'API pour les données météorologiques :

Un endpoint est créé dans Flask pour recevoir les demandes du front-end Angular et renvoyer les données météorologiques.

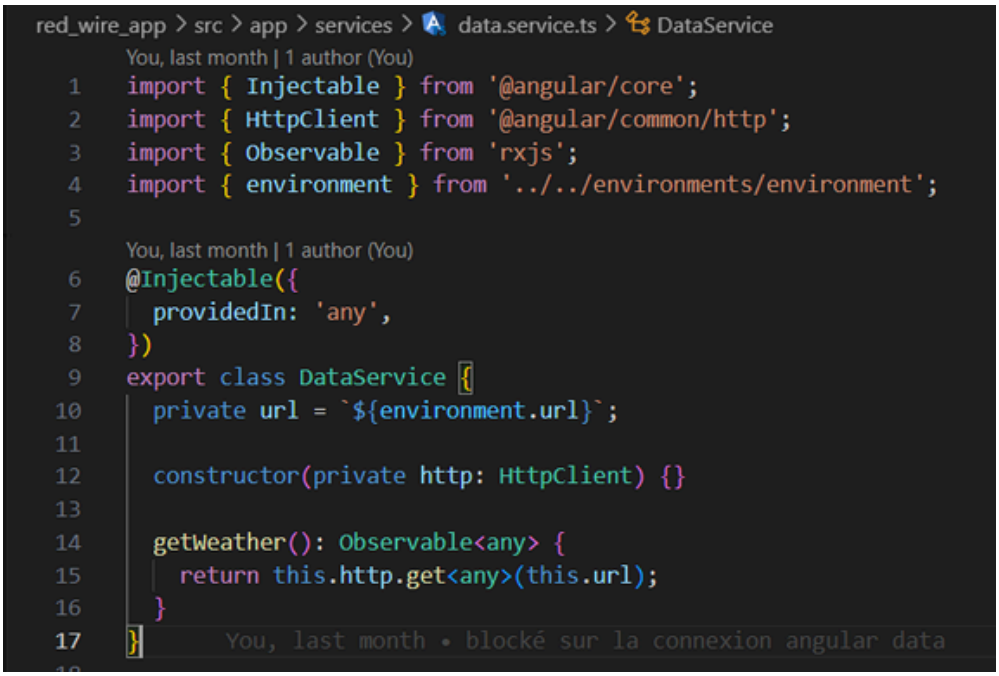
- Intégration de l'API dans Angular :

Utilisation du service HttpClient dans Angular pour envoyer des requêtes au back-end Flask et récupérer les données météorologiques.

III. Fonctionnalités techniques :

Transmission des données au front-end Angular :

- Exemple de code Angular pour récupérer les données de l'API Flask :



En combinant Python pour le back-end avec Flask et Angular pour le front-end, nous avons mis en place un processus efficace pour récupérer les données météorologiques depuis l'API OpenWeather et les transmettre au front-end Angular.
Cette approche garantit une intégration fluide des données météorologiques dans notre application de suivi en temps réel, offrant ainsi une expérience utilisateur enrichie et des fonctionnalités supplémentaires pour nos utilisateurs.

III. Fonctionnalités techniques :

Sécurité de la solution

La sécurité est une préoccupation majeure dans tout projet numérique, surtout lorsqu'il s'agit de données sensibles telles que la géolocalisation des utilisateurs et les données météorologiques. Dans cette section, nous aborderons les mesures de sécurité mises en place pour garantir la protection des données et la confidentialité des utilisateurs dans notre application de suivi en temps réel.

Protection des Données :

- Chiffrement des Données :

Toutes les données sensibles, y compris les données de géolocalisation des utilisateurs et les informations météorologiques, sont chiffrées lorsqu'elles sont stockées dans la base de données MongoDB. Cela garantit que les données sont protégées contre toute tentative d'accès non autorisé.

- Gestion des Identifiants d'API :

Les clés d'API utilisées pour accéder à des services tiers tels que l'API OpenWeather sont stockées de manière sécurisée dans des variables d'environnement à l'aide de la bibliothèque Dotenv. Ces identifiants sont strictement contrôlés et ne sont accessibles qu'aux processus autorisés.

Gestion de l'Authentification et de l'Accès :

- Authentification des Utilisateurs :

Tous les utilisateurs doivent s'authentifier avant d'accéder à l'application. Nous utilisons le service Google Sign-In pour gérer l'authentification des utilisateurs via leur compte Google. Cela garantit que seuls les utilisateurs autorisés peuvent accéder aux fonctionnalités de l'application.

III. Fonctionnalités techniques :

Sécurité de la solution

- Contrôle des Accès :

Une fois authentifiés, les utilisateurs sont soumis à des contrôles d'accès stricts pour limiter leur accès uniquement aux données et fonctionnalités qui leur sont autorisées. Par exemple, chaque utilisateur peut uniquement accéder à ses propres données de géolocalisation et à l'historique de ses déplacements.

Gestion des Sessions et des Cookies :

- Gestion des Sessions :

Les sessions utilisateur sont gérées de manière sécurisée pour garantir l'authenticité et l'intégrité des utilisateurs. Des mécanismes de gestion des sessions, tels que l'utilisation de tokens JWT (JSON Web Tokens), sont mis en place pour assurer une identification sécurisée des utilisateurs tout au long de leur session.

- Utilisation de Cookies Sécurisés :

Des cookies sécurisés sont utilisés pour stocker des informations d'authentification et de session côté client. Ces cookies sont configurés avec des attributs sécurisés tels que HttpOnly et Secure pour réduire les risques d'attaques XSS (Cross-Site Scripting) et CSRF (Cross-Site Request Forgery).

La sécurité est une priorité absolue dans notre application de suivi en temps réel. En mettant en œuvre des mesures de sécurité robustes telles que le chiffrement des données, l'authentification des utilisateurs et la gestion sécurisée des sessions, nous garantissons la protection des données et la confidentialité des utilisateurs tout au long de leur expérience d'utilisation de notre application.

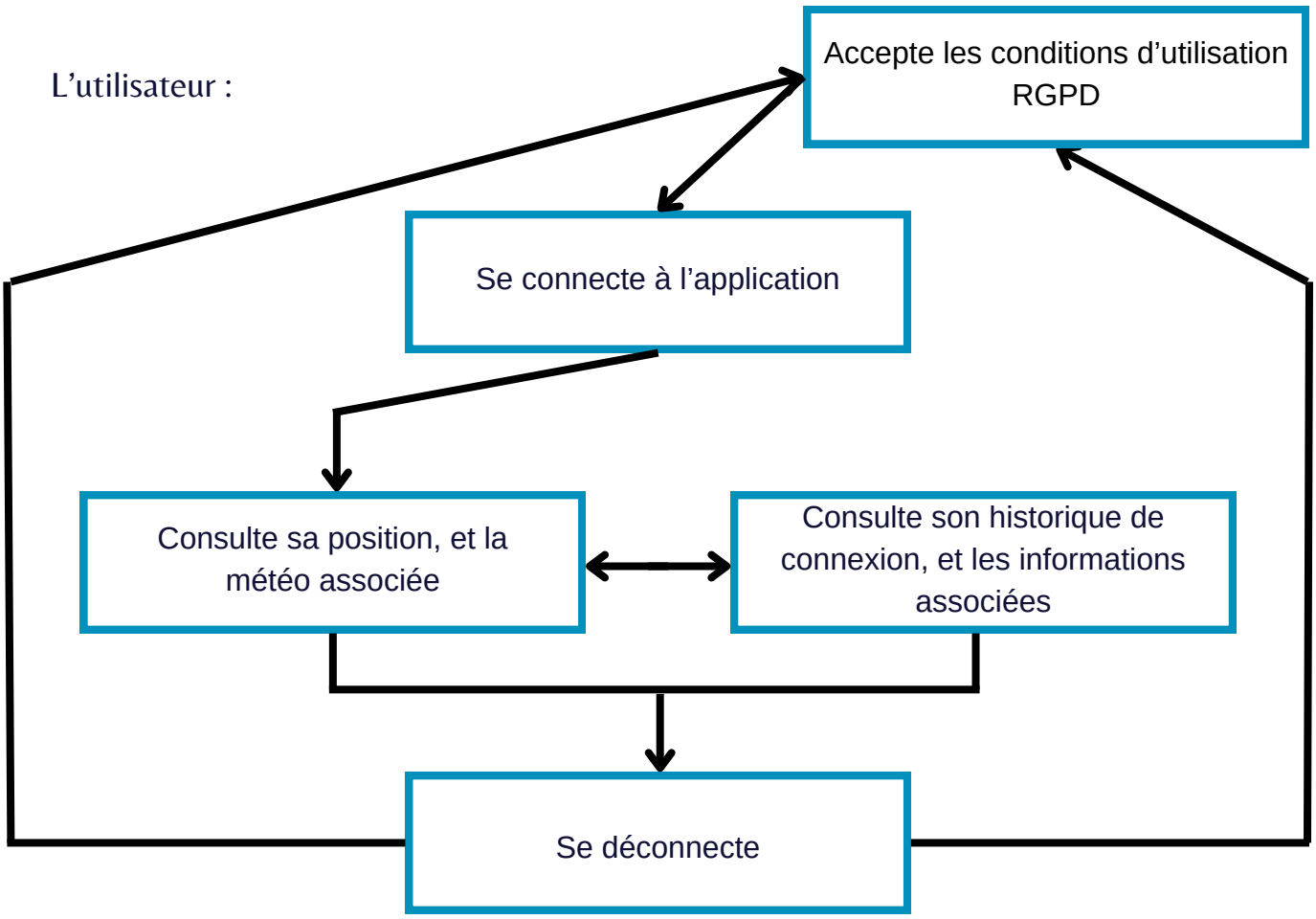
III. Fonctionnalités techniques :

Diagrammes UML

Diagrammes comportementaux

Diagramme d'activité

L'utilisateur :



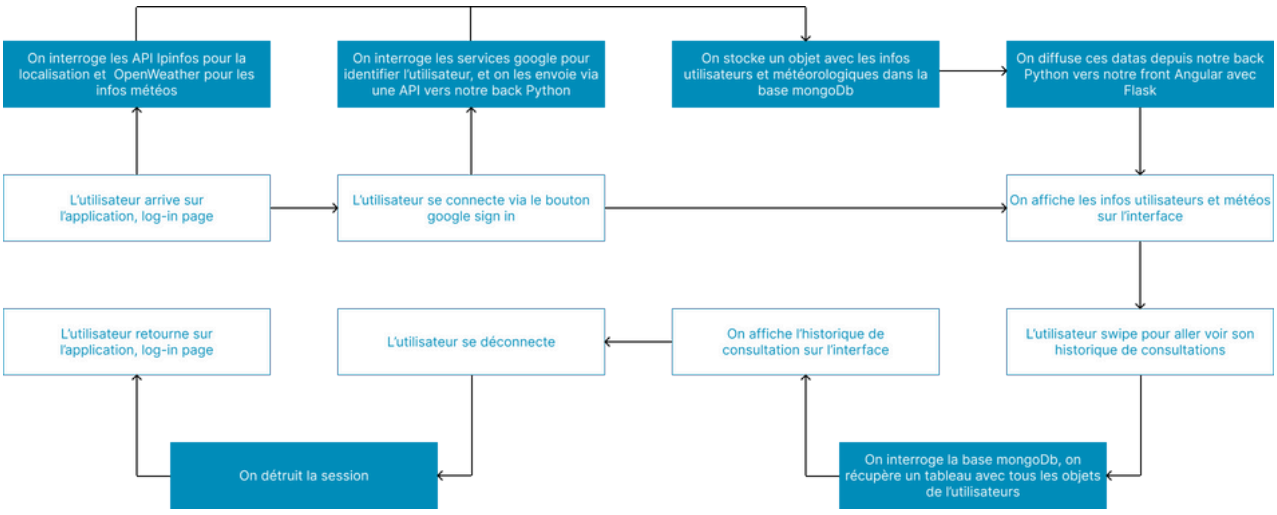
III. Fonctionnalités techniques :

Digital Campus

Diagrammes UML

Diagrammes comportementaux

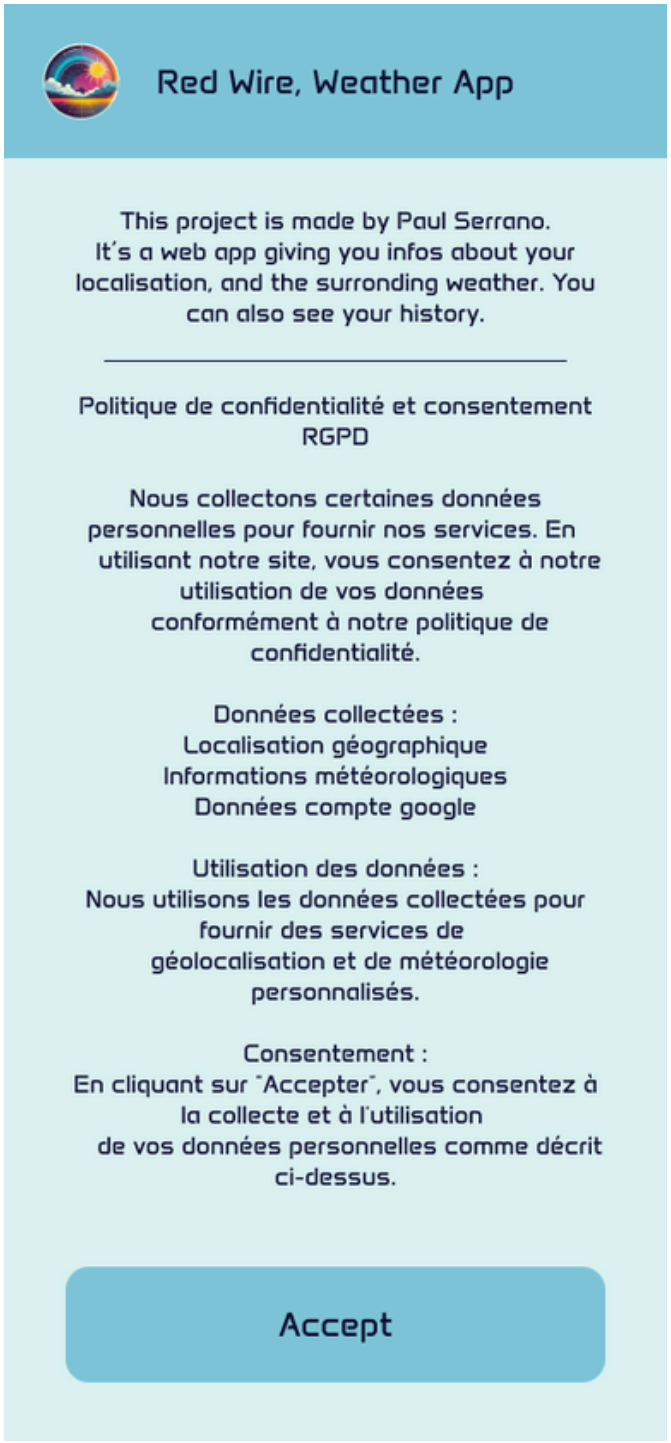
Diagramme de communication

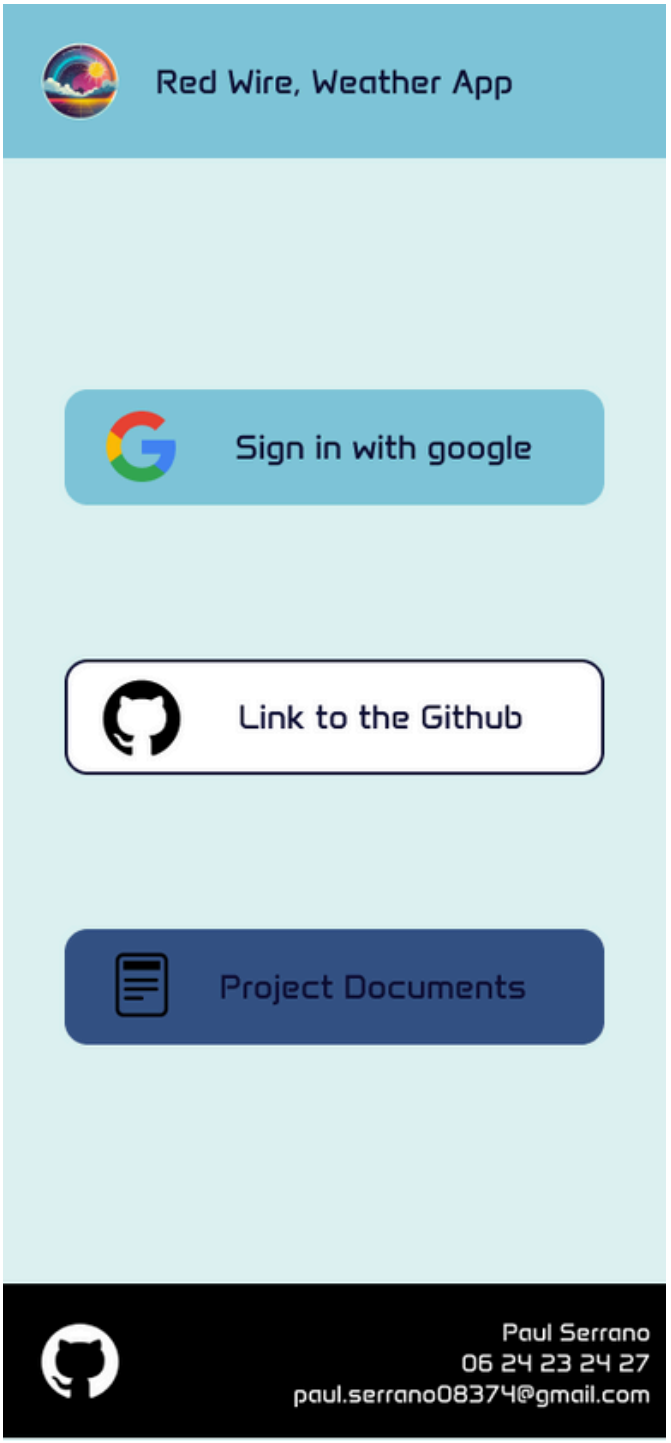


IV. Maquettes

Digital Campus

Mobile





IV. Maquettes

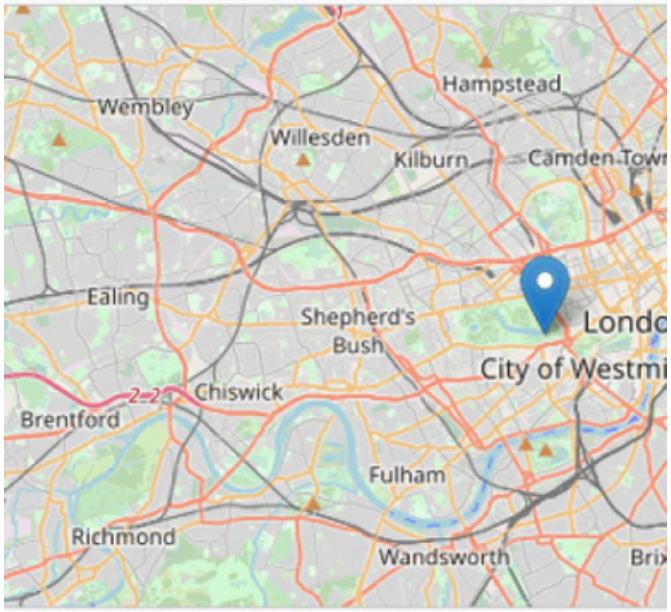
Digital Campus

Mobile

Red Wire, Weather App

Here & Now | History

Welcome in London Florent !



Weather :	Cloudy
Temperature :	11°C
Atm pressure :	25 bar

Paul Serrano
06 24 23 24 27
paul.serrano08374@gmail.com

Red Wire, Weather App

Here & Now | History

Your borwsing history !

22/04/2024 15:12:34	Paris	Cloudy 25°C
13/04/2024 19:25:56	Marseille	Sunny 30°C
02/01/2023 15:12:34	Pau	Rainy 05°C
21/12/2022 00:25:38	Rome	Cloudy 15°C

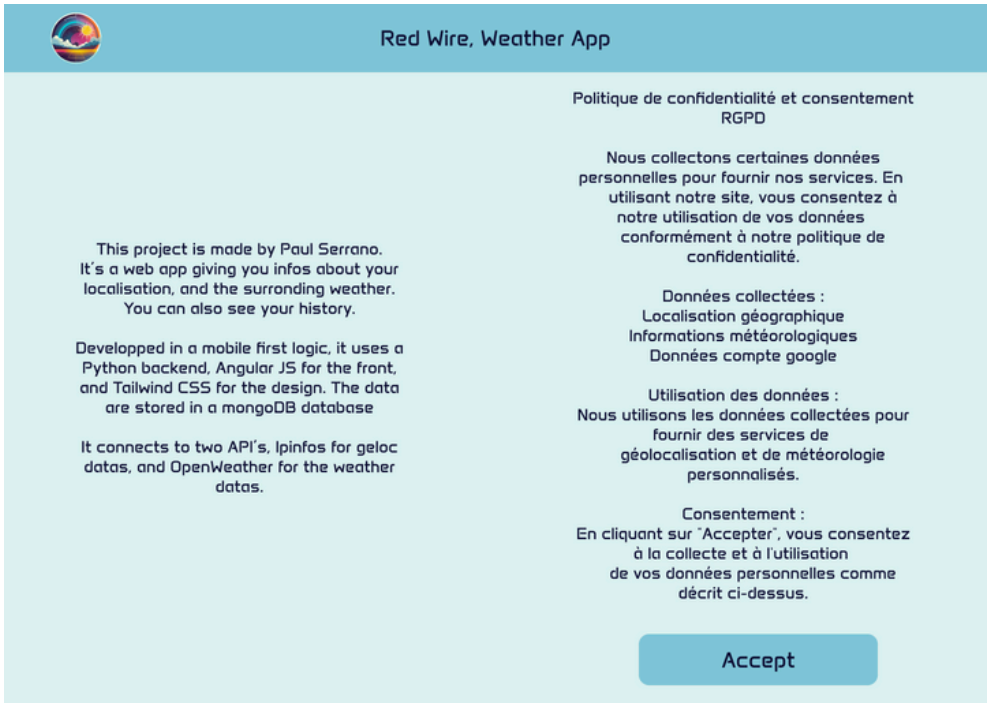
Delete your saved datas & sign out

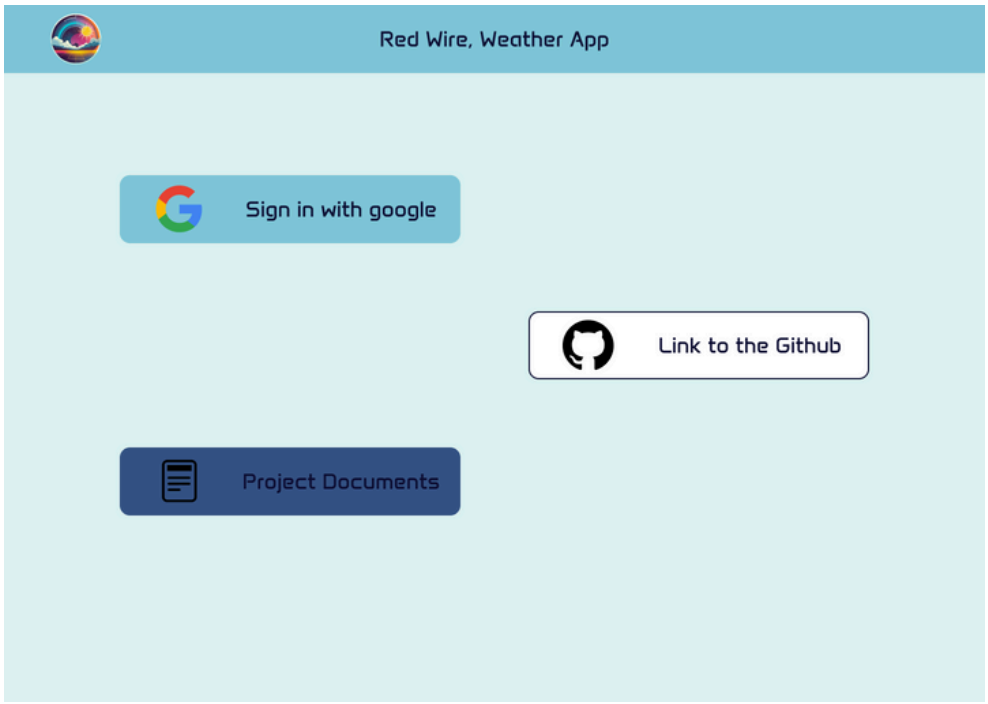
Paul Serrano
06 24 23 24 27
paul.serrano08374@gmail.com

IV. Maquettes

Digital Campus

Desktop

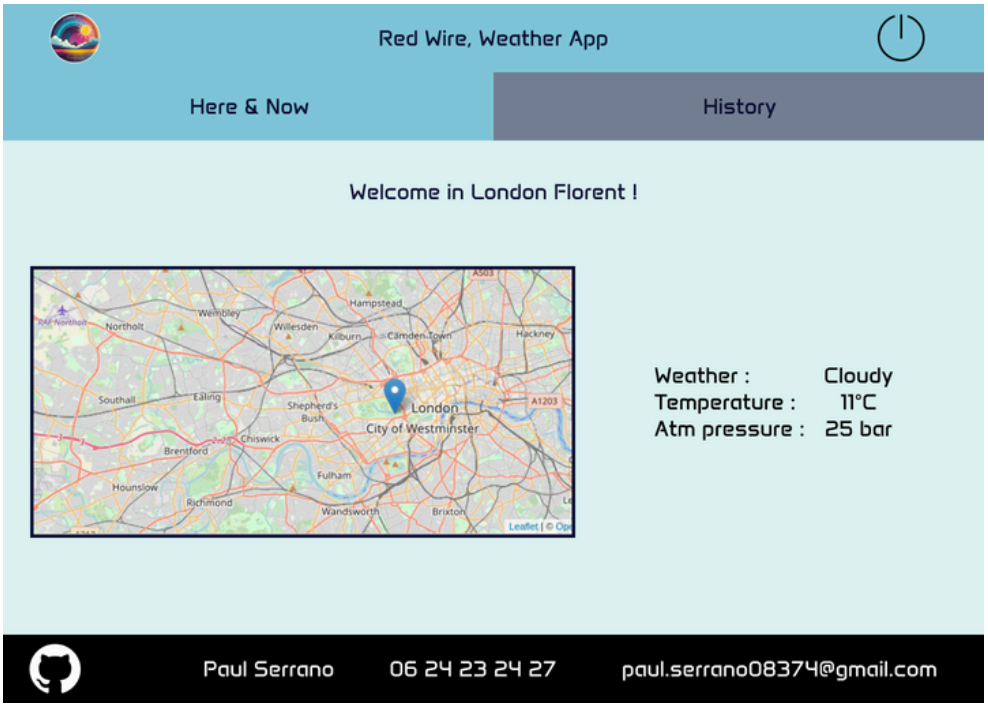


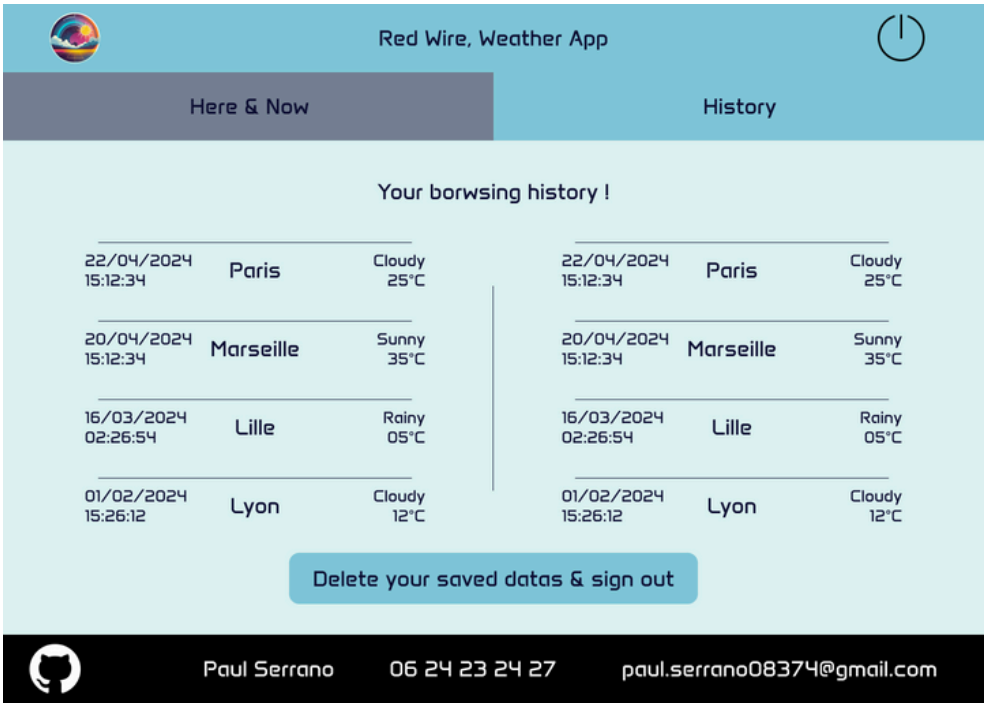


IV. Maquettes

Digital Campus

Desktop





V. Eco-conception :

Stratégie d’éco-responsabilité

Dans un contexte où la sensibilisation à l'environnement est devenue une préoccupation cruciale, il est impératif d'intégrer des pratiques éco-responsables dans le développement et le fonctionnement de nos solutions digitales. Cette section abordera la stratégie d'éco-responsabilité, mettant en lumière les mesures prises pour réduire l'empreinte environnementale de l’application

Choix de Technologies Durables :

- Sélection de Technologies à Faible Impact :

Nous avons opté pour des technologies et des infrastructures cloud réputées pour leur efficacité énergétique et leur faible impact environnemental. Par exemple, nous avons choisi des fournisseurs de cloud qui s'engagent à utiliser des sources d'énergie renouvelable et à réduire leur consommation énergétique globale.

- Utilisation de Matériaux Durables :

Dans la mesure du possible, nous privilégions l'utilisation de matériaux et de composants durables dans le développement de notre solution. Par exemple, nous favorisons l'utilisation d'équipements électroniques économes en énergie et de serveurs basse consommation.

Optimisation des Ressources :

- Gestion Efficace des Serveurs :

Nous avons mis en place des stratégies pour optimiser l'utilisation des serveurs et réduire la consommation d'énergie. Cela inclut l'utilisation de technologies de virtualisation pour maximiser l'utilisation des ressources matérielles et la mise en place de politiques de gestion de l'énergie pour minimiser la consommation en période d'inactivité.

V. Eco-conception :

Stratégie d’éco-responsabilité

- Réduction de la Consommation de Bande Passante :

Nous avons optimisé la transmission des données entre le back-end et le front-end pour réduire la consommation de bande passante. Cela inclut la compression des données, la mise en cache côté client et l'utilisation de techniques de chargement asynchrone pour minimiser les temps de latence et maximiser l'efficacité des transferts de données.

Sensibilisation et Éducation :

- Formation des Utilisateurs :

Nous avons mis en place des programmes de formation et de sensibilisation pour éduquer les utilisateurs sur les bonnes pratiques en matière d'utilisation éco-responsable de notre application. Cela inclut des conseils sur la réduction de la consommation d'énergie des appareils, la gestion des données et la sensibilisation à l'impact environnemental des technologies digitales.

- Promotion de l'Éco-responsabilité :

Nous encourageons activement nos utilisateurs à adopter des comportements éco-responsables dans leur utilisation quotidienne de notre application. Par exemple, nous mettons en place des fonctionnalités permettant aux utilisateurs de visualiser et de suivre leur propre empreinte environnementale, les incitant ainsi à prendre des mesures pour la réduire.

Notre stratégie d'éco-responsabilité reflète notre engagement envers la durabilité environnementale et la protection de notre planète. En intégrant des pratiques éco-responsables dans tous les aspects de notre solution digitale, nous nous efforçons de réduire notre empreinte environnementale et de contribuer à la préservation de l'environnement pour les générations futures.

VI. Conclusion :

Problématiques rencontrées

- Problèmes de CORS :

Les requêtes entre le front-end Angular et le back-end Python ont rencontré des problèmes de politique de partage des ressources entre origines différentes (CORS). Cela a nécessité la mise en place de configurations spécifiques pour autoriser les requêtes entre les deux parties de l'application.

- Rechargement de la page pour afficher la carte :

L'affichage de la carte nécessitait un rechargement de la page après la connexion de l'utilisateur. Cela peut affecter l'expérience utilisateur en entraînant une perte de contexte et en augmentant le temps de chargement global de l'application.

- Problèmes de routage avec des composants imbriqués :

Les composants imbriqués, tels que Weather-Now et Weather-History, ont rencontré des difficultés de routage dans l'architecture de l'application Angular, notamment dans le composant parent Browse. Cela peut avoir entraîné des problèmes d'organisation de l'application et de gestion de l'état.
J’ai résolu ce soucis avec une meilleure gestion du routage.

VI. Conclusion :

Perspectives d’amélioration

- Options de filtre et de tri pour l'historique de consultation :

Permettre aux utilisateurs de filtrer et de trier l'historique des consultations en fonction de différents critères tels que la date, la localisation ou les conditions météorologiques. Cela améliorerait l'expérience utilisateur en offrant une meilleure organisation et une navigation plus intuitive dans l'historique.

- Fonction de suppression sélective des consultations :

Ajouter la possibilité pour les utilisateurs de supprimer une ou plusieurs consultations de leur historique, en leur permettant de sélectionner les consultations à supprimer. Cela offrirait plus de contrôle aux utilisateurs sur leurs données consultées et contribuerait à la gestion de l'historique de manière plus personnalisée.

- Statistiques sur les consultations :

Intégrer des fonctionnalités de génération de statistiques sur les endroits les plus consultés, la température moyenne consultée, les tendances météorologiques observées, etc. Ces statistiques pourraient fournir des insights intéressants aux utilisateurs sur leurs habitudes de consultation et sur les tendances météorologiques locales.

- Amélioration de l'interface utilisateur :

Continuer à peaufiner l'interface utilisateur en s'appuyant sur les retours des utilisateurs pour rendre l'expérience encore plus fluide et intuitive. Cela pourrait inclure des ajustements de conception, une meilleure disposition des éléments et l'optimisation de la navigation pour une expérience utilisateur plus agréable.

- Intégration de fonctionnalités météorologiques avancées :

Explorer la possibilité d'intégrer des fonctionnalités météorologiques avancées telles que les prévisions à long terme, les alertes météorologiques en temps réel, ou des informations détaillées sur les conditions météorologiques spécifiques (par exemple, la vitesse du vent, l'humidité, etc.).